С. Н. Талипов

СОВРЕМЕННОЕ ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ НА JAVA В СРЕДЕ NETBEANS

Кереку
Б. АСПАСЫ
Павлодар

Министерство образования и науки Республики Казахстан

Павлодарский государственный университет
имени С. Торайгырова

С. Н. Талипов

СОВРЕМЕННОЕ ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ НА JAVA В СРЕДЕ NETBEANS

Учебное пособие

Павлодар
Кереку
2019

УДК 004.32 (075.8)
ББК 32.973Я73
Т16

Рекомендовано к изданию Ученым советом Павлодарского государственного университета имени С. Торайгырова

Рецензенты:

Н. А. Испулов – кандидат физико-математических наук, доцент, Павлодарский государственный университет имени С. Торайгырова;

А. Ж. Асамбаев – кандидат технических наук, доцент, Павлодарский государственный педагогический университет;

А. Ж. Кинтонова – кандидат технических наук, доцент, Евразийский университет имени Л. Н. Гумилева.

Талипов С. Н.

Т16 Современное визуальное программирование на Java в среде NetBeans / С. Н. Талипов. – Павлодар : Кереку, 2019. – 194 с.

В учебном пособии рассматриваются вопросы создания кроссплатформенных программ с графическим интерфейсом на современном языке программирования Java в среде разработки NetBeans.

Учебное пособие представляет интерес для студентов специальностей «Вычислительная техника и программное обеспечение», «Информационные системы», «Информатика», а также для всех желающих, которые хотят войти в профессиональный мир программирования на Java.

УДК 004.32 (075.8)
ББК 32.973Я73

Введение

В настоящее время язык программирования Java является основной промышленной платформой для разработки кроссплатформенных программ, больших корпоративных систем, программ для мобильных устройств на Android и встроенных систем в технические устройства.

Для освоения языка программирования Java существует несколько передовых инструментальных средств разработки, лучшая из которых для высшей школы – это NetBeans. Это связано с тем, что данная среда имеет локализированный интерфейс, поддерживает визуальную разработку всех видов Java-программ, имеет встроенные средства работы с базами данных, web-службами, серверами и др. NetBeans позволяет проводить автоматический рефакторинг и улучшение программ, производить совместную разработку проектов с контролем версий, автоматически создавать документацию, отлаживать проекты и много другое.

Многие учебники и пособия по Java ограничиваются разработкой консольных приложений и примерами для устаревших сред разработки. Это не позволяет начинающим войти в мир программирования и делать хотя бы простые программы с приемлемым графическим интерфейсом.

В данном учебном пособии приведен необходимый авторский материал, который позволяет студентам разрабатывать кроссплатформенные программы на Java с графическим интерфейсом. В конце пособия приведены задания с вариантами для выполнения.

Основная особенность данного пособия заключается в том, что акцент делается на быструю начальную разработку программ, а не на глубоком начальном изучении ООП. Данный подход оправдывает себя в современных реалиях и позволяет в дальнейшем без проблем изучать и осваивать более сложный теоретический материал и философию программирования на Java.

Данное пособие предназначается для студентов-программистов, а также всех желающих, которые хотят войти в профессиональный мир программирования на Java.

В качестве дополнения к данному пособию имеется его расширенный электронный вариант в виде учебника для Windows, Linux и Android.

Вопросы, замечания и предложения можно отправлять на адрес taliposn@gmail.com

1 Основные сведения

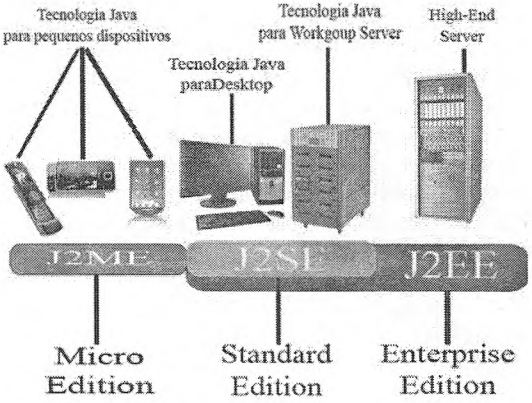
1.1 Особенности Java

Java (произносится Джава; иногда – Ява) – объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle).

Приложения Java компилируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине (JVM) вне зависимости от компьютерной архитектуры. Дата официального выпуска – 23 мая 1995 года.

Эмблемой Java является чашечка с кофе.

Язык программирования Java произошел от языка «OAK», что в переводе означает «Дуб». После своего появления язык Java начал развиваться по нескольким направлениям:

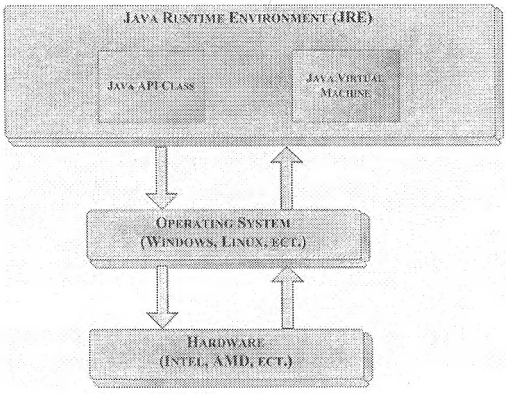


Java 2 Micro Edition, сокращенно J2ME – это редакция языка Java для разработки приложений для микрокомпьютеров (мобильных телефонов, Palm и т.д.). Сейчас имеет популярность в связи с развитием мобильных микропроцессорных устройств. В нее входят "облегченные" стандартные классы и классы для написания мидлетов (Midlets). Мидлеты специально разрабатываются для небольших устройств, в них поддерживается графика, звук, реакция на события (нажатие кнопок и т.д.). Java ME наиболее полно соответствует начальному предназначению Java – платформы для написания программ для бытовых устройств.

Java 2 Standart Edition, сокращенно J2SE – это стандартная редакция языка Java, используемая для разработки обычных Java приложений. Используя данную редакцию можно создавать консольные приложения и приложения с графическим интерфейсом пользователя. Часто встречается аббревиатура J2SE, которая подразумевает Java 2 Standart Edition.

Java 2 Enterprise Edition, сокращенно J2EE – это редакция языка Java для разработки распределенных приложений масштаба предприятия (корпоративных приложений). Данная редакция включает в себя технологию Enterprise Java Beans (EJB), Java Server Pages (JSP) и сервлеты (Servlets). На данный момент J2EE и .Net сейчас два основных соперника на рынке решений для разработки корпоративных приложений.

Механизм исполнения программ на Java включает в себя виртуальную машину Java, операционную систему и аппаратное обеспечение:

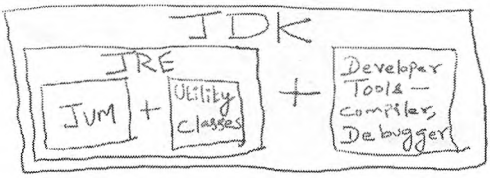


Java Runtime Environment, сокращенно JRE – это исполнительная среда Java, в которой выполняются программы, написанные на этом языке. Среда состоит из виртуальной машины – Java Virtual Machine (JVM) и библиотеки Java классов. По сути это минимальная реализация виртуальной машины, необходимая для исполнения Java приложений, без компилятора и других средств разработки.

Java Virtual Machine, сокращенно JVM – это виртуальная машина Java – основная часть исполняющей среды JRE. Виртуальная машина Java интерпретирует и исполняет байт-код Java. Байт код получают посредством компиляции исходного кода программы с помощью компилятора Java (стандартный – javac). В отличие от классических runtime-библиотек, библиотеки Java-классов входят в состав JRE.

Java Development Kit, сокращенно JDK – это бесплатно распространяемый корпорацией Oracle комплект разработчика приложений на языке Java, включающий в себя компилятор Java (javac), стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java (JRE). В состав JDK не входит интегрированная среда разработки на Java (IDE), поэтому разработчик, использующий только JDK, вынужден использовать внешний текстовый редактор и компилировать свои программы, используя утилиты командной строки.

Несмотря на то, что JRE входит в состав JDK, фирма Oracle распространяет этот набор и отдельным файлом. Это вызвано тем, что установка JRE является необходимым и достаточным условием для выполнения Java-программ. Однако для разработки программ JRE недостаточно, необходимо установить пакет JDK, который может установить и JRE и дополнительные компоненты.



Современные интегрированные среды разработки, такие как NetBeans, Oracle JDeveloper, IntelliJ IDEA, Eclipse служат для удобной разработки программного обеспечения на Java. Они опираются на

сервисы, предоставляемые JDK, и вызывают для компиляции Java-программ компилятор командной строки из комплекта JDK. Поэтому эти среды разработки либо включают в комплект поставки одну из версий JDK, либо требуют для своей работы предварительной установки JDK на машине разработчика.

Таким образом, для разработки программ на Java достаточно установить JRE+JDK+NetBeans, а только лишь для запуска готовой программы на машине пользователя достаточно установить одну JRE.

Запуск готовых java-программ (с расширением jar) из командной строки производят так:

```
java -jar JavaApplication1.jar
```

В данном примере запускается на выполнение программа JavaApplication1.jar.

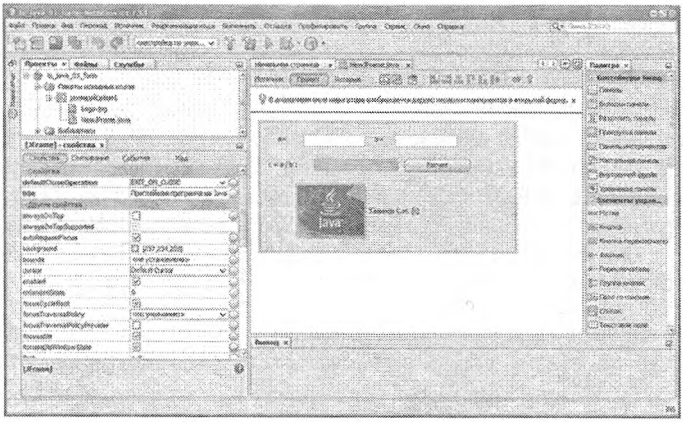
1.2 Среда разработки приложений NetBeans

NetBeans IDE – свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, Python, PHP, JavaScript, C, C++, и ряда других.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведется независимым сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org.

По качеству и возможностям последние версии NetBeans IDE не уступают лучшим коммерческим (платным) интегрированным средам разработки для языка Java, таким, как IntelliJ IDEA, поддерживая рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету, множество предопределённых шаблонов кода и др.

Для разработки программ в среде NetBeans и для успешной инсталляции и работы самой среды NetBeans должен быть предварительно установлен JDK или J2EE SDK подходящей версии. Среда разработки NetBeans поддерживает разработку для платформ J2SE и J2EE и для мобильных платформ J2ME.



Рабочая папка проектов по-умолчанию:
C:\Documents and Settings\<ИМЯ_ПОЛЬЗОВАТЕЛЯ>\Мои документы\NetBeansProjects.

Запуск программы на выполнение: Заходим в меню "Выполнить" – "Запустить проект" (F6).

Закрытие проекта: Заходим в меню "Файл" – "Закрыть проект" (Для закрытия необходимо предварительно выделить проект мышью в окне «Проекты»).

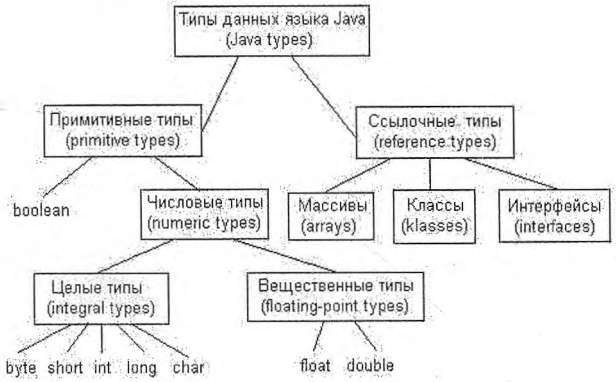
Сохранение проекта: Заходим в меню "Файл" – "Сохранить все" (Ctrl+Shift+S).

Открытие ранее сохраненного проекта: Заходим в меню "Файл" – "Открыть проект" (Ctrl+Shift+O).

Настройка конфигурации проекта: Заходим в меню "Выполнить" – "Установить конфигурацию проекта – Настроить", далее выбираем категорию «Выполнение» – Выбираем главный класс для запуска программы – «ОК».

Получение готового исполняемого jar-файла: заходим в меню "Выполнить" – "Очистить и собрать проект" (Shift+F11). Готовый jar-файл находится в папке "dist" проекта.

1.3 Простые типы данных



Целые типы. Служит для хранения целых чисел.

Тип	Размер, бит	Минимальное значение	Максимальное значение
byte	8	-128	127
short	16	-32768	32767
int	32	-2147483648	2147483647
long	64	-922372036854775808	922372036854775807

```
byte b1 = 50, b2 = -99, b3;
byte a1 = 0xF1, a2 = 0x07;
short det = 0, ind = 1;
int i = -100, j = 100, k = 9999;
long big = 50;
```

Оператор	Название	Пример	Примечание
+	Оператор сложения	i+j	В случае, когда операнды i и j имеют разные типы или типы byte, short или char, действуют правила автоматического преобразования типов
–	Оператор вычитания	i-j	
*	Оператор умножения	i*j	
/	Оператор деления	i/j	Результат округляется до целого путем отбрасывания дробной части как для положительных, так и для

			отрицательных чисел
%	Оператор остатка от целочисленного деления	i%j	Возвращается остаток от целочисленного деления
=	Оператор присваивания	v=i	Сначала вычисляется выражение i, после чего полученный результат копируется в ячейку v
++	Оператор инкремента (увеличения на 1)	v++	v++ эквивалентно v=v+1
--	Оператор декремента (уменьшения на 1)	v--	v-- эквивалентно v=v-1
+=		v+=i	v+=i эквивалентно v=v+i
-=		v-=i	v-=i эквивалентно v=v-i
*=		v*=i	v*=i эквивалентно v=v*i
/=		v/=i	v/=i эквивалентно v=v/i
%=		v%=i	v%=i эквивалентно v=v%i

Символьный тип. Служит для хранения одного символа.

Тип	Размер, бит	Минимальное значение	Максимальное значение
char	16	0	65536

```
char c1 = 'A', c2 = '?', newLine = '\n';
char s2 = '\u0042';
```

Escape-последовательность	Функция	Значение в Unicode
\b	Забой (backspace)	\u0008
\t	Горизонтальная табуляция (horizontaltab)	\u0009
\n	Перевод строки (linefeed)	\u000A
\f	Перевод страницы (form feed)	\u000C
\r	Возврат каретки (carriage return)	\u000D
\"	Двойная кавычка (double quote)	\u0022
\'	Апостроф (single quote)	\u0027
\\	Обратная косая черта (backslash)	\u005C

Вещественные типы. Служат для хранения целых и вещественных чисел.

Тип	Разрядность (бит)	Диапазон	Точность
float	32	3,4e-38 < |x| < 3,4e38	7-8 цифр
double	64	1,7e-308 < |x| < 1,7e308	17 цифр

float x = 0.001, y = -34.789;
double 21 = -16.2305, z2;
float x1 = 3.5f, x2 = 3.7E6f, x3 = -1.8E-7f;

Оператор	Название	Пример	Примечание
+	Оператор сложения	x+y	В случае, когда операнды x и y имеют разные типы, действуют правила автоматического преобразования типов.
–	Оператор вычитания	x-y	
*	Оператор умножения	x*y	
/	Оператор деления	x/y	Результат является вещественным. В случае, когда операнды x и y имеют разные типы, действуют правила автоматического преобразования типов.
%	Оператор остатка от целочисленного деления	x%y	Возвращается остаток от целочисленного деления x на y. В случае, когда операнды x и y имеют разные типы, действуют правила автоматического преобразования типов.
=	Оператор присваивания	v=x	Сначала вычисляется выражение x, после чего полученный результат копируется в ячейку v
++	Оператор инкремента(увеличения на 1)	v++ ++v	эквивалентно v=v+1
--	Оператор декремента(уменьшения на 1)	v-- --v	эквивалентно v=v-1
+=		v+=x	эквивалентно v=v+x
-=		v-=x	эквивалентно v=v-x
*=		v*=x	эквивалентно v=v*x
/=		v/=x	эквивалентно v=v/x
%=		v%=x	эквивалентно v=v%x

Математические функции, а также константы "пи" (Math.PI) и "е" (Math.E) заданы в классе Math, находящемся в пакете java.lang.

Для того чтобы их использовать, надо указывать имя функции или константы, квалифицированное впереди именем класса Math.

Оператор класса Math	Примечание
Тригонометрические и обратные тригонометрические функции	
sin(x)	sin(x) - синус
cos(x)	cos(x) - косинус
tan(x)	tg(x) - тангенс
asin(x)	arcsin(x) - арксинус
acos(x)	arccos(x) – арккосинус
atan(x)	arctg(x) – арктангенс
atan2(y, x)	Возвращает угол, соответствующий точке с координатами x,y, лежащий в пределах
toRadians(angdeg)	angdeg / 180.0 * PI; - перевод углов из градусов в радианы
toDegrees(angrad)	angrad * 180.0 / PI; – перевод углов из радиан в градусы
Степени, экспоненты, логарифмы	
exp(x)	– экспонента
expm1(x)	. При x, близком к 0, дает гораздо более точные значения, чем
log(x)	ln(x) – натуральный логарифм
log10(x)	– десятичный логарифм
log1p(x)	. При x, близком к 0, дает гораздо более точные значения, чем
sqrt(x)	– квадратный корень
cbrt(x)	– кубический корень
hypot(x,y)	– вычисление длины гипотенузы по двум катетам
pow(x, y)	– возведение x в степень y
sinh(x)	– гиперболический синус
cosh(x)	– гиперболический косинус
tanh(x)	– гиперболический тангенс
Модуль, знак, минимальное, максимальное число	
abs(m) abs(x)	Абсолютное значение числа. Аргумент типа int, long, float или double. Результат того же типа, что аргумент
signum(a) signum(x)	Знак числа. Аргумент типа float или double. Результат того же типа, что аргумент
min(m,n) min(x,y)	Минимальное из двух чисел. Аргументы одного типа. Возможны типы: int, long, float, double. Результат того же типа, что аргумент
max(m,n)	Максимальное из двух чисел. Аргументы одного типа.

max(x,y)	Возможны типы: int, long, float, double. Результат того же типа, что аргумент
Округления	
ceil(x)	Ближайшее к x целое, большее или равное x
floor(x)	Ближайшее к x целое, меньшее или равное x
round(a) round(x)	Ближайшее к x целое. Аргумент типа float или double. Результат типа long, если аргументdouble, и типа int – если float. То же, что (int)floor(x + 0.5).
rint(x)	Ближайшее к x целое.
ulp(a) ulp(x)	Расстояние до ближайшего большего чем аргумент значения того же типа ("дискретность" изменения чисел в формате с плавающей точкой вблизи данного значения). Аргумент типа float или double. Результат того же типа, что аргумент
Случайное число, остаток	
random()	Псевдослучайное число в диапазоне от 0.0 до 1.0. При этом [illegible]
IEEEremainder(x,y)	Остаток от целочисленного деления x/y, то есть x-y*n, где n – результат целочисленного деления

Булевский (логический) тип. Служит для хранения логического значения true («Истина») или false («Ложь»).

```
boolean a, b;
a=true; b=a; c=false;
```

Оператор	Название	Пример
&&	логическое "И" (and)	a&&b
||	логическое "ИЛИ" (or)	a||b
^	логическое "исключающее ИЛИ" (xor)	a^b
!	логическое "НЕ" (not)	!a
==	равно	a==b
!=	не равно	a!=b
>	больше	a>b
<	Меньше	a<b
>=	больше или равно	a>=b
<=	меньше или равно	a<=b

1.4. Управляющие конструкции

Определение управляющих конструкций в Java практически во всём совпадает с C++.

Условные конструкции if … else. Наиболее распространённой формой управляющих структур является конструкция if … else, синтаксис которой выглядит следующим образом:

```
if (БулевскоеВыражение) {
   Инструкции1;
} else {
   Инструкции2;
}
```

Сначала осуществляется проверка значения булевского выражения. Если результат равен true, выполняется блок Инструкции1, в противном случае (и при наличии предложения else) – блок Инструкции2. Предложение else может быть пропущено, при этом конструкция if … else принимает более краткий вид:

```
if (БулевскоеВыражение) {
   Инструкции;
}
```

```
int m = 4;
if (m == 4) {
  System.out.println("April");
}
```

run: April

В этом случае при ложном значении булевского выражения никаких операций не выполняется. Возможна также и вложенность конструкций if … else:

```
if (БулевскоеВыражение1) {
   Инструкции1
} else if (БулевскоеВыражение2) {
   Инструкции2
} else {
   Инструкции3
}
```

```
int month = 4;
String season;
```

```
if (month == 12 || month == 1 || month == 2) {
   season = "Winter";
} else if (month == 3 || month == 4 || month == 5) {
   season = "Spring";
} else if (month == 6 || month == 7 || month == 8) {
   season = "Summer";
} else if (month == 9 || month == 10 || month == 11) {
   season = "Autumn";
} else {
   season = "Bogus Month";
}
System.out.println("April is in the " + season + ".");
```

run: April is in the Spring.

Некоторым аналогом конструкции if … else является операция «?» со следующим синтаксисом:

БулевскоеВыражение ? Значение1 : Значение2

где Значение1, Значение2 – вычисляемые значения одного типа.

Результатом этой операции будет Значение1, если БулевскоеВыражение истинно, в противном случае –Значение2.

```
int m = 4; String season;
season = m == 4 ? "April" : "???";
System.out.println(season);
```

run: April

Условные конструкции switch – case. Конструкция switch позволяет передавать управление тому или иному блоку кода, обозначенному оператором case в зависимости от значения выражения:

```
switch (Выражение) {
   case Значение1:
      Инструкции;
   case Значение2:
      Инструкции;
```

```
   ...
   default:
      Инструкции;
}
```

Значение Выражения может иметь один из типов: byte, short, int, char. Каждому оператору case ставится в соответствие константа-значение. Если значение выражения совпадает со значением оператора case, то управление передаётся первой инструкции данного блока case.

Для выхода из конструкции после завершения инструкций блока используется команда break. Если не прервать исполнение командой break, то будет исполняться блок инструкций, соответствующий следующему значению.

Если значение выражения не совпало ни с одним из значений операторов case, то выполняется первая инструкция блока default. Если же метка default отсутствует, выполнение оператора switch завершается.

```
int month = 4;
String season;
switch (month) {
   case 12:
   case 1:
   case 2:
season = "зима"; break;
   case 3:
   case 4:
   case 5:
season = "весна"; break;
   case 6:
   case 7:
   case 8:
season = "лето";  break;
   case 9:
   case 10:
   case 11:
season = "осень"; break;
   default:
season = "Нет такого месяца";
}
```

```
System.out.println("Апрель - это " + season + ".");
```

run: Апрель - это весна.

Циклы for. Выражение for применяется для организации циклического перехода по значениям из заданного диапазона и в общем виде выглядит так:

for (СекцияИнициализации; БулевскоеВыражение; Секция изменения) Инструкция;

Обработка цикла происходит в следующем порядке:

- инициализация;
- проверка условия завершения;
- исполнение тела цикла;
- инкрементация счётчика.

Все секции заголовка цикла for являются необязательными. Если СекцияИнициализации опускается, на её месте остаётся только символ точки с запятой. Если же из заголовка исключается БулевскоеВыражение, в качестве значения логического условия подразумевается литерал true. Исключение всех трёх секций приводит к тому, что цикл становится «бесконечным»:

```
for (;;) {
   Инструкции;
}

for (int i = 1; i <= 10; i++) {
   System.out.println("i = " + i);
}
```

```
run:
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
```

```
i = 10

for (int n = 10; n > 0; n--) {
   System.out.println("n= " + n);
}

run:
n= 10
n= 9
n= 8
n= 7
n= 6
n= 5
n= 4
n= 3
n= 2
n= 1

int a, b;
for (a = 1, b = 4; a < b; a++, b--) {
   System.out.println("a = " + a);
   System.out.println("b = " + b);
}

run:
a = 1
b = 4
a = 2
b = 3
```

Циклы while. Синтаксис циклической конструкции while выглядит так:

```
while (БулевскоеВыражение)
Инструкция;

int n = 5;
while (n > 0) {
   System.out.println("while " + n);
   n--;
}
```

```
run:
while 5
while 4
while 3
while 2
while 1
```

Сначала осуществляется проверка булевского выражения. Если результат равен true, выполняется Инструкция (в качестве инструкции может быть использован блок), после чего булевское выражение проверяется вновь, и процесс повторяется до тех пор, пока в результате проверки не будет получено значение false.

Циклы do … while. Если требуется исполнить тело цикла хотя бы 1 раз, используется конструкция do … while:

```
do
while (БулевскоеВыражение)
```

```
int n = 5;
do {
   System.out.println("do-while " + n);
} while (--n > 0);
```

```
run:
do-while 5
do-while 4
do-while 3
do-while 2
do-while 1
```

В этом случае проверка истинности логического выражения осуществляется после выполнения тела цикла.

В теле циклов можно использовать две особые инструкции:

- break – применяется для завершения выполнения цикла;

- continue – передаёт управление в конец тела цикла (т.е. начинает следующую итерацию). В ситуациях с while и do это приводит к выполнению проверки условия цикла, а при использовании в теле for инструкция continue производит передачу управления секции изменения значения переменных цикла.

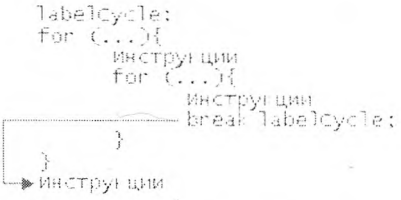
Метки. В Java отсутствует оператор goto. Но хотя программирование с применением goto и считается плохим тоном, существуют задачи, в которых его использование очень удобно. Основным примером такой ситуации является совершение выхода одновременно из двух вложенных циклов.

Эта проблема решается в Java использованием меток. Любая инструкция в программе может быть снабжена меткой, которая представляет собой содержательное имя, позволяющее сослаться на соответствующую инструкцию.

метка: инструкция

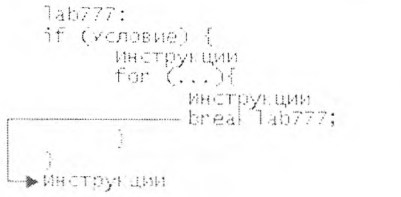
Несмотря на то, что метка может быть установлена перед любой инструкцией, на практике имеет смысл применять метки только перед циклическими конструкциями for, while, do, условными конструкциями if, switch и блоками {}. Чтобы «выбраться» из вложенного цикла или блока, достаточно снабдить меткой соответствующий внешний блок и указать её в команде break, которая передаёт управление первой инструкции, следующей за блоком.

Примеры:



В данном случае break осуществит выход сразу из двух циклов.

Ещё один пример выхода из сложной конструкции:



```
boolean t = true;
a:
{
  b:
  {
c:
{
  System.out.println("До break");
  if (t) {
break b;
}
  System.out.println("Не будет выполнено ");
}
System.out.println("Не будет выполнено ");
}
  System.out.println("После break");
}

run:
До break
После break
```

Выход из методов (процедур). В Java для реализации выхода из методов (процедур) используется метод return, который приведет к немедленному завершению работы и передаче управления коду, вызвавшему этот метод. Ниже приведен пример, иллюстрирующий использование оператора return:

```
boolean t = true;
System.out.println("До return");
if (t) { return; }
System.out.println("Это не будет выполнено ");

run:
До return
```

Для выхода из программы используется метод System.exit(КодОшибки). Если программа заканчивается нормально, то КодОшибки должен быть 0:

```
System.exit(0);
```

1.5 Одномерные статические массивы

Одномерные массивы служат для хранения линейного списка с данными. Статические массивы поддерживаются на уровне Java-синтаксиса. Число элементов указывается явно при создании объекта массива или определяется автоматически при перечислении элементов массива. По умолчанию элементы массива объектов устанавливаются в null или в 0 для простых типов. Индексация элементов начинается с 0. Размер массива определяется через функцию length.

Рассмотрим пример работы с одномерными статическими массивами:

```
package tsn01.array;
import java.util.Arrays;
public class TSN01_Array {
  public static void main(String[] args) {
    // Пример работы с одномерными статическими массивами
    int a[], b[]; // Переменные-массивы целых чисел
    a = new int[10];  // Создание массива без инициализации
    b = new int[]{1, 2, 3, 4}; // Создание массива с инициализацией
    String s[] = new String[] {"Hello ", "world", "!!!"}; // Описываем и создаем массив строк
    Arrays.fill(a, 0); // Заполнить массив нулями
    a[0] = 20; a[1] = 10; a[2] = 5; a[3] = 33; // Устанавливаем значения 4 элементам
    Arrays.fill(a, 5,10,-1); // Присвоить с 5 по 9 (10-1) элементам значение "-1"
    Arrays.sort(a); // Сортируем массив
    b[3] = b[1]*0b11+b[2]*0x2; // Рассчитываем значение для 4 элемента
    System.out.println(Arrays.toString(a)); // Вывод на экран массива "a"
    System.out.println(Arrays.toString(b)); // Вывод на экран массива "b"
    System.out.println(Arrays.toString(s)); // Вывод на экран массива "s"
    // Вывод на экран размеров массивов
    System.out.println("Количество элементов в массиве \"a\": " + a.length);
    System.out.println("Количество элементов в массиве \"b\": " + b.length);
```

```
    System.out.println("Количество элементов в массиве \"str\": " +
s.length);
    System.out.println(s[0] + s[1]); // Доступ к элементам массива
  }
}
```

Результат работы программы:

```
[-1, -1, -1, -1, -1, 0, 5, 10, 20, 33]
[1, 2, 3, 12]
[Hello , world, !!!]
Количество элементов в массиве "a": 10
Количество элементов в массиве "b": 4
Количество элементов в массиве "str": 3
Hello world
```

1.6 Многомерные статические массивы

Многомерные статические массивы служат для хранения двухмерных таблиц с данными. Рассмотрим пример работы с двумерным статическим массивом:

```
package tsn01.matrix;
public class TSN01_Matrix {
  public static void main(String[] args) {
    // Работа с двумерным статическим массивом чисел
    // Создание исходных данных (элементов массива) и вывод их на
экран
    final int r = 4; // Количество строк
    final int c = 5; // Количество столбцов
    int m[][] = new int[r][c]; // Двумерный массив
    int k; System.out.println("Matrix:");
    for (int i = 0; i < r; i++) { // Цикл по строкам
      for (int j = 0; j < c; j++) { // Цикл по колонкам
        k = (int) Math.round(Math.random() * 100); // Получение
случайного числа
        m[i][j] = k; // Присвоение элементу массива числа
        System.out.print(String.format("%5d", m[i][j])); // Вывод на
экран элемента массива
    } System.out.println("");
  }
    // Поиск минимума и максимума в массиве
```

```
    int min = m[0][0], max = m[0][0], maxi = 0, maxj = 0, mini = 0, minj
= 0; // Задаем начальные значения мин и макс
    for (int i = 0; i < r; i++) { // Цикл по строкам
      for (int j = 0; j < c; j++) { // Цикл по колонкам
        k = m[i][j]; // Получаем элемент массива
        if (k > max) {  max = k; maxi = i;  maxj = j; } // Поиск
максимума
        if (k < min) {  min = k; mini = i;  minj = j; } // Поиск минимума
    }
  }
    // Меняем максимальный и минимальный элементы в масиве
местами
    k = m[maxi][maxj]; m[maxi][maxj] = m[mini][minj]; m[mini][minj] =
k;
    // Вывод измененного массива на экран
    System.out.println("New matrix:");
    for (int i = 0; i < r; i++) { // Цикл по строкам
      for (int j = 0; j < c; j++) { // Цикл по колонкам
        System.out.print(String.format("%5d",  m[i][j])); // Вывод на
экран элемента массива
    } System.out.println("");
  }
}
}
```

Результат работы программы:

```
Matrix:
   42   83   94   96    1
    2   64   27   32   10
   20   86   49   14   36
   12   35   14   65   97
New matrix:
   42   83   94   96   97
    2   64   27   32   10
   20   86   49   14   36
   12   35   14   65    1
```

1.7 Динамические массивы-списки

Динамические массивы реализованы на уровне параметризованных классов: Vector и ArrayList. Однако в качестве

элементов простые типы выступать не могут, допускаются только объектные типы.

Для управления элементами эти классы используют методы интерфейсов Collection и List:

- add(E o) – добавление элемента в конец;
- add(int index, E element) – вставка элемента в указанную позицию;
- remove(int index) – удаление элемента в указанной позиции;
- remove(Object o) – удаление первого вхождения объекта в списке;
- clear() – удаление всех элементов;
- isEmpty() – определяет, содержит ли список элементы;
- size() – число элементов;
- set(int index, E element) – заменить элемент в указанной позиции новым;
- get(int index) – получить элемент по указанному индексу;
- contains(Object o) – определение, содержится ли указанный объект в списке элементов;
- lastIndexOf(Object o) – поиск последнего вхождения элемента, возвращается индекс элемента или -1;
- indexOf(Object o) – поиск первого вхождения элемента, возвращается индекс элемента или -1;
- toArray() – возвращает копию в виде статического массива;
- toArray(T[] a) – сохраняет элементы в указанный массив.

Пример работы с динамическим массивом целых чисел:

```
package tsn01.arraylist;
import java.util.ArrayList;
public class TSN01_ArrayList {
   public static void main(String[] args) {
      // Работа с динамическим массивом чисел
      ArrayList<Integer> i = new ArrayList<>(); // Создание
динамического массива целых чисел
      i.add(3); // Добавление значения
      i.add(new Integer(3)); // Добавление значения
      if (i.get(0)==i.get(1)) { System.out.println("Эта строчка не
напечатается..."); }
      if (i.get(0).equals(i.get(1))) { System.out.println("3=3"); }
      i.add(12+5); // Добавление значения
      System.out.println("Размер массива: " + i.size());
```

```
    System.out.println("Элементы массива: " + i.get(0).intValue() + ", "
+ i.get(1)+ ", " + i.get(2));
  }
}
```

Результат работы программы:

```
3=3
Размер массива: 3
Элементы массива: 3, 3, 17
```

Пример работы с динамическим массивом строк:

```
package tsn01.arraylist;
import java.util.ArrayList;
public class TSN01_ArrayList {
  public static void main(String[] args) {
    // Работа с динамическим массивом строк
    ArrayList<String> pozdr = new ArrayList<>(); // Массив пожеланий
    ArrayList<String> fam = new ArrayList<>();  // Массив фамилий
    // Добавление поздравления в массив
    pozdr.add("Удачи"); pozdr.add("Здоровья"); pozdr.add("Денег");
    // добавление фамилии в массив
    fam.add("Петров"); fam.add("Сидоров"); fam.add("Иванов");
    // Проверка количества поздравлений
    if (fam.size() > pozdr.size()) { return; }
    for (int i = 0; i < fam.size(); i++) {
      // Генерируем случайное число в диапазоне от 0 до длины
массива поздравлений
      int p = (int) Math.floor(Math.random() * pozdr.size());
      // Генерация поздравления
      System.out.println("Уважаемый " + fam.get(i)
+ "! Поздравляем Вас с этим прекрасным праздником, и желаем Вам "
+ pozdr.get(p).toString().toLowerCase() + "!");
  pozdr.remove(p); // Удаляем элемент с индексом p из массива
поздравлений
 } }
}
```

1.8 Работа со строками

В Java имеется три типа строк: String, StringBuilder и StringBuffer. Статические строки «String» – обычные строки в Java, в которых нельзя изменить символы и их количество после создания строки.

Динамические строки «StringBuilder» – изменяемые строки для использования в однопоточных программах. В однопоточном использовании StringBuilder практически всегда в 1.2-1.5 раза быстрее, чем StringBuffer.

Динамические строки StringBuffer – изменяемые строки для использования в многопоточных программах. Самый медленный тип, но потокобезопасный.

Переменные типа динамических строк могут менять свои значения и длину во время выполнения программы.

Статические строки. Обычные строки в Java описываются классом String и являются статическими, т.е. в существующей строке нельзя изменить символы и их количество.

Кроме стандартного создания оператором new, строки могут быть созданы напрямую из строковой литералы. При этом в целях оптимизации, объекты созданные таким образом дополнительно сохраняются в отдельной области – строковый пул.

```
String s1 = "d" // строка будет сохранена в пуле
// строка не будет сохранена в пуле и будет уничтожена сборщиком мусора
String s2 = new String("a");
```

Один из плюсов разделения строк на статические и динамические – это повышение безопасности там, где строки используются в качестве аргументов (например, открытие баз данных, интернет соединений, механизм загрузки классов).

Операция сцепления. Для строк доступна операция +, позволяющая соединить несколько строк в одну. Если один из операндов не строка, то он автоматически преобразуется в строку. Для объектов в этих целях используется метод toString.

При каждой операции внутренне используется объект динамической строки StringBuilder или StringBuffer. Поэтому для собирания строки из нескольких все равно оптимальней использовать сразу один StringBuilder/StringBuffer.

Выделение подстроки. Есть замечание относительно метода substring – возвращаемая строка использует тот же байтовый массив,

что и исходная. Например, вы загрузили строку А из файла в 1мб. Что-то там нашли и выделили в отдельную строку Б длиной в 3 символа. Строка Б в реальности тоже занимает те же 1мб.

```
String s ="very .... long string from file";
String sub1 = s.substring(2,4); // совместно использует ту же память что и s
String sub2 = new String(s.substring(2,4)); // этот объект использует отдельный массив на 4 символа
```

Основные методы. Рассмотрим основные методы String:

- equals(Object anObject) – проверяет, идентична ли строка указанному объекту;
- compareTo(String anotherString) – лексиграфическое сравнение строк;
- compareToIgnoreCase(String str) – лексиграфическое сравнение строк без учета регистра символов;
- concat(String str) – возвращает соединение двух строк;
- contains(CharSequence s) – проверяет, входит ли указанная последовательность символов в строку;
- isEmpty() – возвращает true, если длина строки равна 0;
- indexOf(String str) – поиск первого вхождения указанной подстроки;
- replace(CharSequence target, CharSequence replacement) – замена одной подстроки другой;
- substring(int beginIndex, int endIndex) – возвратить подстроку как строку;
- toLowerCase() – преобразовать строку в нижний регистр;
- toUpperCase() – преобразовать строку в верхний регистр;
- trim() – отсечь на концах строки пустые символы;
- length() – определение длины строки;
- valueOf(a) – статические методы преобразования различных типов в строку;
- charAt(int index) – возвращает символ по указанному индексу;
- regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len) – тест на идентичность участков строк, можно указать учет регистра символов;
- regionMatches(int toffset, String other, int ooffset, int len) – тест на идентичность участков строк;
- endsWith(String suffix) – проверяет, завершается ли строка указанным суффиксом;

- startsWith(String prefix) – проверяет, начинается ли строка с указанного префикса;
- startsWith(String prefix, int toffset) – проверяет, начинается ли строка в указанной позиции с указанного префикса;
- getBytes() – возвращает байтовое представление строки;
- getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin) – возвращает символьное представление участка строки;
- hashCode() – хеш код строки;
- indexOf(int ch) – поиск первого вхождения символа в строке;
- indexOf(int ch, int fromIndex) – поиск первого вхождения символа в строке с указанной позиции;
- indexOf(String str, int fromIndex) – поиск первого вхождения указанной подстроки с указанной позиции;
- lastIndexOf(int ch) – поиск последнего вхождения символа;
- lastIndexOf(int ch, int fromIndex) – поиск последнего вхождения символа с указанной позиции;
- lastIndexOf(String str) – поиск последнего вхождения строки;
- lastIndexOf(String str, int fromIndex) – поиск последнего вхождения строки с указанной позиции;
- replace(char oldChar, char newChar) – замена в строке одного символа на другой;
- toUpperCase(Locale locale) – преобразовать строку в верхний регистр, используя указанную локализацию;
- toLowerCase(Locale locale) – преобразовать строку в нижний регистр, используя указанную локализацию;

Методы поиска возвращают индекс вхождения или -1, если искомое не найдено. Методы преобразования (как replace) не изменяют саму строку, а возвращают соответствующий новый объект строки.

Особенности String. Неправильное использование типа String приводит к засорению оперативной памяти и как следствие к медленной работе программы. Рассмотрим пример:

```
public static void main(String[] args) {
    String s = "a";
    for (int i = 0; i < 100; i++) {
        s += 'a';
    }
    System.out.println(s); // Распечатается строка из 100 символов «a»
}
```

Этот код создаст 100 разных строк, которые будут храниться в памяти, пока сборщик мусора Java не удалит их после завершения программы. В результате работы программы переменная «s» будет содержать строку из 100 символов «а», но 99 ранее созданных строк остались «мусором» в памяти без возможности обращения к ним.

Поэтому для изменения строки следует использовать класс обертку StringBuilder. Предыдущий пример нужно переписать следующим образом:

```
public static void main(String[] args) {
    StringBuilder s = new StringBuilder("a");
    for (int i = 0; i < 100; i++) {
      s.append('a');
  }
    System.out.println(s); // Распечатается строка из 100 символов «а»
}
```

Динамические строки. Динамические строки описываются классами StringBuilder и StringBuffer. StringBuffer более медленный, но потокобезопасный. Переменные типа динамических строк могут менять свои значения и длину во время выполнения программы.

Основные методы динамических строк:

- append(A) – преобразовать А в строку и добавить в конец;
- insert(int offset, A) – преобразовать А в строку и вставить ее в указанную позицию;
- delete(int start, int end) – удалить символы с указанной начальной позиции по указанную конечную позицию;
- reverse() – расположить символы в обратном порядке;
- setCharAt(int index, char ch) – заменить символ в указанной позиции;
- setLength(int newLength) – установить новый размер строки;
- substring(int start) – вернуть подстроку с указанной позиции и до конца как строку;
- substring(int start, int end) – вернуть подстроку как строку;
- deleteCharAt(int index) – удалить символ в указанной позиции;
- getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin) – сохранить последовательность символов в массив;
- indexOf(String str) – поиск первого вхождения подстроки;
- indexOf(String str, int fromIndex) – поиск первого вхождения подстроки с указанной позиции;
- lastIndexOf(String str) – поиск последнего вхождения подстроки;

- lastIndexOf(String str, int fromIndex) – поиск последнего вхождения подстроки с указанной позиции;

- replace(int start, int end, String str) – замена участка строки указанной строкой.

Пример преобразования строк. В этом примере массив символов и целое число преобразуются в объекты типа String с использованием методов этого класса:

```
package tsn01.string;
public class TSN01_String {
  public static void main(String[] args) {
    char s[] = {'J', 'a', 'v', 'a'}; // Массив символов
    String str = new String(s); // str="Java"
    if (!str.isEmpty()) {
      int i = str.length(); // i=4
      str = str.toUpperCase(); // str="JAVA"
      String num = String.valueOf(6); // num="6"
      num = str.concat("-" + num); // num="JAVA-6"
      char ch = str.charAt(2); // ch='V'
      i = str.lastIndexOf('A'); // i=3 (-1 если нет)
      num = num.replace("6", "SE"); // num="JAVA-SE"
      str.substring(0, 4).toLowerCase(); // java
      str = num + "-6";// str="JAVA-SE-6"
      String[] arr = str.split("-");
      for (String ss : arr) { // В результате будет выведен массив строк
(в 3 строчки):  JAVA SE 6
        System.out.println(ss);
    }
  } else {  System.out.println("String is empty!"); }
 }
}
```

Пример сравнение строк. В этом примере рассмотрены особенности хранения и идентификации объектов на примере вызова метода equals(), сравнивающего строку String с указанным объектом и метода hashCode(), который вычисляет хэш-код объекта (hashCode - это цифра, которая формируется для объекта по какому то правилу, например для объекта класса String по такой формуле: s[0]*31^(n-1) s[1]*31^(n-2) ... s[n-1]):

```
package tsn01.string;
public class TSN01_String {
   public static void main(String[] args) {
      String s1 = "Java";
      String s2 = "Java";
      String s3 = new String("Java");
      System.out.println(s1 + "==" + s2 + " : " + (s1 == s2)); // true
      System.out.println(s1 + "==" + s3 + " : " + (s1 == s3)); // false
      System.out.println(s1 + " equals " + s2 + " : " + s1.equals(s2)); // true
      System.out.println(s1 + " equals " + s3 + " : " + s1.equals(s3)); // true
      System.out.println(s1.hashCode());
      System.out.println(s2.hashCode());
      System.out.println(s3.hashCode());
   }
}
```

В результате на экран будет выведено:

```
Java==Java : true
Java==Java : false
Java equals Java : true
Java equals Java : true
2301506
2301506
2301506
```

Пример сортировки массива строк методом перебора:

```
package tsn01.string;
public class TSN01_String {
   public static void main(String[] args) {
      String a[] = {" Alena", "Alice ", " alina", " albina", " Anastasya",
         " ALLA ", "AnnA "}; // Массив строк
      for (int j = 0; j < a.length; j++) { // Цикл по массиву строк
         // Удаляем пробелы с концов строк и приводим к верхнему
регистру
         a[j] = a[j].trim().toLowerCase();
   }
      // Сортировка строк методом пузырька
      for (int j = 0; j < a.length - 1; j++) { // Цикл по массиву строк
         for (int i = j + 1; i < a.length; i++) { // Цикл по массиву строк
```

```
            if (a[i].compareTo(a[j]) < 0) { // Сравнение строк
                String temp = a[j]; a[j] = a[i]; a[i] = temp; // Обмен значений
в массиве строк
            }
        }
    }
        int i = -1;
        while (++i < a.length) { System.out.print(a[i] + " "); } // Вывод
массива строк на экран
}
}
```

В результате на экран будет выведено:

```
albina alena alice alina alla anastasya anna
```

Вызов метода trim() обеспечивает удаление всех начальных и конечных символов пробелов. Метод compareTo() выполняет лексикографическое сравнение строк между собой по правилам Unicode.

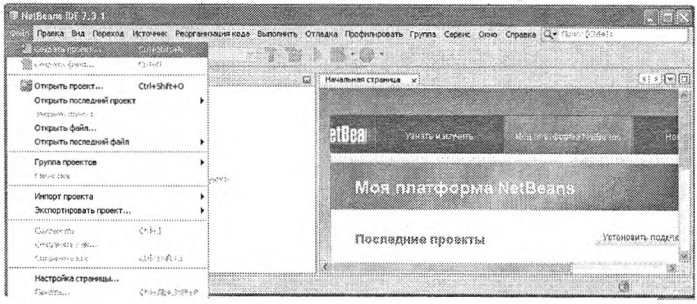
Пример работы с динамическими строками. Рассмотрим пример преобразования переменной типа «StringBuilder» к «String» через метод toString:

```
package tsn01.string;
public class TSN01_String {
    public static void main(String[] args) {
        StringBuilder s = new StringBuilder("abcd");
        s.append('e');//abcde
        s.delete(1, 2);//acde
        s.insert(1, 'b');//abcde
        s.deleteCharAt(2);//abde
        String ans = s.toString();
        System.out.println(ans); // На экран выведется "abde"
    }
}
```

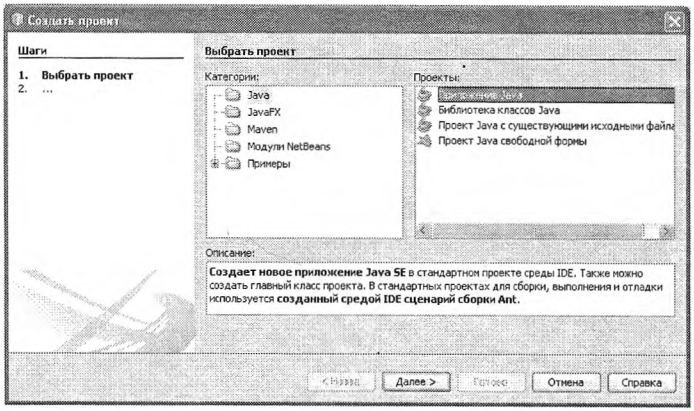
2 Простейшие программы

Консольные программы. Консольные программы на Java – это наиболее простой вид программ, не имеющих собственного графического интерфейса, весь ввод и вывод информации происходит в окне консоли. В настоящее время консольные программы не имеют особого практического применения, их заменяют программами с графическим интерфейсом.

Для создания консольной программы необходимо выбрать в меню опцию «Файл» – «Создать проект»:



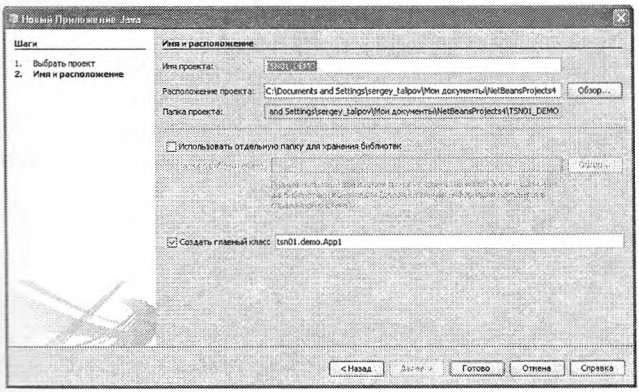
Выбрать категорию «Java» – «Приложение Java»:



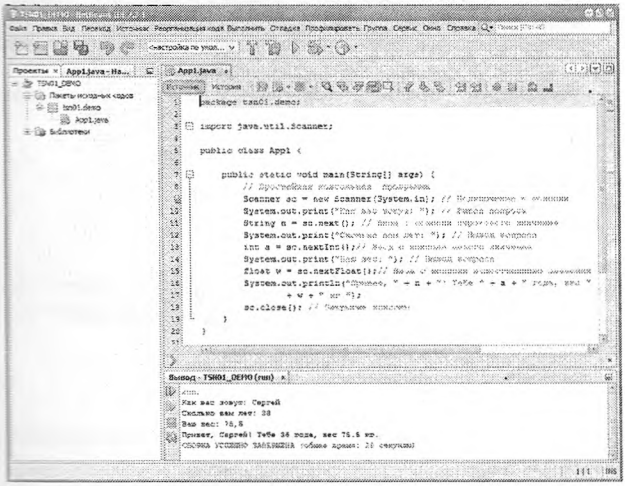
Указать имя проекта и необходимость создать главный класс. В главном классе и будет располагаться консольная программа.

Имя проекта необходимо задать так: «FIOnn_DEMO», где FIO – инициалы автора программы, nn – номер варианта, например, «TSN01_DEMO».

Название главного класса необходимо задать так: «fionn.demo.App1», где fio – инициалы автора программы, nn – номер варианта, например «tsn01.demo.App1».



Рассмотрим пример простейшей консольной программы.



```
package tsn01.demo;
import java.util.Scanner;
public class App1 {
   public static void main(String[] args) {
      // Простейшая консольная  программа
      Scanner sc = new Scanner(System.in); // Подключение к консоли
      System.out.print("Как вас зовут: "); // Вывод вопроса
      String n = sc.next(); // Ввод с консоли строкового значения
      System.out.print("Сколько вам лет: "); // Вывод вопроса
      int a = sc.nextInt(); // Ввод с консоли целого значения
      System.out.print("Ваш вес: "); // Вывод вопроса
      float w = sc.nextFloat();// Ввод с консоли вещественного значения
      System.out.println("Привет, " + n + "! Тебе " + a + " года, вес "
            + w + " кг.");
      sc.close(); // Закрытие консоли
   }
}
```

Запуск программы через командную строку Windows с поддержкой русского языка:

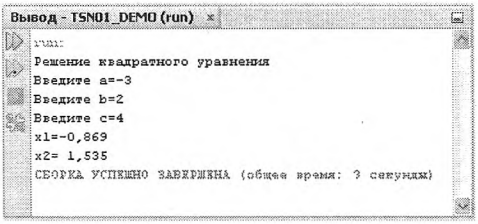
```
java -Dfile.encoding=Cp866 -jar TSN01_DEMO.jar
```

```
C:\...ои документы\NetBeansProjects\TSN01_DEMO\dist>java -Dfile.encoding=Cp866 -jar TSN01_DEMO.jar
Как вас зовут: Сергей
Сколько вам лет: 38
Ваш вес: 75.5
Привет, Сергей! Тебе 38 года, вес 75.5 кг.
```

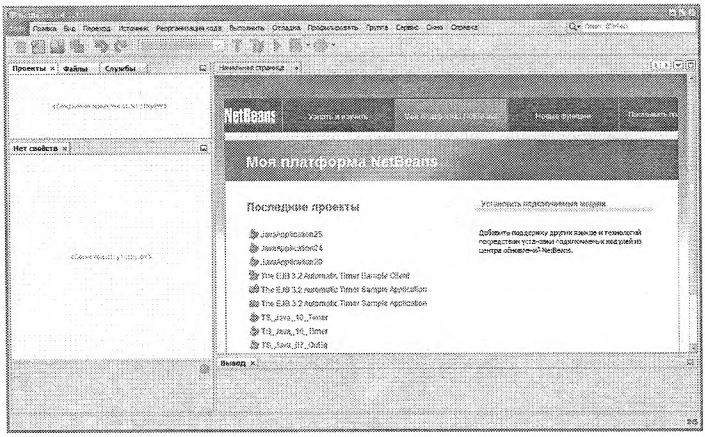
Рассмотрим пример консольной программы для решения квадратного уравнения.

```
package tsn01.demo;
import java.util.Scanner;
public class App1 {
   public static void main(String[] args) {
      // Вычисление квадратного уравнения
      double a, b, c; // Входные переменные
      double x1, x2; // Искомые значения
      double d; // Дискриминант
      try {
         Scanner sc = new Scanner(System.in); //Создаем объект для ввода данных с консоли
         System.out.println("Решение квадратного уравнения");
```

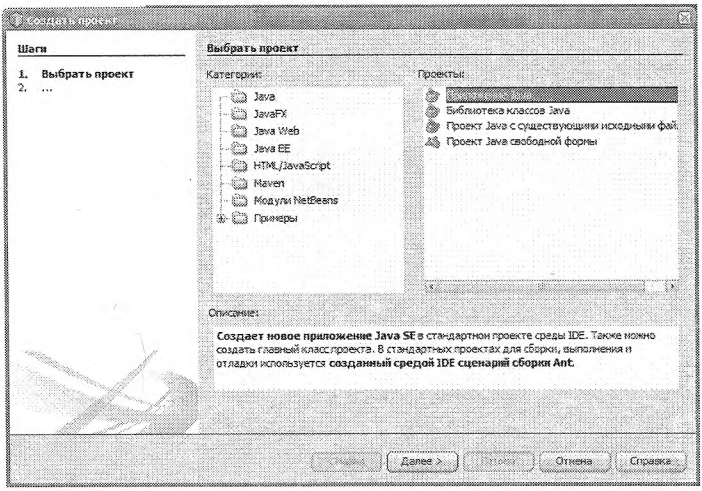
```
        System.out.print("Введите a=");
        a = sc.nextDouble(); // Ввод значения "a" с консоли
        System.out.print("Введите b=");
        b = sc.nextDouble(); // Ввод значения "b" с консоли
        System.out.print("Введите c=");
        c = sc.nextDouble(); // Ввод значения "c" с консоли
        d = (b * b) - 4 * a * c; // Расчет дискриминанта
        x1 = (-b + Math.sqrt(d)) / (2 * a); // Расчет "x1"
        x2 = (-b - Math.sqrt(d)) / (2 * a); // Расчет "x2"
        if (!(Double.isNaN(x1)) && (!Double.isInfinite(x1)) // Проверка
существования значений
                && (!(Double.isNaN(x2)) && (!Double.isInfinite(x2)))) {
            System.out.format("x1=%.3f%nx2= %.3f%n", x1, x2); // Вывод
ответа
        } else {
            System.out.println("Нет решения!"); // Нет решения
        }
        sc.close(); // Закрываем ввод с консоли
    } catch (Exception e) { // Ввели вместо цифр буквы
        System.out.println("Неверные входные данные!");
    }
  }
}
```



Простейшая программа с графическим интерфейсом в среде NetBeans. Заходим в меню "Файл" - "Создать проект":

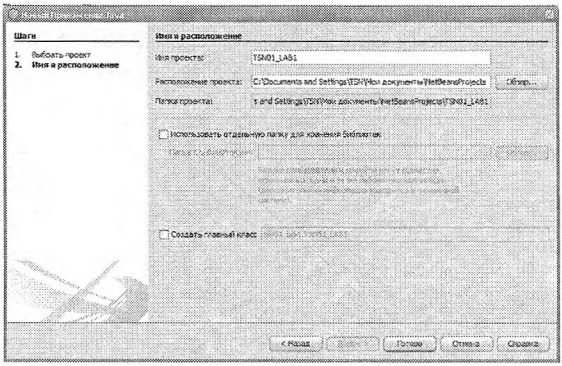


Выбираем категорию "Java", тип проекта "Приложение Java":

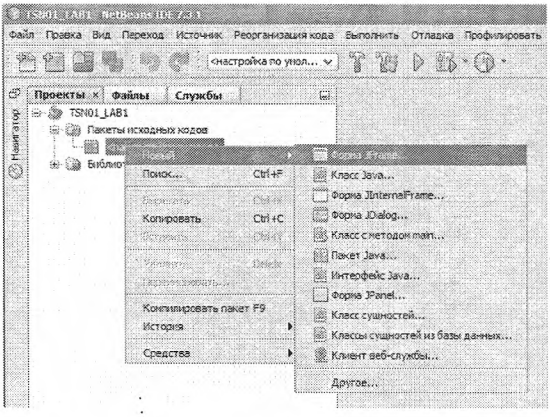


Указываем имя проекта. Имя проекта необходимо задать по шаблону «TSN01_LAB1», где «TSN» – инициалы автора программы большими буквами, «01» – номер варианта (2 цифры), далее идет знак «_», слово «LAB» большими буквами и номер лабораторной работы. При необходимости указываем новый каталог для расположения проекта. Нажимаем кнопку «Готово».

При создании приложения необходимо указать, что создавать главный класс не нужно, убрав соответствующую галочку, т.к. главный класс будет располагаться в окне:



Устанавливаем курсор мыши на «пакет по-умолчанию», после этого нажимаем на этом пункте правую кнопку мыши. В появившемся контекстном меню выбираем «Новый» - «Форма JFrame»:

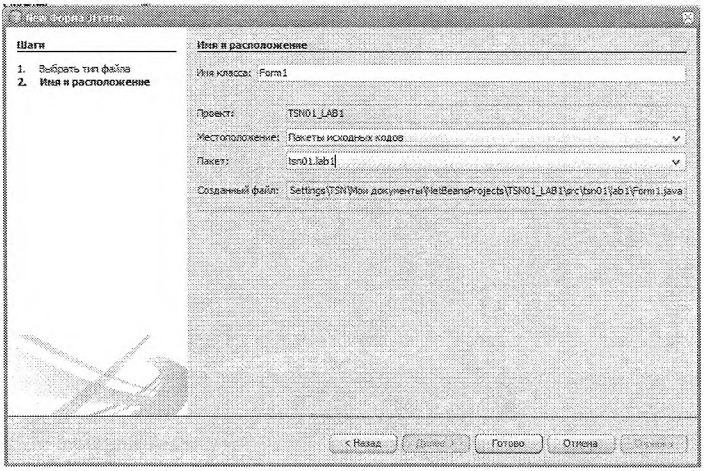


Указываем имя класса (окна). Рекомендуется имя формы задавать по шаблону «Form1», где «Form» – означает, что это форма, первая буква большая, а «1» – номер формы в программе.

Далее, необходимо указать имя пакета. Имя пакета необходимо задать по шаблону «tsn01.lab1», где «tsn» – инициалы автора

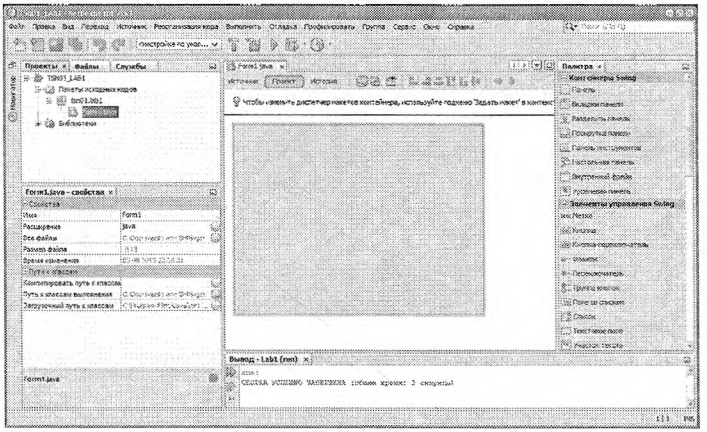
программы маленькими буквами, «01» – номер варианта (2 цифры), далее идет знак «.» (точка), слово «lab» маленькими буквами и номер лабораторной работы.

Нажимаем кнопку «Готово».

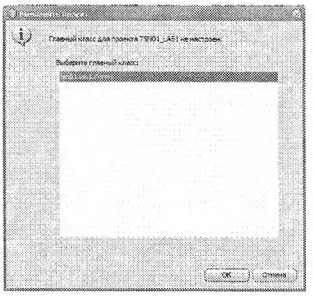


Выделяем в окне «Проекты» полученную форму «Form1». Простейшая программа из пустого окна готова.

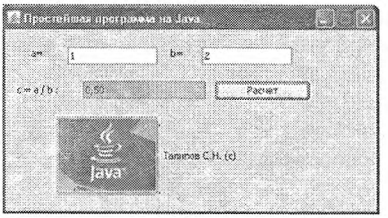
Запускаем полученную программу: «Выполнить» – «Запустить проект» (F6).



Появиться окно для указания главного класса в программе. Необходимо выбрать класс и именем формы и нажать «ОК». После этого запустится программа.

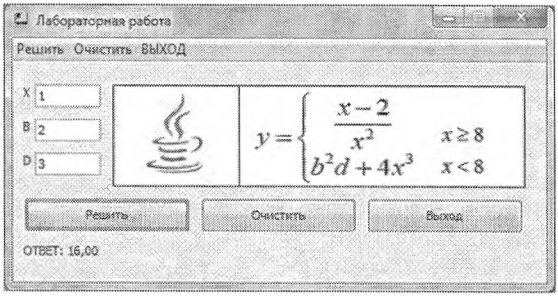


Рассмотрим пример кода для кнопки «Расчет» и интерфейс простейшей программы деления двух цифр:



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Вычисление выражения
    String a, b, c; // Строковые переменные
    double  z; // Вещественные переменные
    double x, y; // Вещественные переменные
// int x, y; // Целочисленные переменные
    a = jTextField1.getText(); // Получение значения из окошка 1
    b = jTextField2.getText(); // Получение значения из окошка 2

    try { // Начало защищенного блока
       x = Double.parseDouble(a); // Преобразование текстового
значения в вещественное
       y = Double.parseDouble(b); // Преобразование текстового
значения в вещественное
// x = Integer.parseInt(a); // Преобразование текстового значения в
целочисленное
// y = Integer.parseInt(b); // Преобразование текстового значения в
целочисленное

       z = x / y; // Вычисление выражения
// z = (double)( x) / (double) (y); // Вычисление выражения

       // Проверка на: 0/0, z/0
       if ((Double.isNaN(z) == true) || Double.isInfinite(z) == true) {
          throw new Exception("error"); // Если нет решение то
генерирование ошибки
    }

// Описание формата вещественного числа
       DecimalFormat df = new DecimalFormat("#0.00");
       c = String.valueOf(df.format(z)); // Преобразование числа в
строку
       jTextField3.setText(c); // Вывод ответа в окошко
// jTextField3.setText(String.valueOf(new
DecimalFormat("#0.00").format(z))); // Вывод ответа в окошко
  } catch (Exception ee) { // Обработчики ошибок защищенного блока
       Toolkit.getDefaultToolkit ().beep (); // Звуковой сигнал
       jTextField3.setText("Неверные данные!"); // Обработка ошибки
ввода или вычисления
  } // Конец защищенного блока
}
```

Рассмотрим пример кода кнопки «Решить» и «Выход» для расчета математического значения по условию:




```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Решение примера
    float x, b, d, y; // Вещественные переменные

    try {
        x = Float.parseFloat(jTextField1.getText()); // Получение данных
        b = Float.parseFloat(jTextField2.getText()); // Получение данных
        d = Float.parseFloat(jTextField3.getText()); // Получение данных
  } catch (Exception ex) {
        Toolkit.getDefaultToolkit().beep(); // Издаем звук
        // Выводим окошко с сообщением об ошибке
        JOptionPane.showMessageDialog(rootPane, "Ошибка введенных
данных!", "Ошибка ввода", JOptionPane.ERROR_MESSAGE);
        jTextField1.requestFocus(); // Устанавливаем фокус на
компонент
        jLabel5.setText("В введенных значениях допущены ошибки.");
        jTextField1.setText(""); // Очистка данных
        jTextField2.setText(""); // Очистка данных
        jTextField3.setText(""); // Очистка данных
        return; // Выход из метода (процедуры)
  }

    if (x >= 8) { // Вычисление выражения
        y = (x - 2) / (x * x);
  } else {
        y = b * b * d + 4 * x * x * x;
  }
```

```
    jLabel5.setText("ОТВЕТ: " + String.format("%.2f", y)); // Выдача
ответа с двумя знаками после запятой
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Выход из программы
    System.exit(0);
}

  private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // Очистка данных
     jLabel5.setText("");
     jTextField1.setText("");
     jTextField2.setText("");
     jTextField3.setText("");
}
```

3 Работа с компонентами графической подсистемы

3.1 Обзор компонент

В стандарт Java входит два пакета для создания оконного пользовательского интерфейса: AWT и его надстройка Swing. Компоненты Swing имеют расширенные возможности по сравнению с аналогичными AWT-компонентами, поэтому рассматриваться будут только они. Во-вторых, в отличие от AWT, компоненты Swing не содержат платформо-зависимого кода.

Контейнеры верхнего уровня:

- JApplet – главное окно апплета;
- JFrame – окно приложения;
- JDialog – диалог приложения;
- JColorChooser – диалог выбора цвета;
- JFileChooser – диалог выбора файлов и директорий;
- FileDialog – диалог выбора файлов и директорий (AWT компонент).

Простые контейнеры:

- JPanel – простая панель для группировки элементов, включая вложенные панели;
- JToolBar – панель инструментов (обычно это кнопки);
- JScroolPane – панель прокрутки, позволяющая прокручивать содержимое дочернего элемента;
- JDesktopPane – контейнер для создания виртуального рабочего стола или приложений на основе MDI (multiple-document interface);
- JEditorPane, JTextPane – контейнеры для отображения сложного документа как HTML или RTF;
- JTabbedPane – контейнер для управления закладками;
- JSplitPane – контейнер, разделяющий два элемента и позволяющий пользователю изменять их размер.

Следующие элементы управления могут использоваться и как контейнеры и как простые компоненты:

- JButton – кнопка;
- JCheckBox – кнопка-флажок;
- JComboBox – выпадающий список;
- JLabel – метка, надпись;
- JList – список;
- JPasswordField – текстовое поле для скрытого ввода;
- JProgressBar – компонент для отображения числового значения в некотором диапазоне;

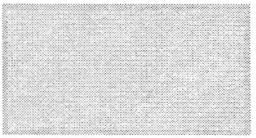
- JRadioButton – переключатели радио-кнопки, обычно используется с компонентом ButtonGroup;
- JSlider – компонент, позволяющий выбрать значение из заданного диапазона;
- JSpinner – компонент, позволяющий выбрать значение из указанной последовательности;
- JTable – визуальная таблица для ввода и отображения данных;
- JTextField – однострочное текстовое поле;
- JFormattedTextField – однострочное текстовое поле, позволяющее вводить значения в определенном формате;
- JTextArea – многострочное текстовое поле;
- JTree – дерево.

3.2 Основные события компонент

№ п/п	Событие	Назначение
1	actionPerformed	Нажатие на компонент
2	valueChanged	Изменение выбранного значения из списка (для JList)
3	stateChanged	Изменение выбранного значения из списка (для JSlider, JSpinner)
4	adjustmentValueChanged	Изменение выбранного значения из списка (для JScrollBar)
5	keyTyped	Ввод символа с клавиатуры
6	keyPressed	Нажатие кнопки на клавиатуре
7	keyReleased	Отжатие кнопки на клавиатуре
8	mousePressed	Нажатие кнопки мыши
9	mouseClicked	Отжатие кнопки мыши
10	mouseReleased	Отжатие кнопки мыши с выделением
11	mouseEntered	Перемещение мыши над компонентом
12	mouseExited	Убирание мыши с компонента
13	mouseWheelMoved	Прокрутка колеса мышки
14	focusGained	Получение фокуса ввода
15	focusLost	Потеря фокуса ввода

3.3 Обычное окно и модальное окно

Окно «JFrame» служит для создания оконного приложения в Java. Это основной контейнер, на котором располагаются другие компоненты, создающие внешний вид и интерфейс программы.



Модальное окно служит JDialog для создания вспомогательных окон, вызываемых по мере необходимости из главного окна JFrame. Особенность окна JDialog состоит в том, что оно открывается в модальном режиме: пока данное окно не будет закрыто, в вызвавшем его основном окне JFrame будет приостановлена работа. Окна JDialog используются для выдачи диалогов «О программе», диалогов подтверждения удаления и т.п.

№ п/п	Свойство	Назначение
1	title	Заголовок формы
2	resizable	Разрешение изменения размеров формы
3	undecorated	Скрытие рамки и заголовка окна
4	defaultCloseOperation	Действие окна при его закрытии: EXIT_ON_CLOSE – выйти из программы (установка по умолчанию для JFrame); DISPOSE – удалиться из памяти (окно больше не нужно) (установка по умолчанию для JDialog); HIDE – спрятаться в памяти (для повторного вызова); DO_NOTHING – ничего не делать (окно не закроется).
5	enabled	Доступность формы
6	alwaysOnTop	Всегда поверх других окон
7	cursor	Вид курсора

№ п/п	События	Назначение
1	windowOpened	Открытие/разворачивание окна
2	windowActivated	Активация окна (переход в окно из другой программы)
3	windowClosing	Закрытие окна
4	windowDeactivated	Деактивация окна (переход из окна в другую программу)
5	windowIconified	Сворачивание окна

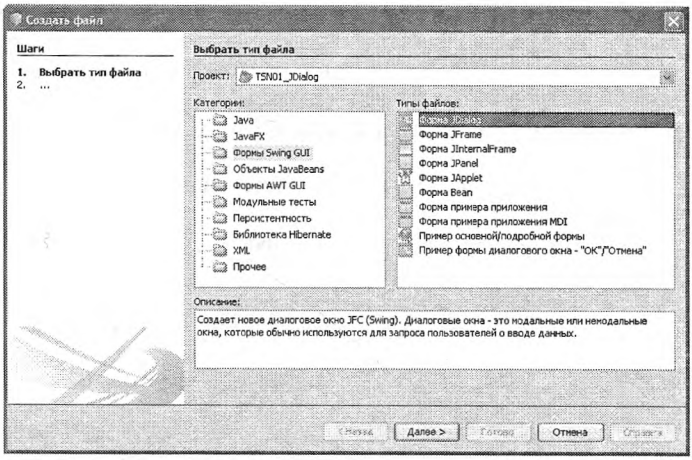
№ п/п	Методы	Назначение
1	setVisible(true)	Включить видимость окна на экране
2	setVisible(false)	Выключить видимость окна на экране
3	dispose	Удалить окно
4	isVisible	Проверка, видно ли окно на экране
5	isActive	Проверка, активно ли окно (программа)
6	setState(JFrame.ICONIFIED)	Свернуть окно
7	setState(JFrame.NORMAL)	Развернуть окно
8	setExtendedState(JFrame.MAXIMIZED_BOTH)	Распахнуть окно на весь экран
9	setExtendedState(JFrame.NORMAL)	Восстановить нормальный размер окна

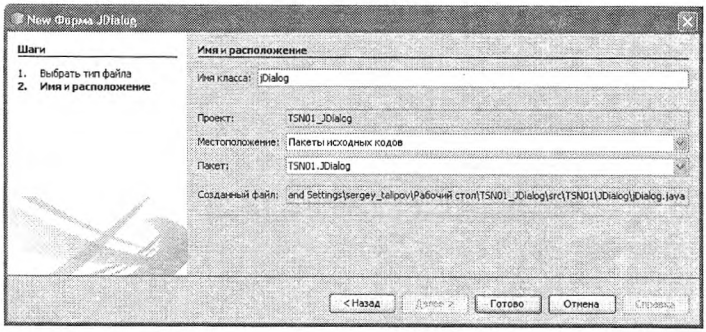
Для закрытия всей программы используется метод «System.exit(0)».

Вспомогательные модальные окна JDialog. Часто в программе необходимо иметь дополнительные вспомогательные окна, которые служат для выдачи какой-либо справочной информации или для запроса ответа на вопрос, например, окно «О программе» или запросы на удаление файлов.

Подобные окна, как правило, открываются в модальном режиме – т.е. поверх всех других окон, и пока данное окно не будет закрыто, вызвавшее его основное окно будет неактивным.

Окна данного типа реализуются через окна типа JDialog. Для добавления в программу окна JDialog необходимо вызвать правой кнопкой мыши контекстное меню на имени пакета, выбрать опцию «Новый» – «Форма JDialog», затем указать имя класса (окна) и пакет, к которому должно относиться окно:



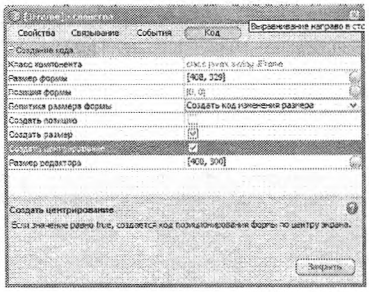


Пример использования модальных окон JDialog:

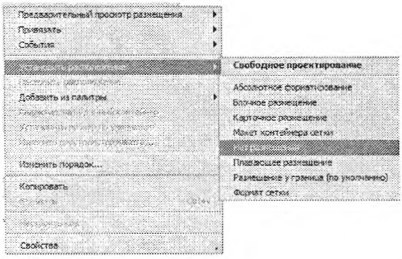
```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt){
    // Вызов окна «О программе»
    new jDialog1(null, true).setVisible(true);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt){
    // Закрытие окна "О программе" (в окне JDialog)
    this.dispose();
}
```

Размещение формы по центру. Для размещения формы по центру необходимо нажать правой кнопкой мыши по окну, в контекстном меню выбрать пункт «Свойства». Далее необходимо выбрать вкладку «Код» и в ней указать:

- «Политика размещения формы» – «Создать код изменения размера»;
- «Создать размер» – нужно установить галочку на данном пункте;
- «Создать центрирование» – нужно установить галочку на данном пункте.



Размещение компонент на форме. Для свободного размещения компонент на форме необходимо нажать правой кнопкой мыши по окну, в контекстном меню выбрать пункт «Установить расположение». Далее необходимо выбрать значение «Нет размещения».

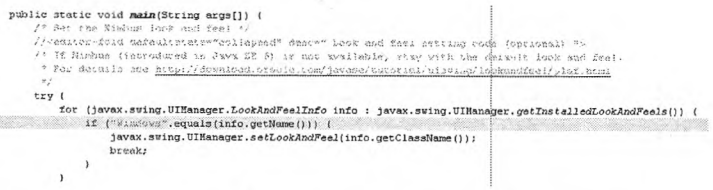


Установка внешнего вида окна в стиле «Windows». Для установки внешнего вида окна в стиле «Windows», необходимо зайти в программный код окна (вкладка «Источник»), раскрыть весь

программный код (нажав на значки «+»), и найти следующий фрагмент кода:

```
if ("Nimbus".equals(info.getName())) {
   javax.swing.UIManager.setLookAndFeel(info.getClassName());
   break;
}
```

Далее, необходимо заменить слово "Nimbus" на слово «Windows», соблюдая регистр символов:



Изменение цвета формы. Для изменения цвета формы необходимо зайти в программный код окна (вкладка «Источник»), раскрыть весь программный код (нажав на значки «+»), и найти следующий фрагмент кода:

```
public NewJFrame() {
    initComponents();
}
```

После строки «initComponents();» необходимо вставить дополнительную новую строку «this.getContentPane().setBackground(java.awt.Color.ORANGE);», где «ORANGE» – желаемый цвет формы:

```
public NewJFrame() {
   initComponents();
   this.getContentPane().setBackground(java.awt.Color.ORANGE);
         // this.getContentPane().setBackground(new java.awt.Color(100,
         100, 200)); // Цвет указан в формате RGB
}
```

Стандартные цвета в Java. Рассмотрим названия основных цветов:

№ п/п	Название цвета	Цвет
1	black	Черный
2	blue	Синий
3	cyan	Голубой
4	darkGray	Темно-серый
5	gray	Серый
6	green	Зеленый
7	lightGray	Светло-серый
8	magenta	Пурпурный
9	orange	Оранжевый
10	pink	Розовый
11	red	Красный
12	white	Белый
13	yellow	Желтый

Изменение иконки формы. Для изменения иконки формы, отображаемой в верхнем левом углу окна, необходимо:

- скинуть иконку формата «jpg» или «png» в папку, где располагаются файлы проекта с расширением «java» (например, «D:\WRK\Java\TSN_lab1\src\tsn_lab1»;
- зайти через Инспектор Объектов в свойство «iconImage»;
- установить использование свойства в значение «Изменяемый код»;
- в открывшемся окне ввести строку:

«java.awt.Toolkit.getDefaultToolkit().createImage(getClass().getResource("icon.png"))»

с именем иконки (в этом примере «icon.png») и нажать на кнопку «OK».

3.4 Кнопка

Кнопки служат для привязки к ним процедур, вызываемых при их нажатии.

№ п/п	Свойство	Назначение
1	background	Цвет фона кнопки
2	enabled	Доступность компонента
3	font	Шрифт текста кнопки
4	foreground	Цвет текста кнопки
5	horizontalAlignment	Выравнивание текста кнопки и картинки по горизонтали
6	horizontalTextPosition	Выравнивание текста кнопки по горизонтали
7	icon	Картинка (jpeg, gif, png) или ссылка из сети
8	text	Текст кнопки
9	opaque	Непрозрачность фона
10	border	Рамка
11	componentPopupMenu	Привязка компонента всплывающего меню
12	contentAreaFilled	Убирание у кнопки рамки и фона
13	cursor	Вид курсора
14	disabledIcon	Картинка кнопки при ее недоступности
15	focusable	Разрешение фокуса ввода по клавише Tab
16	iconTextGap	Отступ между картинкой и текстом
17	label	Текст надписи кнопки
18	pressedIcon	Картинка кнопки при нажатии
19	rolloverIcon	Картинка кнопки при наведении мышки
20	toolTipText	Всплывающая подсказка над компонентом
21	verticalAlignment	Выравнивание текста и картинки по вертикали
22	verticalTextPosition	Выравнивание текста по вертикали
23	Высота	Высота кнопки
24	Ширина	Ширина кнопки

3.5 Компоненты для работы с текстом

Компоненты для работы с текстом служат для ввода/вывода тестовой информации.

Метка «JLabel». Функциональность компонента JLabel сводится к отображению на форме нередактируемой текстовой строки, т.е. надписи.

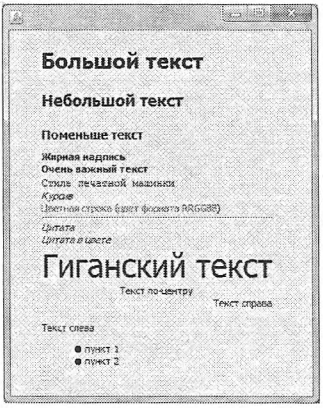
№ п/п	Свойство	Назначение
1	text	Текст метки
2	foreground	Цвет текста
3	background	Цвет фона
4	font	Шрифт текста
5	horizontalTextPosition	Выравнивание текста по горизонтали
6	icon	Картинка (jpeg, gif, png) или ссылка из сети
7	opaque	Непрозрачность фона
8	border	Рамка
9	componentPopupMenu	Привязка компонента всплывающего меню
10	cursor	Вид курсора над компонентом
11	disabledIcon	Картинка компонента при недоступности компонента
12	enabled	Доступность компонента
13	horizontalAlignment	Выравнивание текста и картинки по горизонтали
14	iconTextGap	Отступ между картинкой и текстом
15	toolTipText	Всплывающая подсказка над компонентом
16	verticalAlignment	Выравнивание текста и картинки по вертикали
17	verticalTextPosition	Выравнивание текста по вертикали
18	X	Положение X верхнего левого угла
19	Y	Положение Y верхнего левого угла
20	Высота	Высота компоненты
21	Ширина	Ширина компоненты

В свойстве text можно не только задавать простейший текст, но и задавать его в виде html-текста, например:

```
<html>
<h1>Большой текст</h1>
<h2>Небольшой текст</h2>
<h3>Поменьше текст</h3>
<b>Жирная надпись</b><br>
<strong>Очень важный текст</strong><br>
<kbd>Стиль печатной машинки</kbd><br>
<cite>Курсив</cite><br>
<font color="224956">Цветная строка (цвет формата RRGGBB</font>)<br>
<hr>
<cite>Цитата</cite><br>
<address>Цитата в цвете</address><br>
<p align="center"><font size="7">Гиганский текст</font><br></p>
<p align="center">Текст по-центру</p>
<p align="right">Текст справа</p>
```

```
<br>
<p align="left">Текст слева</p>
<li>пункт 1<br><li>пункт 2<br>
</html>
```

В результате присвоения свойству text данного кода метка станет отображать следующее:



Поле для ввода текста «JTextField». Компонент JTextField представляет собой реализацию однострочной области для ввода/вывода текста.

№ п/п	Свойство	Назначение
1	text	Текст в компоненте
2	editable	Разрешение редактировать текст
3	font	Шрифт текста
4	foreground	Цвет текста
5	background	Цвет фона
6	horizontalAlignment	Выравнивание текста по горизонтали
7	opaque	Непрозрачность фона
8	border	Рамка
9	componentPopupMenu	Привязка компонента всплывающего меню
10	caretColor	Цвет курсора
11	cursor	Вид курсора
12	disabledTextColor	Цвет текста при недоступности компонента

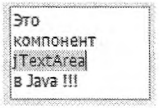
13	enabled	Доступность компонента
14	focusable	Разрешение фокуса ввода/работы с текстом
15	selectedText	Выделенный текст
16	selectedTextColor	Цвет выделенного текста
17	selectionColor	Цвет фона выделенного текста
18	selectionEnd	Конечная позиция выделения текста
19	selectionStart	Начальная позиция выделения текста
20	toolTipText	Всплывающая подсказка над компонентом
21	Высота	Высота компонента
22	Ширина	Ширина компонента

№ п/п	Методы	Назначение
1	getText()	получить текст
2	copy()	копирует выделенный текст в системный буфер обмена
3	cut()	удаляет выделенный текст и копирует его в системный буфер обмена
4	paste()	вставить содержимое системного буфера обмена
5	selectAll()	выделить весь текст
6	getText(int offs, int len)	получить часть текста
7	setText(String t)	установить новый текст
8	replaceSelection(String content)	заменить выделение указанным текстом
9	getSelectedText()	получить выделенный текст
10	getSelectionEnd()	позиция конца выделенного текста
11	getSelectionStart()	позиция начала выделенного текста
12	select(int selStart, int selEnd)	выделить часть текста

Пример работы с JTextField:

```
private void jTextField1KeyPressed(java.awt.event.KeyEvent evt) {
    // Нажатие клавиши "Enter"
    int key = evt.getKeyCode(); // Получаем код нажатой клавиши
    if (key == KeyEvent.VK_ENTER) { // Если нажата, клавиша "Enter", то
      // jButton1.requestFocus(); // Установить фокус на кнопку jButton1
      jButton1ActionPerformed(null); // Вызвать обработчик нажатия на кнопку jButton1
   }
 }
```

Многострочное поле для ввода/вывода текста «JTextArea». Компонент JTextArea служит для ввода/вывода многострочных данных.



№ п/п	Свойство	Назначение
1	text	Текстовое содержимое компонента
2	editable	Разрешение редактирования текста в компоненте
3	enabled	Доступность компонента
4	lineWrap	Автоматически переносить длинный текст на новые строки (true – перенос включен)
5	wrapStyleWord	При включенном режиме «lineWrap» разрешение или запрет на разрыв слов при переносе (true – не разрывать слова)
6	tabSize	Размер отступа при нажатии клавиши «Tab» при вводе текста в компонент
7	font	Шрифт текста списка
8	foreground	Цвет текста списка
9	background	Цвет фона списка
10	selectedTextColor	Цвет выделенного текста
11	selectionColor	Цвет фона выделенного текста
12	disabledTextColor	Цвет текста при включенной недоступности компонента (enabled=false)
13	caretColor	Цвет курсора
14	opaque	Непрозрачность фона
15	border	Дополнительная рамка у компонента
16	cursor	Вид курсора
17	focusable	Разрешение фокуса ввода с клавиатуры
18	toolTipText	Всплывающая подсказка
19	componentPopupMenu	Привязка компонента всплывающего меню

№ п/п	Методы	Назначение
1	setText	Установка текста в компоненте
2	getText	Получение подстроки
3	selectAll	Выделить весь текст
4	setText("")	Удалить весь текст в компоненте
5	copy	Скопировать выделение в буфер обмена
6	cut	Вырезать выделенный текст в буфер обмена
7	paste	Вставить из буфера обмена текст
8	append	Добавить новую строку текста в конец

9	getSelectedText	Получить выделенный текст
10	replaceRange	Замена фрагмента текста на новый текст
11	replaceSelection	Замена выделенного фрагмента текста на новый текст
12	insert	Вставляет новый текст в указанное место имеющегося текста
13	getLineCount	Получить количество строк
14	getCaretPosition	Получить позицию курсора
15	getLineOfOffset	Получить номер строки, где находится курсор
16	getLineStartOffset	Определение начала строки
17	getLineEndOffset	Определение конца строки

При добавлении на форму компонента JTextArea предварительно автоматически вставляется компонент JScrollPane, и уже в него вставляется JTextArea. Это дает компоненту JTextArea возможность горизонтальной и вертикальной прокрутки. Свойства «X», «Y», «Ширина», «Высота» перенесены в компонент JTextArea.

Примеры работы с JTextArea:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Простейшая работа с JTextArea
    jTextArea1.setText("Все работает!!!"); // Выдача текста
    jTextArea1.selectAll(); // Выделить весь текст
    jTextArea1.copy(); // Скопировать выделение в буфер обмена
    //jTextArea1.cut(); //Вырезать выделенный текст в буфер обмена
    //jTextArea1.paste(); //Вставить из буфера обмена текст
    // Получить выделенный текст
    jTextField1.setText(jTextArea1.getSelectedText());
    jTextArea1.replaceSelection("Да!"); // Заменить выделенный текст
    jTextField2.setText(jTextArea1.getText()); // Получить весь текст
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Выбор строки и замена подстроки в JTextArea
    try {
      // Номер выбранной строки
       int line =
jTextArea1.getLineOfOffset(jTextArea1.getCaretPosition());
       int start = jTextArea1.getLineStartOffset(line); // Номер символа
начала выбранной строки
       int end = jTextArea1.getLineEndOffset(line);  // Номер символа
конца выбранной строки
```

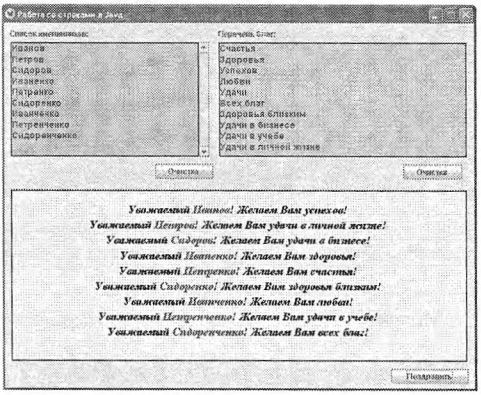
```
        String s = jTextArea1.getText(start, end - start); // Узнаем текст выбранной строки
        jTextField1.setText(s);
        jTextArea1.replaceSelection("Yes!"); // Заменяем выделенный текст на значение «Yes!»
    } catch (Exception e) { jTextField1.setText(""); }
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // Проход по строкам в JTextArea
    try {
        int line_count = jTextArea1.getLineCount() - 1; // Количество строк
        for (int i = 0; i <= line_count; i++) { // Цикл по всем строкам
            int start = jTextArea1.getLineStartOffset(i); // Определим начало i-ой строки
            int end = jTextArea1.getLineEndOffset(i); // Определим конец i-ой строки
            String s = jTextArea1.getText(start, end - start); // Узнаем текст i-ой строки
            jTextArea1.replaceRange(Integer.toString(i + 1) + ") " + s, start, end); // Изменим строку
    }} catch (Exception f) { jTextField1.setText(""); }
 }

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // Добавление строк в JTextArea
    jTextArea1.setText(""); // Удалить весь текст в компоненте
    jTextArea1.append("Строка 1\n"); // Добавить значение и перейти на новую строку
    jTextArea1.append("Строка 2\n"); // Добавить значение и перейти на новую строку
    jTextArea1.append("Строка 3"); // Добавить значение
}
```

Пример работы с JTextArea и динамическими массивами ArrayList:

```
private void jButton_PazdrActionPerformed(java.awt.event.ActionEvent evt) {
    // Нажатие кнопки поздравить
```

```
    ArrayList<String> pozdr = new ArrayList<>(); // Массив пожеланий
    ArrayList<String> fam = new ArrayList<>();  // Массив фамилий
    jEditorPane_Pozdr.setText(null); // Очищаем область поздравлений
    try { // Проверка введенных пользователем пожеланий
       for (int i = 0; i < jTextArea_Gel.getLineCount(); i++) {
          int end = jTextArea_Gel.getLineEndOffset(i);
          int start = jTextArea_Gel.getLineStartOffset(i);
          // Проверяем является ли строки пустой ("==" - не работает)
          if (jTextArea_Gel.getText(start, end - start).trim().equals("")) {
continue; }
          // Добавление поздравления в массив
          pozdr.add(jTextArea_Gel.getText(start, end - start).trim());
    }
  } catch (Exception ex) { JOptionPane.showMessageDialog(rootPane,
"Ошибка чтения пожеланий",
          "Ошибка чтения", JOptionPane.ERROR_MESSAGE);
       return; // Выход
  }
    try { // Проверка введенных пользователем фамилий
       for (int i = 0; i < jTextArea_Fam.getLineCount(); i++) {
          int end = jTextArea_Fam.getLineEndOffset(i);
          int start = jTextArea_Fam.getLineStartOffset(i);
          // Проверяем является ли строки пустой (== - не работает)
          if (jTextArea_Fam.getText(start, end - start).trim().equals("")) {
continue; }
          fam.add(jTextArea_Fam.getText(start, end - start).trim()); //
добавление фамилии в массив
    }
  } catch (Exception ex) { JOptionPane.showMessageDialog(rootPane,
"Ошибка чтения фамилий",
          "Ошибка чтения", JOptionPane.ERROR_MESSAGE);
       return; // Выход
  }
    // Проверка количества поздравлений
    if (fam.size() > pozdr.size()) { // Если фамилий больше чем
поздравлений
       JOptionPane.showMessageDialog(rootPane, "Пожеланий на всех
не хватит! Уменьшите количество фамилий.", "Ошибка ввода",
JOptionPane.ERROR_MESSAGE);
       return;
  }
```

```
    StringBuilder s = new StringBuilder("<p
align=\"center\"><cite><b><font size=\"5\">");
    try { // ПОЗДРАВЛЯЕМ !!!
      for (int i = 0; i < fam.size(); i++) {
        // Генерируем случайное число в диапазоне от 0 до длины
массива поздравлений
        int p = (int) Math.floor(Math.random() * pozdr.size());
        // Генерация поздравления
        s.append("Уважаемый <font color=\"990505\">" +
fam.get(i)+"</font>"
            + "! Желаем Вам " +
            pozdr.get(p).toString().toLowerCase() + "!<br>");
//добавляем символ перехода на новую строку
        pozdr.remove(p); // Удаляем элемент с индексом p из массива
поздравлений
    }
  } catch (Exception ex) { JOptionPane.showMessageDialog(rootPane,
"Ошибка генерации поздравления",
        "Ошибка чтения", JOptionPane.ERROR_MESSAGE);
  }
    s.append("<br>");
    jEditorPane_Pozdr.setText(s.toString());
}
```



Панель редактора текста «JEditorPane». Компонент JEditorPane служит для ввода/вывода многострочных данных с возможностью сложного оформления и разбивки текста. Компонент

поддерживает отображение обычного текста, текста формата html, rtf, а также web-станиц с локальных и удаленных ресурсов.

№ п/п	Свойство	Назначение
1	contentType	Режим работы компонента с текстом: «text/plain» - обычный текст (по-умолчанию); «text/html» – расширенное оформление текста формата «html»; «text/rtf» – расширенное оформление текста формата «rtf». Примечание: Для корректного отображения русских букв rtf-файл должен быть сохранен в LibreOffice или Google document, т.е. файлы, сохраненные MS Office, открываются с некорректным отображением русских букв!
2	text	Текстовое содержимое компонента
3	editable	Разрешение редактирования текста в компоненте
4	font	Шрифт текста списка
5	foreground	Цвет текста списка
6	background	Цвет фона списка
7	enabled	Доступность компонента
8	border	Дополнительная рамка у компонента
9	toolTipText	Всплывающая подсказка
10	selectedTextColor	Цвет выделенного текста
11	selectionColor	Цвет фона выделенного текста
12	disabledTextColor	Цвет текста при включенной недоступности компонента (enabled=false)
13	caretColor	Цвет курсора
14	opaque	Непрозрачность фона
15	cursor	Вид курсора
16	focusable	Разрешение фокуса ввода с клавиатуры
17	componentPopupMenu	Привязка компонента всплывающего меню

№ п/п	Методы	Назначение
1	setText	Установить текст в компоненте
2	selectAll	Выделить весь текст
3	copy	Скопировать выделение в буфер обмена
4	cut	Вырезать выделенный текст в буфер обмена
5	paste	Вставить из буфера обмена текст
6	getSelectedText	Получить выделенный текст

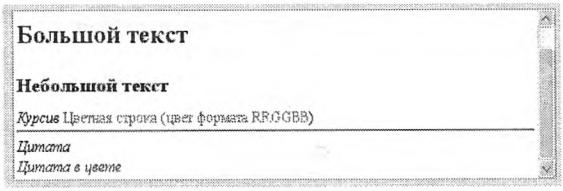
7	replaceSelection	Заменить выделенный текст
8	getText	Получить весь текст
9	setContentType	Установить формат текста: «text/plain» – обычный текст (по-умолчанию); «text/html» – расширенное оформление текста формата «html»; «text/rtf» – расширенное оформление текста формата «rtf».
10	setPage	Установка адреса/имени файла для отображения информации
11	setEditable	Установка/запрет режима редактирования текста

При добавлении на форму компонента JEditorPane предварительно автоматически вставляется компонент JScrollPane, и уже в него вставляется JEditorPane. Это дает компоненту JEditorPane возможность горизонтальной и вертикальной прокрутки. Свойства «X», «Y», «Ширина», «Высота» перенесены в компонент JEditorPane.

При режиме сложного оформления текста (contentType = text/html) в свойстве text можно задавать не просто обычный текст, но и текст в виде html, например:

```
<html>
 <head>
 </head>
 <body>
  <p style="margin-top: 0">
<h1>Большой текст</h1>
<h2>Небольшой текст</h2>
<cite>Курсив</cite><br>
<font color="224956">Цветная строка (цвет формата RRGGBB</font>)<br>
<hr>
<cite>Цитата</cite><br>
<address>Цитата в цвете</address><br>
  </p>
 </body>
</html>
```

В результате присвоения свойству text данного кода JEditorPane станет отображать следующее:



Пример выдачи текста с обычным оформлением в JEditorPane:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Работа с JEditorPane
    jEditorPane1.setText("Все работает!!!"); // Выдача текста
    jEditorPane1.selectAll(); // Выделить весь текст
    jEditorPane1.copy(); // Скопировать выделение в буфер обмена
    //jEditorPane1.cut(); Вырезать выделенный текст в буфер обмена
    //jEditorPane1.paste(); Вставить из буфера обмена текст
    // Получить выделенный текст
    jTextField1.setText(jEditorPane1.getSelectedText());
    jEditorPane1.replaceSelection("Да!"); // Заменить выделенный
текст
    jTextField2.setText(jEditorPane1.getText()); // Получить весь текст
}
```

Пример выдачи текста с html-форматированием в JEditorPane:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Работа с JEditorPane
    jEditorPane1.setContentType("text/html;Content-Type=windows-
1251");
    jEditorPane1.setText("<html><body>Все
работает!!!</body></html>");
}
```

Пример выдачи текста с расширенным оформлением текста в JEditorPane:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Работа с JEditorPane
    HTMLDocument hd = new HTMLDocument(); // HTMLDocument
    jEditorPane1.setContentType("text/html"); // Режим сложного
оформления текста
```

```
    jEditorPane1.setDocument(hd); // Связывание jEditorPane1 и
HTMLDocument
    jEditorPane1.setText("Hello World"); // Выдача текста в компонент
    MutableAttributeSet attr = new SimpleAttributeSet(); // Атрибуты
текста
    StyleConstants.setFontFamily(attr, "Arial"); // Установка шрифта
    StyleConstants.setBackground(attr, Color.red); // Установка цвета
    StyleConstants.setItalic(attr, true); // Установка наклонного стиля
текста
    hd.setCharacterAttributes(6, 4, attr, false); // Установить атрибуты 4
символам начиная с 6-го
}
```

Пример подсветки искомого значения в JEditorPane (только для расширенного оформления текста):

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Подсветка искомого значения в JEditorPane
    // Получение содержимого документа
    DefaultStyledDocument doc = (DefaultStyledDocument)
jEditorPane1.getDocument();
    MutableAttributeSet attr = new SimpleAttributeSet(); // Атрибуты
текста
    StyleConstants.setBackground(attr, new Color(255, 255, 177)); //
Установка цвета
    String s2 = jTextField1.getText().toUpperCase(); // Искомый текст
большими буквами
    String s = null; // Текст документа
    int d = s2.length(); // Длина искомого текста
    try {
      // Получение текста документа большими буквами
      s = doc.getText(0, doc.getLength()).toUpperCase();
  } catch (BadLocationException ex) {
  }
    int k = 1, k2 = 0; // Инициализация переменных
    while (k > 0) { // Цикл пока находится искомое значение
      k = s.indexOf(s2, k2); // Поиск искомого значения в тексте
документа
      if (k > 0) { // Если поиск успешен, то
        // Установка атрибутов тексту документа
        doc.setCharacterAttributes(k, d, attr, false);  k2 = k + d;
```

```
    }
  }
}
```

Пример выдачи текста с html-станицы в JEditorPane:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Загрузка внешней WEB html-страницы в JEditorPane
    jEditorPane1.setContentType("text/html"); // Режим сложного
оформления текста
    jEditorPane1.setEditable(false); // Запрет на редактирование
документа
    try { jEditorPane1.setPage("http://www.google.ru"); // Загрузка
данных с html-страницы
  } catch (Exception e) { }
 }
```

Пример загрузки внешней локальной html-страницы в JEditorPane:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Загрузка внешней локальной html-страницы в JEditorPane
    try {
      jEditorPane1.setContentType("text/html");// Режим сложного
оформления текста
      jEditorPane1.setEditable(false); // Запрет на редактирование
документа
      // Загрузка данных из html-страницы
      jEditorPane1.setPage(new File("d:/demo.html").toURI().toURL());
  } catch (MalformedURLException ex) {
  } catch (IOException ex) {
  }
}
```

Пример загрузки локальной (из каталога готового jar-файла) html-страницы в JEditorPane:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Загрузка локальной (из каталога готового jar-файла) html-
страницы в JEditorPane
    try {
```

```
        jEditorPane1.setContentType("text/html");// Режим сложного
оформления текста
        jEditorPane1.setEditable(false); // Запрет на редактирование
документа
        jEditorPane1.setPage(getClass().getClassLoader().
              getResource("mypackage/demo.html").toURI().toURL());//
Загрузка данных из html-страницы
    } catch (MalformedURLException | URISyntaxException ex) {
    } catch (IOException ex) {
    }
}

private void
jEditorPane1HyperlinkUpdate(javax.swing.event.HyperlinkEvent evt) {
      // Установка возможности перехода по гипер-ссылкам в
JEditorPane
      HyperlinkEvent.EventType type = evt.getEventType();
      final URL url = evt.getURL();
      if (type != HyperlinkEvent.EventType.ENTERED)
         if (type == HyperlinkEvent.EventType.ACTIVATED) {
         Document doc = jEditorPane1.getDocument();
         try { jEditorPane1.setPage(url); }
           catch (IOException ioException) {
jEditorPane1.setDocument(doc); }
    }
}
```

Пример выдачи текста с rtf-документа в JEditorPane:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // Загрузка с диска rtf-файла в JEditorPane
      try {
         jEditorPane1.setContentType("text/rtf");// Режим сложного
оформления текста
         jEditorPane1.setEditable(false); // Запрет на редактирование
документа
         jEditorPane1.setPage(new File("d:/demo.rtf").toURI().toURL());
    } catch (MalformedURLException ex) {
    } catch (IOException ex) {
    }
}
```

Пример загрузки локального (из каталога готового jar-файла) rtf-файла в JEditorPane:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Загрузка локального (из каталога готового jar-файла) rtf-файла в
JEditorPane
    jEditorPane1.setContentType("text/rtf");// Режим сложного
оформления текста
    jEditorPane1.setEditable(false); // Запрет на редактирование
документа
    URL urlToRtf =
getClass().getClassLoader().getResource("mypackage/demo.rtf");
    try { jEditorPane1.setPage(urlToRtf); } catch (IOException ex) {
  }
}
```

Пример загрузки rtf-файла с текущего каталога в JEditorPane:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Загрузка rtf-файла с текущего каталога в JEditorPane
    try { // Получаем полное имя RTF-файла в текущем каталоге
      String FileName = new
File(".").getAbsoluteFile().getParentFile().getAbsolutePath()
          + System.getProperty("file.separator") + "demo.rtf";
      jEditorPane1.setContentType("text/rtf"); // Устанавливаем режим
rtf-документов
      jEditorPane1.setPage(new File(FileName).toURI().toURL()); //
Загружаем RTF-документ
  } catch (MalformedURLException ex) {
  } catch (IOException ex) {
  }
}
```

Пример записи данных из JEditorPane в rtf-файл текущего каталога:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Запись данных из JEditorPane в rtf-файл текущего каталога
    try { // Получаем полное имя RTF-файла в текущем каталоге
      String FileName = new
File(".").getAbsoluteFile().getParentFile().getAbsolutePath()
```

```
            + System.getProperty("file.separator") + "demo.rtf";
        // Считываем содержимое документа
        Document doc = jEditorPane1.getDocument();
        FileOutputStream out = new FileOutputStream(FileName);
        RTFEditorKit rtf = new RTFEditorKit(); // Инструментарий
работы с RTF
        rtf.write(out, doc, 0, doc.getLength()); // Запись документа в rtf-
файл на диск
    } catch (IOException | BadLocationException ex) {
    }
}
```

Пример очистки данных в JEditorPane:

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // Очистка данных в JEditorPane
    jEditorPane1.setText("");
}
```

Пример установки типа и размера шрифта по-умолчанию в jEditorPane для html и rtf-режимов:

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    // Установка типа и размера шрифта по-умолчанию в jEditorPane
    new StyledEditorKit.FontFamilyAction("Tahoma",
"Tahoma").actionPerformed(null);
    new StyledEditorKit.FontSizeAction("16", 16).actionPerformed(null);
}
```

Пример загрузки txt-файла в JEditorPane с помощью диалога выбора файла:

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // Загрузка txt-файла в JEditorPane с помощью диалога выбора
файла
    try {
        JFileChooser fileopen = new JFileChooser(); // Создаем объект
выбора файла
        int ret = fileopen.showDialog(null, "Открыть файл"); // Вызываем
диалог
```

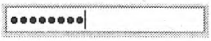
```
        if (ret != JFileChooser.APPROVE_OPTION) { return; } // Если
файл не выбран, то выход
        File FileName = fileopen.getSelectedFile(); // Получение имени
файла
        // Открываем потоки чтения из файла
        FileInputStream stream = new FileInputStream(FileName);
        InputStreamReader reader = new InputStreamReader(stream);
        BufferedReader buffered_reader = new BufferedReader(reader);
        String txt = ""; String buf; // Инициализация переменных
        // Загрузка текста из файла
        while ((buf = buffered_reader.readLine()) != null) {
          txt += buf + "\r\n";
      }
        jEditorPane1.setText(txt); // Установить считанный из файла
текст в jEditorPane1
        buffered_reader.close(); reader.close(); stream.close(); // Закрытие
потоков чтения
    } catch (FileNotFoundException ex) {
    } catch (IOException ex) {
    }
}
```

Пример сохранения текста из JEditorPane в txt-файл с помощью диалога:

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
      // Сохранение текста из JEditorPane в txt-файл с помощью диалога
выбора файла
      try {
        JFileChooser fileopen = new JFileChooser(); // Создаем объект
выбора файла
        int ret = fileopen.showDialog(null, "Сохранить файл"); //
Вызываем диалог
        if (ret != JFileChooser.APPROVE_OPTION) { return; } // Если
файл не выбран, то выход
        File FileName = fileopen.getSelectedFile(); // Получение имени
файла
        // Считываем содержимое документа
        PlainDocument doc = (PlainDocument)
jEditorPane1.getDocument();
```

```
        OutputStreamWriter out = new OutputStreamWriter(new
FileOutputStream(FileName));
        String txt = doc.getText(0, doc.getLength()); // Считываем текст
из jEditorPane1
        out.write(txt); // Сохраняем текст в файл
        out.close(); // Закрываем файл
    } catch (IOException | BadLocationException ex) {
    }
}
```

Компонент «JPasswordField» (поле пароля). JPasswordField представляет собой компонент для ввода пароля.



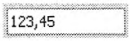
№ п/п	Свойство	Назначение
1	editable	Разрешение редактировать текст
2	font	Шрифт текста
3	foreground	Цвет текста
4	background	Цвет фона
5	horizontalAlignment	Выравнивание текста по горизонтали
6	opaque	Непрозрачность фона
7	border	Рамка
8	componentPopupMenu	Привязка компонента всплывающего меню
9	caretColor	Цвет курсора
10	cursor	Вид курсора
11	disabledTextColor	Цвет текста при недоступности компонента
12	enabled	Доступность компонента
13	focusable	Разрешение фокуса ввода/работы с текстом
14	selectedText	Выделенный текст
15	selectedTextColor	Цвет выделенного текста
16	selectionColor	Цвет фона выделенного текста
17	toolTipText	Всплывающая подсказка над компонентом
18	Высота	Высота компонента
19	Ширина	Ширина компонента

№ п/п	Методы	Назначение
1	getPassword()	получить пароль

Пример работы с JPasswordField:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Получить введенный пароль из компонента JPasswordField
    jTextField1.setText(String.valueOf(jPasswordField1.getPassword()));
}
```

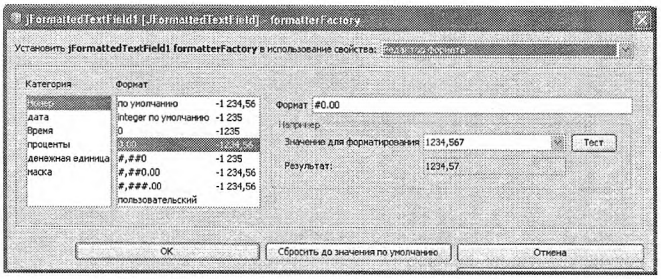
Компонент «JFormattedTextField» (форматируемое поле). Класс JFormattedTextField представляет собой компонент для ввода текстового значения по заданному шаблону.



№ п/п	Свойство	Назначение
1	text	Текстовое значение в компоненте
2	formatterFactory	Шаблон для ввода данных
3	editable	Разрешение редактировать текст
4	font	Шрифт текста
5	foreground	Цвет текста
6	background	Цвет фона
7	horizontalAlignment	Выравнивание текста по горизонтали
8	opaque	Непрозрачность фона
9	border	Рамка
10	componentPopupMenu	Привязка компонента всплывающего меню
11	caretColor	Цвет курсора
12	cursor	Вид курсора
13	disabledTextColor	Цвет текста при недоступности компонента
14	enabled	Доступность компонента
15	focusable	Разрешение фокуса ввода/работы с текстом
16	selectedText	Выделенный текст
17	selectedTextColor	Цвет выделенного текста
18	selectionColor	Цвет фона выделенного текста
19	toolTipText	Всплывающая подсказка над компонентом
20	Высота	Высота компонента
21	Ширина	Ширина компонента

№ п/п	Методы	Назначение
1	getText()	Получить введенное значение с учетом шаблона
2	getValue()	Получить введенное значение без учета шаблона

Шаблоны ввода (значение свойства formatterFactory) в JFormattedTextField:

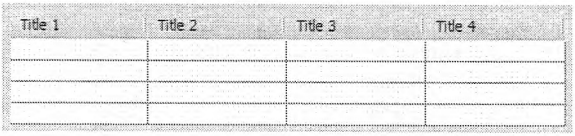


Пример работы с JFormattedTextField:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Пример чтения данных с компонента JFormattedTextField
    jTextField1.setText(jFormattedTextField1.getText()); // Получить
введенное значение по шаблону
    jTextField2.setText(jFormattedTextField1.getValue().toString());//
Получить введенное значение
}
```

3.6 Работа с визуальными таблицами

Компонент JTable предназначен для отображения данных в виде визуальной таблицы.



№ п/п	Свойство	Назначение
1	model	МОДЕЛЬ ТАБЛИЦЫ
2	selectionModel	Выбор режима выделения данных (выбрать «Одиночное выделение»)
3	autoResizeMode	Режим масштабирования ширины столбцов таблицы: установить в «OFF» - использовать заданные размеры столбцов, не использовать авто масштабирование столбцов; SUBSEQUENT_COLUMNS – вписать все столбцы в размер таблицы, используя авто масштабирование.
4	showHorizontalLines	Показать горизонтальные линии сетки (да)
5	showVerticalLines	Показать вертикальные линии сетки (да)

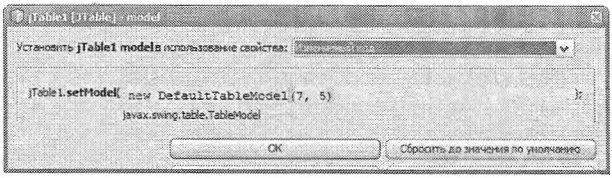
6	autoCreateRowSorter	Разрешить сортировку строк по щелчку на заголовке столбца (нет)
7	tableHeader	Разрешить менять мышью столбцы местами и их ширину (нет)
8	cellSelectionEnabled	Разрешить выделять ячейку (да)
9	columnSelectionAllowed	Разрешит выбирать столбец (да)
10	rowSelectionAllowed	Разрешить выбирать строку (да)
11	border	Рамка над областью с данными (установить в «LineBorder»)
12	fillsViewportHeight	Заливать фоном свободные от строк области компонента
13	columnCount	Количество колонок (только чтение)
14	rowCount	Количество строк (только для чтения)
15	rowHeight	Высота строк
16	font	Шрифт текста
17	foreground	Цвет текста
18	background	Цвет фона
19	gridColor	Цвет сетки
20	selectionBackground	Цвет фона выделенной ячейки
21	selectionForeground	Цвет текста выделенной ячейки
22	enabled	Доступность компонента
23	opaque	Непрозрачность фона
24	focusable	Разрешение фокуса ввода с клавиатуры
25	cursor	Вид курсора над компонентом
26	componentPopupMenu	Привязка компонента всплывающего меню
27	toolTipText	Всплывающая подсказка над компонентом

При добавлении таблицы NetBeans автоматически добавляет компонент «JScrollPane», в который встраивает «JTable». Компонент «JScrollPane» необходим для добавления возможности горизонтальной и вертикальной прокрутки у таблицы и не требует какой-либо отдельной настройки.

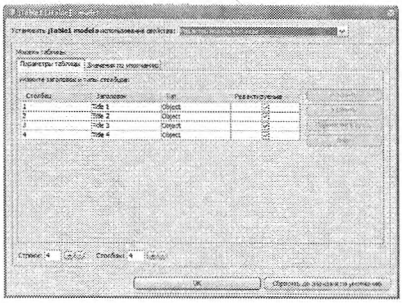
При использовании у формы (JFrame) стиля по умолчанию «Nimbus» линии сетки отображаться не будут. Необходимо выбрать любой другой стиль, например «Windows».

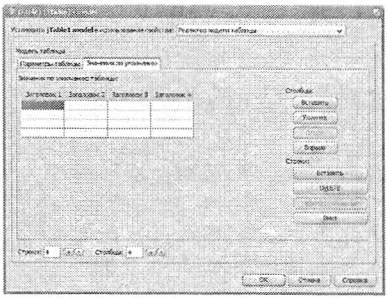
Стандартная модель таблицы JTable. Для того чтобы задать стандартную двумерную модель таблицы, необходимо войти в свойство «model» таблицы, выбрать режим «Изменяемый код» и вписать в открывшееся поле значение «new DefaultTableModel(7, 5)», где первая цифра - желаемое количество строк, вторая цифра – желаемое количество столбцов. Нажать кнопку «OK».

Данная стандартная модель позволяет динамически (программно) добавлять и удалять строки и столбцы таблицы.

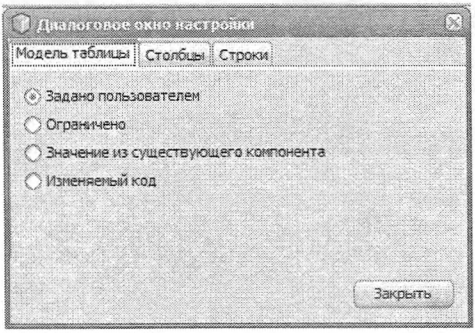


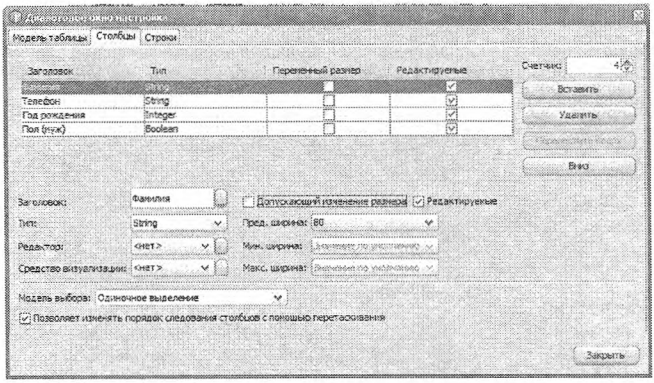
Пользовательская модель таблицы JTable. Для того чтобы задать пользовательскую модель таблицы – т.е. то, что она должна хранить и отображать, необходимо войти в свойство «model», выбрать режим «Редактор модели таблицы» и указать желаемое количество строк, столбцов, тип столбцов (полей):

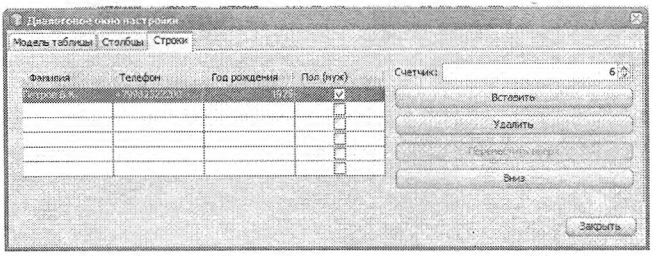




Настройка пользовательских полей таблицы JTable. Для того, чтобы детально настроить содержимое таблицы (структуру и тип ее колонок, количество строк, столбцов, ширину каждого столбца), необходимо щелкнуть на таблице правой кнопкой мыши, в появившемся контекстном меню выбрать опцию «Содержание таблицы» и произвести в открывшемся окне необходимые настройки:



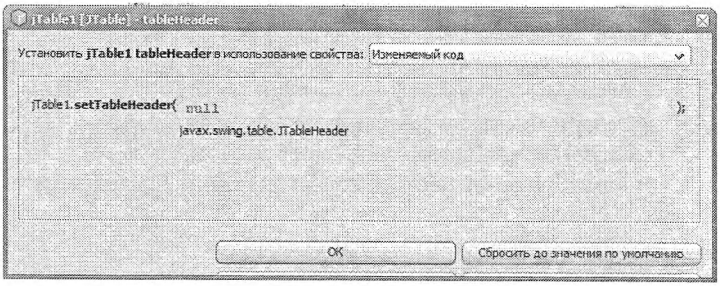




Если использовать в типах столбцов какое-либо значение, кроме «Object», то возможность динамически (программно) менять количество столбцов в таблице будет недоступна. Поэтому, указывать конкретный тип столбцов вместо общего типа «Object» целесообразно

для случаев, когда таблица должна сама контролировать вводимые данные (например, для полей типа «Integer») и нет необходимости программно менять количество столбцов в таблице. Если необходимо программно менять количество столбцов в таблице, то все поля в модели должны быть типа «Object» и автоматической проверки данных таблица осуществлять не будет.

Выключение заголовков в таблице JTable. Для того чтобы убрать у таблицы заголовок с названиями столбцов, необходимо зайти в свойство «tableHeader», установить свойство из режима «Редактор заголовка таблицы» в режим «Изменяемый код» и вписать в открывшееся поле значение «null»:



Пример заполнения случайными числами JTable:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt){
    // Случайные числа в JTable
    long k;
    for (int i = 0; i < jTable1.getRowCount(); i++) {
       for (int j = 0; j < jTable1.getColumnCount(); j++) {
          k = Math.round(Math.random() * 10);
          jTable1.setValueAt(String.valueOf(k), i, j);
    }
  }
    // jButton2ActionPerformed(null);
}
```

Пример суммы цифр в ячейках JTable:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Сумма цифр в JTable
    long s = 0;
```

```
    try {
      for (int i = 0; i < jTable1.getRowCount(); i++) {
        for (int j = 0; j < jTable1.getColumnCount(); j++) {
          s += Integer.parseInt(jTable1.getValueAt(i, j).toString());
        }
      }
      jLabel1.setText(String.valueOf(s));
    } catch (Exception ee) {
      jLabel1.setText("Error");
    }
}
```

Пример добавления и вставки строк в JTable:

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // Добавить и вставка строк в JTable
    DefaultTableModel dtm = new DefaultTableModel();
    dtm = (DefaultTableModel) jTable1.getModel();
    dtm.setRowCount(dtm.getRowCount() + 1); // Добавить в конец
новую строку
    dtm.insertRow(1, new Object [] {"Первый столбец","123"}); //
Вставить во 2 строку новую строку
}
```

Пример удаление и перемещение строк в JTable:

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // Удаление и перемещение строк в JTable
    DefaultTableModel dtm = new DefaultTableModel();
    dtm = (DefaultTableModel) jTable1.getModel();
    try {
      dtm.setRowCount(dtm.getRowCount() - 1); // Удалить
последнюю строку
      dtm.removeRow(1); // Удалить 2 строку
      dtm.moveRow(0, 2, 3); // Переместить строки 1,2,3 за строкой 4
    } catch (Exception e) {;
    }
}
```

Пример добавления в конец нового столбца в JTable (только для стандартной модели «new DefaultTableModel (кол_строк,

кол_столбцов)» или пользовательской модели со всеми столбцами типа «Object»):

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    // Добавить столбец в JTable
    DefaultTableModel dtm = (DefaultTableModel) jTable1.getModel();
    dtm.addColumn("New");
}
```

Пример удаления последнего столбеца в JTable (работает только для стандартной модели или пользовательской модели со всеми столбцами типа «Object»):

```
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    // Удалить столбец в JTable
    DefaultTableModel dtm = new DefaultTableModel();
    dtm = (DefaultTableModel) jTable1.getModel();
    try {
       dtm.setColumnCount(dtm.getColumnCount() - 1);
  } catch (Exception e) {
  }
}
```

Пример выбора ячейки мышью в JTable:

```
private void jTable1_mouseClicked(java.awt.event.MouseEvent evt) {
    // Выбор ячейки мышью в JTable
    int row = jTable1.getSelectedRow(); // определяем какая строка
выбрана
    int column = jTable1.getSelectedColumn(); // определяем какой
столбец
    jLabel2.setText("Значение: " + jTable1.getValueAt(row, column));
}
```

Пример выбора ячейки клавиатурой в JTable:

```
private void jTable1KeyReleased(java.awt.event.KeyEvent evt) {
    // Выбор ячейки клавиатурой в JTable
// jTable1_mouseClicked(null);
    int row = jTable1.getSelectedRow(); // определяем какая строка
выбрана
```

```
    int column = jTable1.getSelectedColumn(); // определяем какой
столбец
    jLabel2.setText("Значение: " + jTable1.getValueAt(row, column));
}
```

Пример инициализации таблицы в JTable:

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    // Инициализация таблицы в JTable
    for (int i = 0; i < jTable1.getColumnCount(); i++) {
      // Установка ширины столбцов
      jTable1.getColumnModel().getColumn(i).setPreferredWidth(60);
      // jTable1.getColumnModel().getColumn(i).setMaxWidth(60);
      // jTable1.getColumnModel().getColumn(i).setMinWidth(60);
  }
    for (int i = 0; i < jTable1.getRowCount(); i++) {
      // Установка высоты строк
      jTable1.setRowHeight(i, 18);
  }
    for (int i = 0; i < jTable1.getRowCount(); i++) {
      for (int j = 0; j < jTable1.getColumnCount(); j++) {
        jTable1.setValueAt("0", i, j);
    }
  }
}
```

Пример обработки данных в таблице JTable:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Обработка данных в таблице JTable
    double fk = 1, sum = 0, pr = 1, y = 0;
    for (int i = 0; i < jTable1.getRowCount(); i++) {
      if (jTable1.getValueAt(i, 0) == null) {
        JOptionPane.showMessageDialog(rootPane, "Проверьте
правильностью заполнения столбца K(i)", "Ошибка ввода",
JOptionPane.ERROR_MESSAGE);
        return;
    }
  }
    for (int i = 0; i < jTable1.getRowCount(); i++) {
```

```
        sum += Integer.parseInt(jTable1.getValueAt(i, 0).toString()); //
Расчет суммы
        try {
          pr *= Integer.parseInt(jTable1.getValueAt(i, 0).toString()); //
Расчет произведения
          fk *= i + 1; // Расчет факториала
          y = (Math.sqrt((pr - sum) / fk) / (i + 1)) * (Math.pow(Math.tan((i
+ 1) / 2), 2)); // Расчет по заданной формуле
          jTable1.setValueAt(
              BigDecimal.valueOf(y).setScale(2,
BigDecimal.ROUND_HALF_DOWN).doubleValue(), i, 1); // Вывод
результата
      } catch (Exception e) {
          jTable1.setValueAt(0, i, 1);
      }
    }
}
```

3.7 Контейнеры-панели

Компонент JPanel (панель). Компонент JPanel служит для декоративных целей, а также как контейнер для размещения на нем других компонент.

Для произвольного размещения на панели компонент необходимо щелкнуть по панели правой кнопкой мыши, из контекстного меню выбрать опцию «Установить расположение» – «Нет размещения».

№ п/п	Свойство	Назначение
1	border	Вид рамки у панели
2	foreground	Цвет панели
3	toolTipText	Всплывающая подсказка

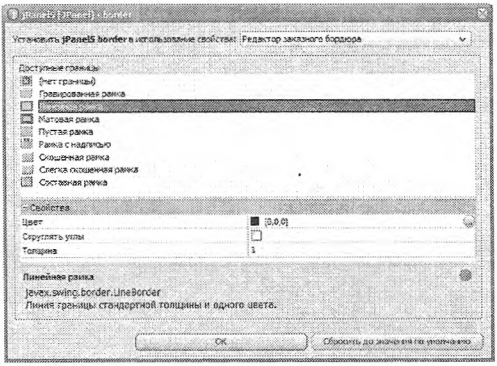
№	Методы	Назначение

п/п		
1	setVisible	Включить/выключить видимость панели. При выключении видимости панель, и все компоненты на ней, становятся невидимыми, после включения видимости панель становится видимой и все компоненты на ней

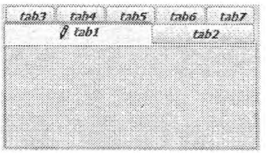
Пример работы с панелью JPanel:

```
private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Включение/выключение видимости панели JPanel
    if (jCheckBox1.isSelected()) { // Если флажок выбран, то включить
видимость панели
        jPanel1.setVisible(true); // Включить видимость панели
  } else {
        jPanel1.setVisible(false); // Выключить видимость панели
  }
}
```

Виды рамок у панелей JPanel (свойство border):



Компонент JTabbedPane (вкладки панели). Компонент JTabbedPane служит для создания панелей вкладок.



Для создания панелей вкладок необходимо:

- нанести компонент JTabbedPane;
- нанести на компонент JTabbedPane нужное количество панелей JPanel. Каждая нанесенная панель будет являться вкладкой в компоненте JTabbedPane.

№ п/п	Свойство	Назначение
1	selectedIndex	Номер активной вкладки (отсчет с 0)
2	tabPlacement	Место расположения ярлыков вкладок (вверху, внизу, слева, справа)
3	tabLayoutPolicy	Режим отображения ярлыков вкладок при множестве вкладок: SCROLL_TAB_LAYOUT – вкладки в один ряд с прокруткой; WRAP_TAB_LAYOUT – несколько рядов вкладок без прокрутки.
4	font	Шрифт надписей вкладки
5	foreground	Цвет надписей (ярлыков) панели
6	border	Вид рамки у панели
7	toolTipText	Всплывающая подсказка
8	X	Положение верхнего левого угла компонента, координата X
9	Y	Положение верхнего левого угла компонента, координата Y
10	Ширина	Ширина компонента
11	Высота	Высота компонента

№ п/п	События	Назначение
1	stateChanged	Переключение активной вкладки

№ п/п	Методы	Назначение
1	getSelectedIndex	Получение номера текущей (активной) вкладки (отсчет с 0)
2	setSelectedIndex	Установка желаемой активной (текущей) вкладки (отсчет с 0)
3	getTabCount	Получение количества вкладок в компоненте (отсчет с 1)

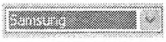
При добавлении панелей-вкладок JPanel у панелей появляются дополнительные новые свойства: «Заголовок вкладки» – для задания названия вкладки, «Значок вкладки» – для выбора картинки вкладки (располагается слева от заголовка вкладки), «Подсказка вкладки» – всплывающая подсказка при наведении курсора мыши на вкладку.

Примеры работы с вкладками JTabbedPane:

```
private void jTabbedPane1StateChanged(javax.swing.event.ChangeEvent
evt) {
    // Выдача номер текущей вкладки в JTabbedPane при
переключении вкладок
    JOptionPane.showMessageDialog(rootPane,
String.valueOf(jTabbedPane1.getSelectedIndex()));
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Выдача количества вкладок в JTabbedPane
    JOptionPane.showMessageDialog(rootPane,
String.valueOf(jTabbedPane1.getTabCount()));
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Переключение активной вкладки в JTabbedPane
    jTabbedPane1.setSelectedIndex(2);
}
```

3.8 Компоненты выбора и отображения значения

Компонент «JComboBox» (поле со списком). Компонент JComboBox служит для выбора какого-либо одного текстового значения из выпадающего списка.



№ п/п	Свойство	Назначение
1	model	СПИСОК ЗНАЧЕНИЙ СПИСКА (набор текстовых строк, без пустых строк в конце)
2	font	Шрифт текста списка
3	foreground	Цвет текста списка
4	background	Цвет фона списка
5	selectedIndex	Номер выбранной строки (отсчет с 0)

6	selectedItem	Значение выбранного элемента
7	editable	Разрешение редактирования значений
8	maximumRowCount	Количество отображаемых строк в выпадающем списке, после которых включается вертикальная прокрутка списка
9	toolTipText	Всплывающая подсказка
10	border	Дополнительная рамка у компонента
11	componentPopupMenu	Привязка компонента всплывающего меню
12	enabled	Доступность компонента
13	focusable	Разрешение фокуса ввода с клавиатуры
14	opaque	Непрозрачность фона
15	X	Положение верхнего левого угла компонента, координата X
16	Y	Положение верхнего левого угла компонента, координата Y
17	Ширина	Ширина компонента
18	Высота	Высота компонента

№ п/п	События	Назначение
1	actionPerformed	Основное событие на изменение выбранного значения в списке

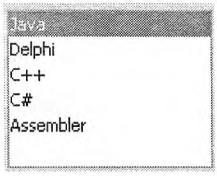
№ п/п	Методы	Назначение
1	getSelectedItem()	Получение выбранного значения из списка
2	getSelectedIndex()	Получение номера выбранного значения из списка (отсчет с 0)
3	removeAllItems	Удалить все строки списка
4	addItem	Добавить новую строку в список
5	removeItemAt	Удалить нужную строку из списка
6	setSelectedIndex()	Установка желаемого номера строки для выбора (отсчет с 0)
7	setSelectedItem()	Установка желаемого значения строки для выбора
8	getItemCount()	Количество строк в списке (отсчет с 1)

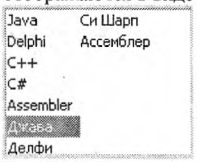
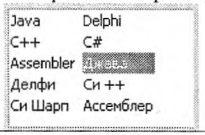
Примеры работы с JComboBox:

```
private void jComboBox1ActionPerformed(java.awt.event.ActionEvent
evt) {
    // Обработка выбора значения из списка JComboBox
    try {
   // Получение значения выбранной строки
        jTextField1.setText(jComboBox1.getSelectedItem().toString());
  } catch (Exception e) { // Значение в компоненте не выбрано
```

```
        jTextField1.setText("");
    };
}

private void jComboBox1ActionPerformed(java.awt.event.ActionEvent
evt) {
    // Обработка выбора значения из списка JComboBox по
выбранному номеру строки
    int k = jComboBox1.getSelectedIndex(); // Получение номера
выбранной строки
        if (k==-1) return; // Если ничего не выбрано, то выход из
процедуры

jTextField1.setText(jComboBox1.getModel().getElementAt(k).toString());
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Пример изменения набора строк в компоненте JComboBox
    jComboBox1.removeAllItems(); // Очистка всех элементов
    jComboBox1.addItem("Delphi"); // Добавление элемента
    jComboBox1.addItem("Java"); // Добавление элемента
    jComboBox1.addItem("C#"); // Добавление элемента
    jComboBox1.addItem("C++"); // Добавление элемента
    jComboBox1.removeItemAt(0); // Удаление элемента
    jComboBox1.setSelectedIndex(0); // Выбор значения по номеру
    // jComboBox1.setSelectedItem("C++"); // Выбор значения по
названию
}
```

Компонент «JList» (список). Компонент JList служит для выбора какого-либо одного или нескольких текстовых значений из списка.



№ п/п	Свойство	Назначение
1	model	СПИСОК ЗНАЧЕНИЙ СПИСКА (набор текстовых строк, без пустых строк в конце)
2	font	Шрифт текста списка
3	foreground	Цвет текста списка
4	background	Цвет фона списка
5	selectedIndex	Номер выбранной строки (отсчет с 0, -1 – значение не выбрано)
6	selectionMode	Режим выбора значения из списка: SINGLE – выбор одного значения; SINGLE_INTERVAL – выбор произвольного непрерывного интервала значений; MULTIPLE_INTERVAL – выбор произвольного набора значений.
7	layoutOrientation	VERTICAL – все элементы списка отображаются в виде одного столбца  VERTICAL_WRAP - элементы списка отображаются в виде столбцов  HORIZONTAL_WRAP – элементы списка также располагаются в столбцах, но количество элементов по горизонтали фиксировано 
8	visibleRowCount	Количество отображаемых строк (остальные в прокрутке)
9	fixedCellHeight	Высота строк (-1 – аторазмер)
10	fixedCellWidth	Ширина столбцов строк для много столбцового режима (-1 – аторазмер)
11	toolTipText	Всплывающая подсказка
12	border	Дополнительная рамка у компонента
13	componentPopupMenu	Привязка компонента всплывающего меню

14	enabled	Доступность компонента
15	focusable	Разрешение фокуса ввода с клавиатуры
16	opaque	Непрозрачность фона

При добавлении на форму компонента JList предварительно автоматически вставляется компонент JScrollPane, и уже в него вставляется JList. Это дает компоненту JList возможность горизонтальной и вертикальной прокрутки. Свойства «X», «Y», «Ширина», «Высота» перенесены в компонент JScrollPane.

№ п/п	События	Назначение
1	valueChanged	Основное событие на изменение выбранного значения в списке

№ п/п	Методы	Назначение
1	getSelectedIndex()	Получение номера выбранной строки (отсчет с 0, -1 – значение не выбрано)
2	getSelectedValue()	Получение выбранного значения из списка
3	setSelectedIndex()	Установка номера строки выбора (для режима одиночного выбора, отсчет с 0)
4	getSelectedIndices()	Возвращает номера выбранных строк (для режима мульти-выбора, отсчет с 0)
5	getSelectedValuesList()	Возвращает список значений всех выделенных элементов списка (с JDK ver. 1.7)
6	setSelectedIndices()	Установка номеров строк для выбора (для режима мульти-выбора, отсчет с 0)
7	setSelectedValue()	Установка выбора нужной строки по ее значению
8	setSelectionInterval	Установка выбора через указание интервала номеров строк
9	setListData	Установка новых значений списка

Примеры работы с JList:

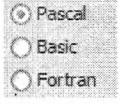
```
private void jList1_valueChanged(javax.swing.event.ListSelectionEvent
evt) {
    // Получение значения выбранной строки в JList
    try {
    jTextField1.setText(jList1.getSelectedValue().toString());
  } catch (Exception e) { jTextField1.setText(""); };
}
```

```
private void jList1_valueChanged(javax.swing.event.ListSelectionEvent
evt) {
    // Получение значения выбранной строки в JList по номеру
выделенной строки
    int k = jList1.getSelectedIndex(); // Получение номера выделенной
стоки
        if (k==-1) return; // Если не выбрана строка то выход их
процедуры
 // Получение выделенной строки
    jTextField1.setText(jList1.getModel().getElementAt(k).toString());
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Получение номеров выбранных строк из JList
    int[] k = jList1.getSelectedIndices(); // Получаем номера выбранных
строк
    String s = Arrays.toString(k); // Преобразуем цифры номеров сток в
строку типа «[0, 2, 5]»
    jTextField1.setText(s); // Выводим номера выбранных строк
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Получение значений выбранных строк из JList
    // Получаем строку типа «[Java, Delphi, Джава]»
    jTextField1.setText(jList1.getSelectedValuesList().toString());
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Установка нужной строки в JList
    jList1.setSelectedIndex(2); // Выбираем 3 стоку (отсчет с 0)
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Установка выбранных строк в  JList
    jList1.setSelectedIndices(new int[] {1,3,4}); // Выбираем строки 2, 4
и 5 (отсчет с 0)
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Установка строки по значению в  JList
```

```
    // Второй параметр означает необходимость автопрокрутки до
выбранной строки
    jList1.setSelectedValue( "Java", true);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Установка выбора строк через указание интервала номеров
строк в JList
    jList1.getSelectionModel().setSelectionInterval(0, 3); // Выделить
первые 4 строки
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Установка новых данных (строк) в JList
    // jList1.setListData(new String[]{""}); // Очистка данных
    // Очистка данных с установкой новых значений
    jList1.setListData(new String[]{"Pavlodar", "Astana", "Moskov"});
    jList1.setSelectedValue("Moskov", true); // Выбор значения по
названию
}
```

Компонент «JRadioButton» (кнопка-переключатель) и «ButtonGroup» (группа кнопок). Если нужно организовать выбор только одного из нескольких вариантов, используется компонент JRadioButton.

Несколько радио-кнопок объединяют в группу с помощью не визуального компонента ButtonGroup (в рамках которого осуществляется выбор). После щелчка мыши на одной из радио-кнопок группы будет выбрана только она.



№ п/п	Свойство	Назначение
1	buttonGroup	Ссылка на компонент группы ButtonGroup
2	actionCommand	Название команды. Данное название придумывается самостоятельно, например это может быть номер пункта: «1» и т.п. Наличие команды обязательно для работы кнопки
3	text	Текстовое значение пункта
4	selected	Выбор пункта (да/нет)
5	font	Шрифт текста списка
6	foreground	Цвет текста списка
7	background	Цвет фона списка
8	icon	Картинка вместо «точки» для невыбранного состояния
9	iconTextGap	Расстояние от картинки «icon» до надписи кнопки
10	selectedIcon	Картинка вместо «точки» для выбранного состояния
11	rolloverIcon	Картинка вместо «точки» при наведении на нее мышки для невыбранного состояния
12	rolloverSelectedIcon	Картинка вместо «точки» при наведении на нее мышки для выбранного состояния
13	horizontalAlignment	Выравнивание текста и картинки кнопки относительно ее размеров по горизонтали
14	horizontalTextPosition	Положение картинки и надписи кнопки относительно друг друга по горизонтали
15	verticalAlignment	Выравнивание текста и картинки кнопки относительно ее размеров по вертикали
16	verticalTextPosition	Положение картинки и надписи кнопки относительно друг друга по вертикали
17	border	Дополнительная рамка у компонента
18	componentPopupMenu	Привязка компонента всплывающего меню
19	enabled	Доступность компонента
20	focusable	Разрешение фокуса ввода с клавиатуры
21	opaque	Непрозрачность фона
22	X	Положение верхнего левого угла компонента, координата X
23	Y	Положение верхнего левого угла компонента, координата Y
24	Ширина	Ширина компонента
25	Высота	Высота компонента

№ п/п	События	Назначение
1	actionPerformed	Основное событие на выбор пункта

Для работы с радио-кнопками необходимо сделать следующее:

- сделать обработчик события для какой-либо радио-кнопки группы, например первой: «jRadioButton1_actionPerformed»;
- указать всем остальным радио-кнопкам, что событие «actionPerformed» должно обрабатываться уже созданным методом «jRadioButton1_actionPerformed». Для этого нужно всем кнопкам, через Инспектор Объектов, в поле события «actionPerformed» вручную вписать строку «jRadioButton1_actionPerformed».

Таким образом для всех радио-кнопок будет вызываться один и тот же метод «jRadioButton1_actionPerformed», в котором и будет производиться обработка выбора одного из значений.

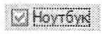
Узнать данные о том, какая именно кнопка нажалась в группе радио-кнопок, можно через метод «getSelection().getActionCommand()» компонента ButtonGroup.

Примеры работы с JRadioButton:

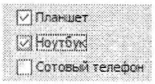
```
private void jRadioButton1_actionPerformed(java.awt.event.ActionEvent
evt) {
    // Выбор значения из группы JRadioButton по имени «акции»
(этот метод нужен один для всех радио-кнопок группы)
    // Получаем название команды (номера кнопки)
    String s = buttonGroup1.getSelection().getActionCommand();
    switch (s) {
      case "1": jTextField1.setText("Oracle"); break;
      case "2": jTextField1.setText("Microsoft"); break;
      case "apple": jTextField1.setText("Apple"); break;
      default: jTextField1.setText("???");
  }
}

private void jRadioButton1_actionPerformed(java.awt.event.ActionEvent
evt) {
    // Выбор значения из группы JRadioButton по ссылке на
выбранную кнопку (этот метод нужен один для всех радио-кнопок
группы)
 // Получаем ссылку на выбранную радио-кнопку
    JRadioButton button = (JRadioButton)evt.getSource();
    jTextField1.setText(button.getText());  // Получаем название
выбранной кнопки-переключателя
```

```
jTextField2.setText(button.getActionCommand()); // Получаем название команды кнопки
}
```

Компонент «JCheckBox» (флажок). Если нужно организовать выбор только одного варианта из одного, то используется компонент JCheckBox. После щелчка мыши на компоненте он будет помечен как выбранный, при повторном щелчке выделение снимется.



Если необходимо сделать несколько разных независимых выборов, то нужно использовать несколько разных компонент JCheckBox: для каждого выбора по одному компоненту. Пример из трех компонент:



№ п/п	Свойство	Назначение
1	actionCommand	Название команды. Данное название придумывается самостоятельно, например это может быть номер пункта: «1» и т.п. Наличие команды обязательно для работы кнопки
2	text	Текстовое значение пункта
3	selected	Выбор пункта (да/нет)
4	font	Шрифт текста списка
5	foreground	Цвет текста списка
6	background	Цвет фона списка
7	icon	Картинка вместо «галочки» ☑ для невыбранного состояния
8	iconTextGap	Расстояние от картинки «icon» до надписи кнопки
9	selectedIcon	Картинка вместо «галочки» для выбранного состояния
10	rolloverIcon	Картинка вместо «галочки» при наведении на нее мышки для невыбранного состояния
11	rolloverSelectedIcon	Картинка вместо «галочки» при наведении на нее мышки для выбранного состояния
12	horizontalAlignment	Выравнивание текста и картинки кнопки относительно ее размеров по горизонтали
13	horizontalTextPosition	Положение картинки и надписи кнопки относительно друг друга по горизонтали
14	verticalAlignment	Выравнивание текста и картинки кнопки

		относительно ее размеров по вертикали
15	verticalTextPosition	Положение картинки и надписи кнопки относительно друг друга по вертикали
16	border	Дополнительная рамка у компонента
17	componentPopupMenu	Привязка компонента всплывающего меню
18	enabled	Доступность компонента
19	focusable	Разрешение фокуса ввода с клавиатуры
20	opaque	Непрозрачность фона
21	X	Положение верхнего левого угла компонента, координата X
22	Y	Положение верхнего левого угла компонента, координата Y
23	Ширина	Ширина компонента
24	Высота	Высота компонента

№ п/п	События	Назначение
1	actionPerformed	Основное событие на выбор пункта

Для работы с кнопкой-флажком необходимо сделать следующее:

- сделать обработчик события для какой-либо кнопки-флажка, например, первой: «jCheckBox1ActionPerformed»;

- указать всем остальным кнопкам-флажкам, что событие «actionPerformed» должно обрабатываться уже созданным методом «jCheckBox1ActionPerformed». Для этого нужно всем кнопкам-флажкам, через Инспектор Объектов, в поле события «actionPerformed» вручную вписать строку «jCheckBox1ActionPerformed».

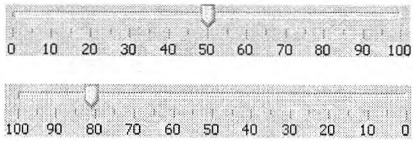
Таким образом для всех кнопок-флажков будет вызываться один и тот же метод «jCheckBox1ActionPerformed», в котором и будет производиться обработка выбора значений.

Примеры работы с JCheckBox:

```
private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Работа с несколькими JCheckBox (этот метод нужен один для
всех флажков "группы")
    JCheckBox button = (JCheckBox)evt.getSource();  // Получаем
ссылку на нажатую кнопку-флажок
    if (button.isSelected()) { // Если флажок выбран
    String s = button.getActionCommand(); // Получаем имя "команды"
кнопки
    switch (s) {
```

```
            case "1": jTextField1.setText("Oracle"); break;
            case "2": jTextField1.setText("Microsoft"); break;
            case "3": jTextField1.setText("Apple"); break;
            default: jTextField1.setText("???");
    } } else jTextField1.setText(""); // Если флажок не выбран
}

private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Работа с одиночным флажком JCheckBox
    if (jCheckBox1.isSelected()) { // Если флажок выбран, то включить
видимость панели
        jPanel1.setVisible(true) } else { jPanel1.setVisible(false); }
}
```

Компонент JSlider (ползунок). Компонент JSlider служит для графического задания значения числовой величины.



№ п/п	Свойство	Назначение
1	value	Текущее значение
2	minimum	Минимальное значение
3	maximum	Максимальное значение
4	orientation	HORIZONTAL – горизонтальное расположение, VERTICAL – вертикальное
5	inverted	Расположение шкалы справа на лево
6	minorTickSpacing	Шаг маленьких делений шкалы
7	majorTickSpacing	Шаг больших делений шкалы
8	paintLabels	Отображать числовые значения шкалы
9	paintTicks	Отображать деления шкалы
10	paintTrack	Отображать ось шкалы
11	snapToTicks	Устанавливать ползунок ровно по меткам шкалы
12	font	Шрифт текста списка
13	foreground	Цвет текста списка
14	background	Цвет фона списка
15	toolTipText	Всплывающая подсказка
16	border	Дополнительная рамка у компонента
17	borderPainted	Разрешение отображения рамки border

18	cursor	Вид курсора
19	componentPopupMenu	Привязка компонента всплывающего меню
20	enabled	Доступность компонента
21	focusable	Разрешение фокуса ввода с клавиатуры
22	opaque	Непрозрачность фона
23	X	Положение верхнего левого угла компонента, координата X
24	Y	Положение верхнего левого угла компонента, координата Y
25	Ширина	Ширина компонента
26	Высота	Высота компонента

№ п/п	События	Назначение
1	stateChanged	Основное событие на изменение значения ползунка

№ п/п	Методы	Назначение
1	getValue()	Текущее значение
2	setValue()	Установить текущее значение
3	setMinimum()	Установить минимальное значение
4	setMaximum()	Установить максимальное значение
5	getMinimum()	Получить минимальное значение
6	getMaximum()	Получить максимальное значение

Пример работы JSlider:

```
private void jSlider1stateChanged(javax.swing.event.ChangeEvent evt) {
    // Изменение значения слайдера JSlider
    jProgressBar1.setValue(jSlider1.getValue());
}
```

Компонент JScrollBar (полоса прокрутки). Компонент JScrollBar служит для графического задания значения числовой величины.

№ п/п	Свойство	Назначение
1	value	Текущее значение
2	minimum	Минимальное значение
3	maximum	Максимальное значение

4	orientation	HORIZONTAL – горизонтальное расположение, VERTICAL – вертикальное
5	unitIncrement	Шаг изменения значения на щелчке по стрелкам компонента
6	blockIncrement	Шаг изменения значения на щелчке по самому компоненту
7	visibleAmount	Ширина ползунка (максимальное значение, которое можно установить ползунком, будет равно maximum- visibleAmount). Рекомендуется установить в ноль
8	toolTipText	Всплывающая подсказка
9	border	Дополнительная рамка у компонента
10	componentPopupMenu	Привязка компонента всплывающего меню
11	cursor	Вид курсора
12	enabled	Доступность компонента
13	focusable	Разрешение фокуса ввода с клавиатуры
14	opaque	Непрозрачность фона
15	X	Положение верхнего левого угла компонента, координата X
16	Y	Положение верхнего левого угла компонента, координата Y
17	Ширина	Ширина компонента
18	Высота	Высота компонента

№ п/п	События	Назначение
1	adjustmentValueChanged	Основное событие на изменение значения

№ п/п	Методы	Назначение
1	getValue()	Текущее значение
2	setValue()	Установить текущее значение
3	setValues()	Установить текущее значение, ширину ползунка, минимальное значение, максимальное значение
4	setMinimum()	Установить минимальное значение
5	setMaximum()	Установить максимальное значение
6	getMinimum()	Получить минимальное значение
7	getMaximum()	Получить максимальное значение

Примеры работы с JScrollBar:

```
private void
jScrollBar1_adjustmentValueChanged(java.awt.event.AdjustmentEvent
evt) {
    // Получение значения из JScrollBar
    jProgressBar1.setValue(jScrollBar1.getValue());
```

```
        jTextField1.setText(Integer.toString(jScrollBar1.getValue()));
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // Установка значения в JScrollBar
       // jScrollBar1.setValue(20); // Установить текущее значение, равное
20
        jScrollBar1.setValues(50,0,1,100); // Установить новые значения
данных (текущее значение = 50, минимум = 1, максимум = 100)
}
```

Компонент «JSpinner» (счетчик). Класс JSpinner представляет собой компонент для ввода (выбора) значения с возможностью его уменьшения и увеличения без использования клавиатуры, используя мышь.

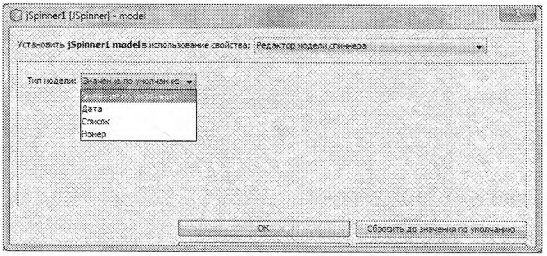
№ п/п	Свойство	Назначение
1	model	МОДЕЛЬ СЧЕТЧИКА
2	value	Значение счетчика
3	font	Шрифт текста
4	opaque	Непрозрачность фона
5	border	Рамка
6	enabled	Доступность компонента
7	focusable	Разрешение фокуса ввода/работы с текстом
8	cursor	Вид курсора
9	toolTipText	Всплывающая подсказка над компонентом
10	componentPopupMenu	Привязка компонента всплывающего меню
11	X	Положение верхнего левого угла компонента, координата X
12	Y	Положение верхнего левого угла компонента, координата Y
13	Высота	Высота компонента
14	Ширина	Ширина компонента

№ п/п	События	Назначение
1	stateChanged	Основное событие на изменение значения

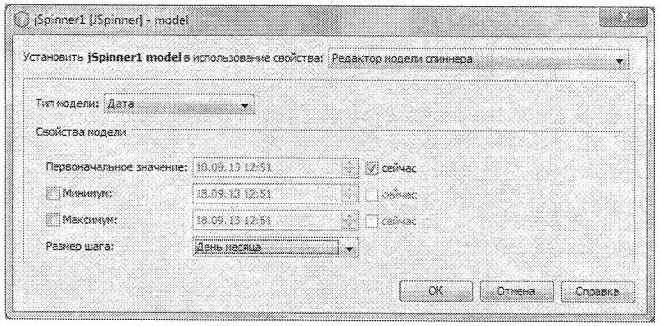
№ п/п	Методы	Назначение
1	getValue()	Получить значение

2	setValue()	Установить значение
3	getNextValue()	Получить следующее значение

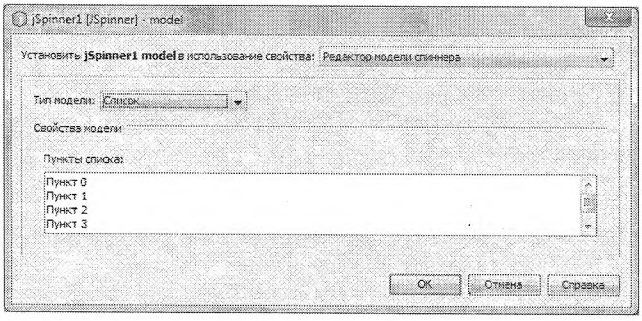
Модели счетчика (значение свойства model) JSpinner:



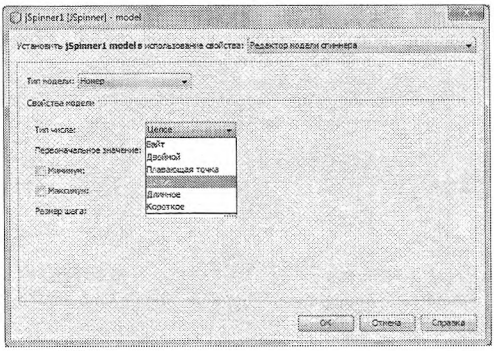
Модели счетчика – «Дата» (значение свойства model):



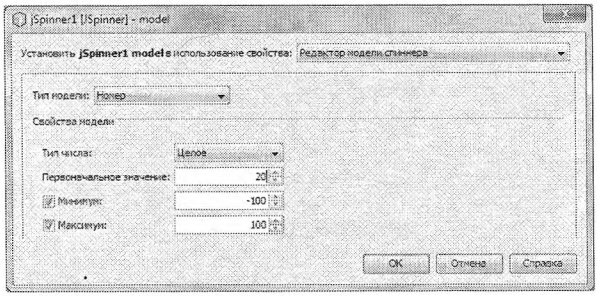
Модели счетчика – «Список» (значение свойства model):



Модели счетчика – «Номер» (значение свойства model):



Модели счетчика – «Номер – Целое» (значение свойства model):

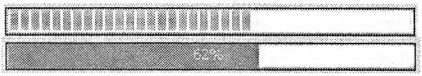


Примеры работы с JSpinner:

```
private void jSpinner1_stateChanged(javax.swing.event.ChangeEvent evt)
{
    // Установка и чтение значения из JSpinner
    jSpinner1.setValue(12); // Устанавливаем значение 12
    jLabel1.setText((jSpinner1.getValue()).toString()); // Получаем
значение из счетчика
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Работа с JSpinner (model:  «Номер – Целое»)
    try {
```

```
        int k = (Integer) jSpinner1.getNextValue(); // Считываем
следующее значение
        jSpinner1.setValue(k); // Устанавливаем новое значение
    } catch (Exception e) {  }
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // Работа с JSpinner (model: «Номер – Двойной»)
      try {
        double k = (Double) jSpinner1.getValue()+12.23; // Считываем
значение и увеличиваем его на 12.23
        jSpinner1.setValue(k); // Устанавливаем новое значение
    } catch (Exception e) {  }
}
```

Компонент JProgressBar (индикатор выполнения). Компонент JProgressBar служит для графического отображения значения числовой величины.



№ п/п	Свойство	Назначение
1	value	Текущее значение
2	minimum	Минимальное значение
3	maximum	Максимальное значение
4	orientation	0 – горизонтальное расположение, 1 – вертикальное
5	stringPainted	Разрешение отображения текстовой надписи значения в процентах
6	string	Надпись текущего значения в процентах (при разрешенном stringPainted)
7	indeterminate	Включение неопределенного режима («бесконечно бегущего») индикатора
8	font	Шрифт текста
9	foreground	Цвет текста
10	background	Цвет фона
11	toolTipText	Всплывающая подсказка
12	border	Дополнительная рамка у компонента
13	borderPainted	Разрешение отображения рамки border
14	cursor	Вид курсора
15	componentPopupMenu	Привязка компонента всплывающего меню

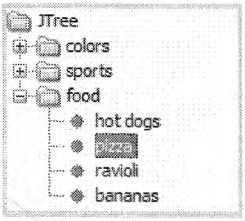
16	enabled	Доступность компонента
17	focusable	Разрешение фокуса ввода с клавиатуры
18	opaque	Непрозрачность фона
19	X	Положение верхнего левого угла компонента, координата X
20	Y	Положение верхнего левого угла компонента, координата Y
21	Ширина	Ширина компонента
22	Высота	Высота компонента

№ п/п	Методы	Назначение
1	getValue()	Текущее значение
2	getString()	Текущее значение в процентах (текст за знаком «%»)
3	getMinimum()	Получить минимальное значение
4	getMaximum()	Получить максимальное значение
5	getPercentComplete()	Текущее значение в процентах /100 (значения от 0.0 до 1.0)
6	setValue()	Установить текущее значение
7	setMinimum()	Установить минимальное значение
8	setMaximum()	Установить максимальное значение
9	setIndeterminate	Включение/выключение неопределенного режима («бесконечно бегущего») индикатора

Пример работы с JProgressBar:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
  // Работа с JProgressBar в отдельном потоке для прорисовки
изменений
  new Thread(new Runnable() {
   public void run() {
    // Цикл от минимума до максимума значений
    for (int i = jProgressBar1.getMinimum(); i <=
jProgressBar1.getMaximum(); i++) {
     jProgressBar1.setValue(i);
     try {
      Thread.sleep(10);
     } catch (InterruptedException ex) {
     }
    } }
  }).start();
}
```

JTree (дерево). Компонент JTree служит для отображения древовидной структуры данных и выбора нужного значения.



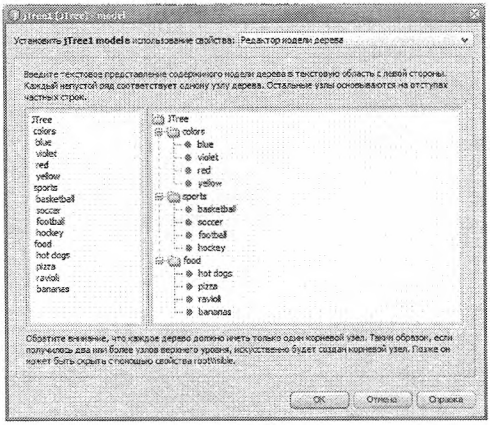
№ п/п	Свойство	Назначение
1	model	МОДЕЛЬ ДЕРЕВА
2	rootVisible	Запрет показа корневого узла и подузлов (остаются видимыми только дочерние к корневому узлу узлы)
3	editable	Разрешение редактирования значений дерева
4	showsRootHandles	Показать значок «-» у корневого узла
5	font	Шрифт надписей вкладки
6	foreground	Цвет фона компонента
7	border	Вид рамки у панели
8	toolTipText	Всплывающая подсказка

№ п/п	События	Назначение
1	valueChanged	Выбор нужного узла/значения в дереве

№ п/п	Методы	Назначение
1	getSelectionPath	Получение выбранной ветви (значения) (строка типа «[JTree, food, pizza]»)

При добавлении на форму компонента JTree предварительно автоматически вставляется компонент JScrollPane, и уже в него вставляется JTree. Это дает компоненту JTree возможность горизонтальной и вертикальной прокрутки. Свойства «X», «Y», «Ширина», «Высота» находятся в компоненте JScrollPane.

Редактор Модели JTree (свойство model):

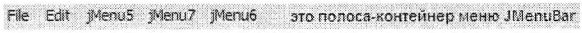


Примеры работы с JTree:

```
private void jTree1ValueChanged(javax.swing.event.TreeSelectionEvent
evt) {
    // Получение выбранной ветви (значения) в JTree
    // Выдаст строку типа «[JTree, food, pizza]»
    JOptionPane.showMessageDialog(rootPane,

jTree1.getSelectionPath(),"Выбрано",JOptionPane.INFORMATION_MES
SAGE);
}
```

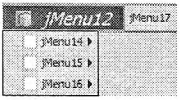
3.9 Создание меню

Компонент JMenuBar (строка меню). Компонент-контейнер JMenuBar служит для создания главного основного меню в окне. На полосу JMenuBar наносятся компоненты-пункты JMenu.



№ п/п	Свойство	Назначение
1	background	Цвет фона полосы меню
2	opaque	Непрозрачность фона полосы меню
3	border	Дополнительная рамка по контуру полосы меню
4	toolTipText	Всплывающая подсказка

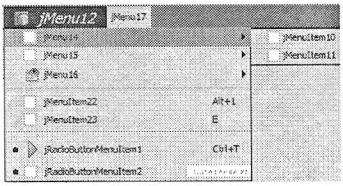
Компонент JMenu (меню/выпадающее подменю). Компоненты JMenu служат для создания пунктов основного меню JMenuItem и выпадающих подпунктов подменю.



№ п/п	Свойство	Назначение
1	text	Наименование пункта меню/подменю
2	mnemonic	Символ, нажатие которого совместно с клавишей «Alt», выберет данный пункт меню/подменю («горячая клавиша»), а при повторном нажатии на данный символ активирует сам пункт
3	enabled	Доступность пункта меню/подменю
4	font	Шрифт пункта
5	foreground	Цвет текста пункта
6	background	Цвет фона пункта
7	toolTipText	Всплывающая подсказка
8	icon	Картинка пункта меню (формата jpeg, gif, png)
9	disabledIcon	Картинка пункта меню при недоступности пункта (enabled=false)
10	iconTextGap	Расстояние от картинки «icon» до надписи кнопки
11	horizontalAlignment	Выравнивание текста и картинки кнопки относительно ее размеров по горизонтали
12	horizontalTextPosition	Положение картинки и надписи кнопки относительно друг друга по горизонтали
13	verticalAlignment	Выравнивание текста и картинки кнопки относительно ее размеров по вертикали
14	verticalTextPosition	Положение картинки и надписи кнопки относительно друг друга по вертикали

№ п/п	Событие	Назначение
1	menuKeyTyped	Активация пункта «горячей клавишей» или клавишей «Enter» при выборе пункта стрелками клавиатуры
2	mouseClicked	Активация пункта щелчком мыши

Компонент JMenuItem (пункт меню) и JRadioButtonMenuItem (пункт меню/переключатель). Компоненты JMenuItemи и JRadioButtonMenuItem служат для создания пунктов для JMenu.



№ п/п	Свойство	Назначение
1	text	Наименование пункта
2	accelerator	Комбинация клавиш для открытия подменю, где находится пункт (но выбора и активизации самого пункта при этом не происходит !) При повторном нажатии комбинации активируется пункт
3	mnemonic	Символ, нажатие которого активирует данный пункт (только при отрытом подменю, где располагается пункт)
4	selected	Установка/снятие «галочки» с пункта (только для JRadioButtonMenuItem)
5	buttonGroup	Ссылка на компонент типа ButtonGroup для работы в группе (режиме выбора одной «точки» из нескольких (только для JRadioButtonMenuItem)
6	enabled	Доступность пункта меню
7	font	Шрифт пункта
8	foreground	Цвет текста пункта
9	background	Цвет фона пункта
10	toolTipText	Всплывающая подсказка
11	icon	Картинка пункта меню (формата jpeg, gif, png)
12	disabledIcon	Картинка пункта меню при недоступности пункта (enabled=false)
13	iconTextGap	Расстояние от картинки «icon» до надписи кнопки
14	horizontalAlignment	Выравнивание текста и картинки кнопки относительно ее размеров по горизонтали
15	horizontalTextPosition	Положение картинки и надписи кнопки относительно друг друга по горизонтали
16	verticalAlignment	Выравнивание текста и картинки кнопки относительно ее размеров по вертикали
17	verticalTextPosition	Положение картинки и надписи кнопки относительно друг друга по вертикали

№ п/п	События	Назначение
1	actionPerformed	Активация пункта

№ п/п	Методы	Назначение
1	isSelected()	Проверка установки «галочки» (только для JRadioButtonMenuItem)
2	doClick()	Активация пункта (вызов метода, обрабатывающего событие actionPerformed)

Пример работы с JRadioButtonMenuItem:

```
 private void
jRadioButtonMenuItem6ActionPerformed(java.awt.event.ActionEvent evt)
{
 // Пример работы с JRadioButtonMenuItem
   if (jRadioButtonMenuItem6.isSelected())
JOptionPane.showMessageDialog(rootPane, "да");
}
```

Компонент JSeparator (разделитель). Компонент JSeparator служит для визуального разделения пунктов и подпунктов меню.

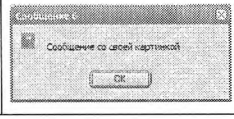
№ п/п	Свойство	Назначение
1	foreground	Цвет линии

Компонент JPopupMenu (всплывающее меню). Компонент JPopupMenu служит для создания всплывающего меню (вызываемого правой кнопкой мыши) для других компонент, через их свойство componentPopupMenu для привязки к нужному компоненту JPopupMenu.

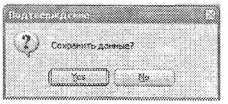
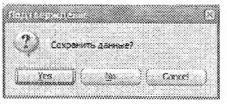
Всплывающее меню JPopupMenu конструируется с помощью компонент JMenu, JMenuItem и JRadioButtonMenuItem.

3.10 Диалоги

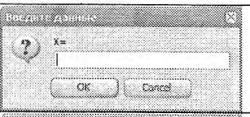
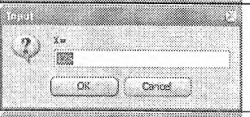
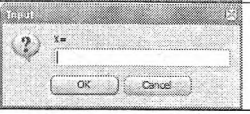
Диалоги вывода JOptionPane.showMessageDialog. Диалоги вывода JOptionPane.showMessageDialog служат для отображения в отдельном всплывающем окошке нужного сообщения с нужной картинкой.

№ п/п	Программный код	Результат
1	JOptionPane.showMessageDialog(null,"Ошибка!", "Сообщение 1",0); или JOptionPane.showMessageDialog(null, "Ошибка!", "Сообщение 1", JOptionPane.ERROR_MESSAGE);	
2	JOptionPane.showMessageDialog(null, "Информирование!", "Сообщение 2", 1); или JOptionPane.showMessageDialog(null, "Информирование!", "Сообщение 2", JOptionPane.INFORMATION_MESSAGE);	
3	JOptionPane.showMessageDialog(null, "Предупреждение!", "Сообщение 3", 2); или JOptionPane.showMessageDialog(null, "Предупреждение!", "Сообщение 3", JOptionPane.WARNING_MESSAGE);	
4	JOptionPane.showMessageDialog(null, "Вопрос!", "Сообщение 4", 3); или JOptionPane.showMessageDialog(null, "Вопрос!", "Сообщение 4", JOptionPane.QUESTION_MESSAGE);	
5	JOptionPane.showMessageDialog(null, "Просто текст без картинки", "Сообщение 5", -1); или JOptionPane.showMessageDialog(null, "Просто текст без картинки", "Сообщение 5", JOptionPane.PLAIN_MESSAGE);	
6	JOptionPane.showMessageDialog(null, "Сообщение со своей картинкой", "Сообщение 6", 0, new javax.swing.ImageIcon(getClass().getResource("/mypackage/6.png")));	

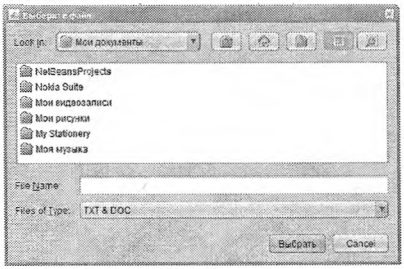
Диалоги подтверждения JOptionPane.showConfirmDialog. Диалоги подтверждения JOptionPane.showConfirmDialog служат для вывода в отдельном всплывающем окошке сообщением с возможностью выбора нужного ответа.

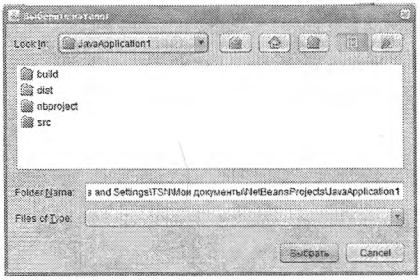
№ п/п	Программный код	Результат
1	if (JOptionPane.showConfirmDialog(null, "Сохранить данные?", "Подтверждение", 0) == JOptionPane.YES_OPTION) { JOptionPane.showMessageDialog(null, "Да", "Сообщение", 1); };	
2	if (JOptionPane.showConfirmDialog(null, "Сохранить данные?", "Подтверждение", 2) == JOptionPane.OK_OPTION) { JOptionPane.showMessageDialog(null, "Да", "Сообщение", 1); };	
3	if (JOptionPane.showConfirmDialog(null, "Сохранить данные?", "Подтверждение", 1) == JOptionPane.NO_OPTION) { JOptionPane.showMessageDialog(null, "Нет", "Сообщение", 1); };	

Диалоги ввода JOptionPane.showInputDialog. Диалоги ввода JOptionPane.showInputDialog служат для ввода в отдельном всплывающем окошке нужного значения.

№ п/п	Программный код	Результат
1	String s = JOptionPane.showInputDialog(null, "X=", "Введите данные", 3);	
2	String s = JOptionPane.showInputDialog(null, "X=", "125");	
3	String s = JOptionPane.showInputDialog(null, "X=");	

Диалоги выбора файла / каталога JFileChooser. Диалоги выбора JFileChooser служат запроса у пользователя имени файла или каталога.





JFileChooser можно добавить в любое место пользовательского интерфейса, поскольку это весьма гибкий компонент, позволяющий тонко настраивать внешний вид. При необходимости можно полностью изменить стандартное расположение входящих в JFileChooser компонентов и добавить дополнительные элементы, такие как панели предварительного просмотра файлов.

Основные методы JFileChooser:

Метод	Описание
File getCurrentDirectory()	Функция чтения текущей директории
String getDialogTitle()	Функция чтения заголовка окна
int getDialogType()	Функция чтения типа диалогового окна
FileFilter getFileFilter()	Функция чтения текущего фильтра
File getSelectedFile()	Функция чтения выделенного файла
File[] getSelectedFiles()	Функция получения списка выделенных файлов, если установлен флаг выделения нескольких файлов MULTI_SELECTION_ENABLED_CHANGED_PROPERTY
void setCurrentDirectory(File dir)	Метод определения текущей директории
void setDialogTitle(String dialogTitle)	Метод определения заголовка диалогового окна

void setDialogType(int dialogType)	Метод определения типа диалогового окна
void setFileFilter(FileFilter filter)	Метод установки файлового фильтра
void setFileSelectionMode(int mode)	Метод определения выделяемых объектов – файлы, директории или файлы с директориями
void setMultiSelectionEnabled(boolean b)	Метод определения возможности выделения нескольких файлов
void setSelectedFile(File file)	Метод выделения файла
void setSelectedFiles(File[] selectedFiles)	Метод выделения списка файлов, если установлен флаг выделения нескольких файлов MULTI_SELECTION_ENABLED_CHANGED_PROPERTY
int showDialog(Component parent, String approveButtonText)	Функция открытия окна выбора файла с настроенным наименованием кнопки
int showOpenDialog(Component parent)	Функция открытия диалогового окна «Открыть файл»
int showSaveDialog(Component parent)	Функция открытия диалогового окна «Сохранить файл»

Вовращаемые компонентом JFileChooser значения:

- APPROVE_OPTION – выбор файла в диалоговом окне прошел успешно; выбранный файл можно получить методом getFile();
- CANCEL_OPTION – выбор файла отменен нажатием на кнопке Cancel;
- ERROR_OPTION – при выборе файла произошла ошибка, или было закрыто диалоговое окно выбора файла.

Пример работы с JFileChooser:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Запрос имени файла
    JFileChooser chooser = new JFileChooser(); // Создаем объект выбора имени файла
    chooser.setDialogTitle("Выберите файл"); // Заголовок окна диалога
    chooser.setFileSelectionMode(JFileChooser.FILES_ONLY); // Выбор только файлов
    // Установить первое значение фильтра по-умолчанию
    chooser.setFileFilter(new FileNameExtensionFilter("TXT & DOC", "txt", "doc"));
    // Добавить дополнительные фильтры
    chooser.addChoosableFileFilter(new FileNameExtensionFilter("MP3 & AVI", "mp3", "avi"));
```

```
    chooser.addChoosableFileFilter(new FileNameExtensionFilter("JPG
& PNG", "jpg","png"));
    chooser.setAcceptAllFileFilterUsed(false); // Запретить показывать
фильтр "All files"
    int ret = chooser.showDialog(null, "Выбрать"); // Вызываем диалог
    if (ret != JFileChooser.APPROVE_OPTION) { return; } // Если файл
не выбран, то выход
    //File file = chooser.getSelectedFile(); // Подключение к файлу
    JOptionPane.showMessageDialog(null,
chooser.getSelectedFile().getName(), "Имя файла", 1);
    JOptionPane.showMessageDialog(null,
chooser.getSelectedFile().getPath(), "Полный путь с именем файла", 1);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Запрос имени каталога
    JFileChooser chooser = new JFileChooser(); // Создаем объект
выбора имени файла
    chooser.setDialogTitle("Выберите каталог"); // Заголовок окна
диалога
    chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
// Выбор только каталогов
    //
chooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES
); // Выбор или каталога или файла
    chooser.setAcceptAllFileFilterUsed(false); // Отключить выбор типа
файлов
    chooser.setCurrentDirectory(new
File(".").getAbsoluteFile().getParentFile()); // Установка текущего
каталога
    int ret = chooser.showDialog(null, "Выбрать"); // Вызываем диалог
    if (ret != JFileChooser.APPROVE_OPTION) { return; } // Если
каталог не выбран, то выход
    JOptionPane.showMessageDialog(null,
chooser.getSelectedFile().getName(), "Имя каталога", 1);
    JOptionPane.showMessageDialog(null,
chooser.getSelectedFile().getPath(), "Полный путь с именем каталога",
1);
}
```

3.11 Работа с файлами

Концепция работы с файлами в Java включает две составляющие:

- работа с файлами и папками с помощью объектов типа File. Обеспечивает работу с именами файлов (проверка существования файла или папки с заданным именем, нахождение абсолютного пути по относительному и наоборот, проверка, и установка атрибутов файлов и папок);

- работа с потоками ввода/вывода. Обеспечивает взаимодействие не только с файлами, но и с памятью, а также с различными устройствами ввода/вывода.

Работа с файлами и папками с помощью объектов типа File. Объекты типа File могут рассматриваться как абстракции, инкапсулирующие работу с именами файлов и папок. При этом папка считается разновидностью файла, обладающей особыми атрибутами.

Объект типа File создается с помощью конструкторов, имеющих следующие варианты:

- File(Имя папки);
- File(Имя файла);
- File(Имя папки, Имя файла).

При этом имена могут быть как короткими (локальными) без указания пути к файлу или папке, так и длинными (абсолютными) с указанием пути. В таблице приведены важнейшие файловые операции, заданные в классе File. При этом файлы (папки) ищутся по имени в соответствии с правилами поиска файлов в операционной системе. Для платформы Windows символ \ в строках, соответствующих путям, следует заменять последовательностью символов \\.

Поле или метод	Что содержит или делает
Переменные класса	
String pathSeparator	Содержит строку с символом разделителя пути в операционной системе. Это "/" в UNIX–подобных системах и "\" в Windows
char pathSeparatorChar	Содержит символ разделителя пути в операционной системе. Это '/' в UNIX–подобных системах и '\' в Windows
String separator	Содержит строку с символом разделителя между именами файлов и файловых масок в операционной системе
char separatorChar	Содержит символ разделителя между именами файлов и файловых масок в операционной системе
Проверка параметров файла или папки	
boolean exists()	Возвращает true, когда файл (или папка) с заданным

	в конструкторе именем существует, иначе – false
long length()	Возвращает длину файла в байтах в случае, когда файл с заданным в конструкторе именем существует и не является папкой, иначе – 0L
boolean canRead()	Возвращает true, когда файл (или папка) с заданным в конструкторе именем существует и доступен по чтению, иначе – false. (В UNIX-подобных системах существуют файлы, доступные только по записи.) Может возбуждать SecurityException
boolean setReadOnly()	Возвращает true в случае, когда файл (или папка) с заданным в конструкторе именем существует, и ему удалось установить статус "доступен только по чтению", иначе – false
boolean canWrite()	Возвращает true, когда файл (или папка) с заданным в конструкторе именем существует и доступен по записи, иначе – false. (В операционных системах существуют файлы, доступные только по чтению.) Может возбуждать SecurityException
boolean canWrite()	Возвращает true, когда файл (или папка) с заданным в конструкторе именем существует и доступен по записи, иначе – false. (В операционных системах существуют файлы, доступные только по чтению.) Может возбуждать SecurityException
boolean isDirectory()	Возвращает true в случае, когда файл или папка с заданным в конструкторе именем существует и является папкой, иначе – false
boolean isFile()	Возвращает true, когда файл или папка с заданным в конструкторе именем существует и является файлом, иначе – false
boolean isHidden()	Возвращает true, когда файл или папка с заданным в конструкторе именем существует и является скрытым, иначе – false. В UNIX-образных системах скрытыми являются файлы, имена которых начинаются с точки, в Windows – те, которые имеют атрибут hidden (скрытый)
long lastModified()	Возвращает время последней модификации файла, если он существует и доступен по чтению, иначе – 0L. Время отсчитывается в миллисекундах, прошедших с 0 часов 1 января 1970 года (по Гринвичу)
Boolean setLastModified(long time)	Устанавливает время последней модификации файла. Возвращает true, если он существует и доступен по записи, иначе – false. Время отсчитывается в миллисекундах, прошедших с 0 часов 1 января 1970 года (по Гринвичу)
Путь и имя файла (папки)	
String getName()	Возвращает короткое имя файла или папки
String getParent()	Возвращает абсолютное имя родительской папки, т.

	е. папки, в которой находится файл (или папка), соответствующий файловому объекту
String getAbsolutePath()	Возвращает абсолютный путь к файлу или папке, включая имя файла. При этом если в имени файла в конструкторе была задана относительная адресация, то соответствующая часть пути сохраняется в возвращаемой строке
String getCanonicalPath()	Возвращает абсолютный путь к файлу или папке, включая имя файла. При этом если в имени файла в конструкторе была задана относительная адресация, то соответствующая часть пути заменяется в возвращаемой строке на канонический вариант адресации – без элементов относительной адресации. Возбуждает IOException, если канонический путь не может быть построен
int compareTo(File f)	Сравнивает имена файлов (папок), сопоставляемых текущему файловому объекту и объекту f. Возвращает 0 в случае, когда абсолютные имена файлов (папок) совпадают, иначе возвращает число, зависящее от разницы в длинах имен и кодов составляющих их символов. Сравнение зависимо от операционной системы – в UNIX-образных системах регистр символов имеет значение, в Windows – не имеет. Соответствие понимается абстрактно на уровне имен и путей (самих файлов может не существовать)
boolean isAbsolute()	Возвращает true в случае, когда адресация к имени файла (папки) текущего файлового объекта является абсолютной. Хотя может содержать элементы относительной адресации, т. е. не быть канонической
boolean equals(Object obj)	Возвращает true тогда и только тогда, когда текущий объект и параметр obj соответствуют одному и тому же файлу (папке) (с учетом правил о путях и регистрах символов, задаваемых операционной системой). Соответствие понимается абстрактно на уровне имен и путей (самих файлов может не существовать)
Создание/уничтожение/переименование файлов и папок	
boolean createNewFile()	Предпринимает попытку создания файла или папки по имени, которое было задано в конструкторе объекта. В случае успеха возвращается true, иначе – false. Возбуждает IOException, если файл не может быть создан (например, уже существует)
File createTempFile(String prefix, String suffix) File createTempFile(String	Метод класса. Обеспечивает создание пустого файла (или папки), задаваемого коротким именем prefix+suffix в папке операционной системы, предназначенной для временных файлов.

prefix, String suffix, File folder)	Возвращает ссылку на объект. Префикс должен быть не менее трех символов. Возбуждает IOException, если файл не может быть создан (например, уже существует)
boolean mkdir()	Предпринимает попытку создания папки по имени, которое было задано в конструкторе объекта. Возвращает true в случае успешного создания и false в других случаях
boolean mkdirs()	Предпринимает попытку создания папки по имени, которое было задано в конструкторе объекта, причем заодно создаются все папки, заданные в пути, если они не существовали. Возвращает true в случае успешного создания и false в других случаях
boolean delete()	Предпринимает попытку удаления файла или папки по имени, которое было задано в конструкторе объекта. Возвращает true в случае успешного удаления и false в других случаях
boolean renameTo(File dest)	Предпринимает попытку переименования файла или папки с имени, которое было задано в конструкторе объекта, на новое, задаваемое параметром dest. Возвращает true в случае успешного переименования и false в других случаях
Создание нового файлового объекта с помощью имеющегося	
File getAbsoluteFile()	Создает новый файловый объект по абсолютному пути, соответствующему текущему файловому объекту
File getCanonicalFile()	Создает новый файловый объект по каноническому пути, соответствующему текущему файловому объекту. Возбуждает IOException, если канонический путь не может быть построен
File getParentFile()	Создает новый файловый объект по абсолютному пути, соответствующему родительской папке для текущего файлового объекта
Списки папок и файлов	
String[] list() String[] list(FilenameFilter filter)	Возвращает массив строк (список) коротких имен, находящихся в папке файлов и папок. Имена элементов, находящихся во вложенных папках, не показываются. Если файловый объект не соответствует существующей папке, то возвращает null. При наличии фильтра возвращаются только те имена, которые соответствуют маске фильтра
File[] listFiles() File[] listFiles(FilenameFilter filter)	Возвращает массив файловых объектов, соответствующих файлам и папкам, вложенным в ту папку, из файлового объекта которой делается вызов. Элементы, находящиеся во вложенных папках, не учитываются. Если текущий файловый объект не соответствует существующей папке, то возвращает null. При наличии фильтра

	возвращаются объекты только для тех имен, которые соответствуют маске фильтра
File[] listRoots()	Возвращает массив файловых объектов, соответствующих возможным на данном компьютере корневым папкам. В UNIX это папка с именем "/", в Windows – корневые папки всех возможных дисков

Пример суммы первых двух чисел из текстового файла:

```
package tsn01.fsc;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class App1 {
  public static void main(String[] args) throws FileNotFoundException,
IOException {
    // Сумма 2 чисел из текстового файла
    // Открытие входного файла
    Scanner in = new Scanner(new BufferedReader(new
FileReader("d:/input.txt")));
    // Открытие выходного файла
    PrintWriter out = new PrintWriter(new FileWriter("d:/output.txt"));
    int a = in.nextInt(); // Считывает первое целое число из файла
    int b = in.nextInt(); // Считывает второе целое число из файла
    out.println(a + b); // Заносит в выходной файл сумму двух
значений, считанных из входного файла
    in.close(); // Закрытие входного файла
    out.flush(); // Сохранение выходного файла
    out.close();// Закрытие выходного файла
  }
}
```

Пример суммы всех целых чисел в текстовом файле:

```
package tsn01.fsc;
import java.io.File;
```

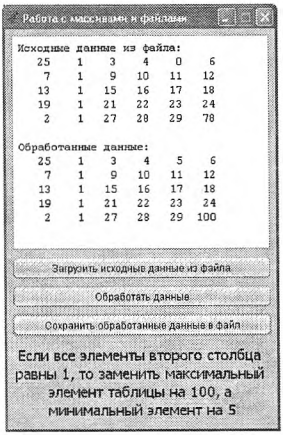
```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.util.Scanner;

public class App1 {
  public static void main(String[] args) throws FileNotFoundException,
IOException {
    // Сумма целых чисел тестового файла
    try {
      // Определение каталога запуска программы (текущего
каталога)
      String dir = new
File(".").getAbsoluteFile().getParentFile().getAbsolutePath() +
System.getProperty("file.separator");
      // Открытие входного файла
      Scanner in = new Scanner(new FileInputStream(dir + "input.txt"));
      // Открытие выходного файла
      PrintWriter out = new PrintWriter(new OutputStreamWriter(new
FileOutputStream(dir + "output.txt")));
      int s = 0; // Переменная для суммы чисел
      while (in.hasNextInt()) { // Пока есть целые числа в файле
        // Считывает текущее целое значение из файла и добавляет
его в переменную
        s += in.nextInt();
    }
      out.println(s); // Вывод в выходной файл суммы всех чисел
входного файла
      System.out.println("Sum = " + s); // Вывод на экран суммы всех
чисел входного файла
      out.flush(); // Сохранение выходного файла
      out.close(); // Закрытие выходного файла
      in.close(); // Закрытие входного файла
  } catch (Exception e) { // Если будет ошибка
      System.out.println("Error ..."); // Сообщение в случаи ошибки
  };
}}
```

Пример копирования данных из файла в файл:

```
package tsn01.fsc;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.util.Scanner;

public class App1 {
  public static void main(String[] args) throws FileNotFoundException,
IOException {
    // Копирование данных из файла в файл
    try {
      // Определение каталога запуска программы (текущего
каталога)
      String dir = new
File(".").getAbsoluteFile().getParentFile().getAbsolutePath()+
System.getProperty("file.separator");
      // Открытие входного файла
      Scanner in = new Scanner(new FileInputStream(dir + "input.txt"),
"cp866"); // или «cp1251»
      // Открытие выходного файла        PrintWriter out = new
PrintWriter(new OutputStreamWriter(new FileOutputStream(dir +
"output.txt"), "cp866")); // или «cp1251»
      while (in.hasNext()) { // Пока есть строки в файле
        // Считывает текущее текстовое значение из входного файла
и добавляет его в конец выходного файла
        String s = in.next(); // Чтение текстового данного из входного
файла
        System.out.println(s); // Вывод считанного данного на экран
        out.println(s);// Добавление считанного данного в выходной
файл
    }
      out.flush(); // Сохранение выходного файла
      out.close(); // Закрытие выходного файла
      in.close(); // Закрытие входного файла
  } catch (Exception e) { // Если будет ошибка
```

```
            System.out.println("Error ..."); // Сообщение в случаи ошибки
        };
    }
}
```

Пример визуальной программы работы с файлами и массивами:




```
...

public class Form1 extends javax.swing.JFrame {

    public String FileName, DirName; // Имя входного файла с данными и
его каталог
    public int mass[][] = new int[5][6]; // Массив для обработки данных

    public Form1() {
        initComponents();
}

...

private void jButton_InputActionPerformed(java.awt.event.ActionEvent
evt) {
        // Загрузка исходных данных из файла в массив
```

```
    JFileChooser chooser = new JFileChooser(); // Создаем объект
выбора файла
            // Устанавливаем текущий каталог
    chooser.setCurrentDirectory(new
File(".").getAbsoluteFile().getParentFile());
    chooser.setFileFilter(new FileNameExtensionFilter("TXT files",
"txt")); // Устанавливаем фильтр
    chooser.setDialogTitle("Выбор файла"); // Устанавливаем значение
диалогу
    chooser.setAcceptAllFileFilterUsed(false); // Выключаем значение
"все файлы" в фильтре
    int ret = chooser.showDialog(null, "Выбрать"); // Вызываем диалог
    if (ret != JFileChooser.APPROVE_OPTION) {
      return;
  } // Если файл не выбран, то выход
    FileName = chooser.getSelectedFile().getPath(); // Получение имени
файла
    DirName = chooser.getSelectedFile().getParent() +
System.getProperty("file.separator"); // Получение каталога файла
    try {
      Scanner fIN = new Scanner(new FileInputStream(FileName));
      // Читаем с файла данные
      for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 6; j++) {
          mass[i][j] = fIN.nextInt();
      }
    }
      // Вывод считанного массива
      jTextArea_Out.setText("Исходные данные из файла:\n");
      for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 6; j++) {
          jTextArea_Out.append(String.format("%5d", mass[i][j]));
      } jTextArea_Out.append("\n");
    }
  } catch (Exception ex) {
      jTextArea_Out.setText("Error read file!");
  }
}

private void jButton_TaskActionPerformed(java.awt.event.ActionEvent
evt) {
```

```
    // Запись обработанного массива в выходной файл
    try {
      PrintWriter fOUT = new PrintWriter(new OutputStreamWriter(new
FileOutputStream(DirName + "output.txt")));
      // Вывод результирующего массива в файл
      for (int i = 0; i < 5; i++) { fOUT.println("");
        for (int j = 0; j < 6; j++) {
          fOUT.print(String.format("%5d", mass[i][j]));
      }
    }
      // Сохранение и закрытие файла
      fOUT.flush(); fOUT.close();
      // Вывод результирующего массива на экран
      jTextArea_Out.append("\nОбработанные данные:\n");
      for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 6; j++) {
          jTextArea_Out.append(String.format("%5d", mass[i][j]));
      } jTextArea_Out.append("\n");
    }
  } catch (Exception ex) {
      jTextArea_Out.setText("Error read file!");
  }
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Обработка данных
    int s = 0; // Считаем сколько едениц во втором столбце
    for (int i = 0; i < 5; i++) { if (mass[i][1] == 1) { s++; } }
    if (s == 5) { // Если их 5 тогда ищем макс и мин элемент и меняем
их на 100 и 5
      int min = mass[0][0], minI = 0, minJ = 0, max = mass[0][0], maxI =
0, maxJ = 0;
      for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 6; j++) {
          if (mass[i][j] > max) { max = mass[i][j]; maxI = i; maxJ = j; }
          if (mass[i][j] < min) { min = mass[i][j]; minI = i; minJ = j; }
        }
    } mass[maxI][maxJ] = 100;  mass[minI][minJ] = 5;
  }
}
...
```

Работа с файлами параметров. Часто в приложениях возникает необходимость сохранять настройки в файл и считывать их оттуда. Когда приложение запускается – оно считывает настройки из файла с параметрами, а когда закрывается – записывает нужные параметры в файл.

В приложениях на языке Java параметры сохраняются в так называемых «файлах свойств (параметров)».

Файлы параметров – это обычные текстовые файлы с расширением «.properties» или «.conf», которые можно редактировать и просматривать при помощи любого текстового редактора.

Файл параметров представляет собой пары "ключ=значение", где ключ – это имя какого-то параметра. Файлы «.properties» имеют следующий формат:

```
# Начало файла ".properties".
# Это комментарий
! Восклицательный знак тоже можно отметить как комментарий.
website = http://mail.ru/
language = Russian
# Обратная косая черта указывает на продолжение значения параметра на следующей строке.
message = Добро пожаловать в \
      мир Java!
# Добавление пробелов в ключе
key\ with\ spaces = Это значение должно искаться ключом "key with spaces".
# Юникод
tab = \u0009
```

В приведенном выше примере, «website» является ключем, а «http://mail.ru/» – значением.

В то время как знак номера (#) и восклицательный знак (!) отмечают текст как комментарии, это не имеет никакого эффекта, если это – значение параметра. Таким образом, значение ключа «message» будет «Добро пожаловать в мир Java!», а не «Добро пожаловать в мир Java». Нужно заметить также, что все пробелы перед «мир Java!» полностью исключены.

Работа с файлами свойств в Java осуществляется с помощью специального класса – Properties. В этом классе есть методы, которые позволяют считать параметр и задать его – это методы getProperty() и setProperty().

Пример работы с файлом параметров в консольном приложении:

```
package tsn01.prop;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Date;
import java.util.Properties;

class Props {    // Работа с файлами настроек
  // Получение каталога расположения файла параметров (место
запуска программы)
  String dir = new
File(".").getAbsoluteFile().getParentFile().getAbsolutePath()
      + System.getProperty("file.separator");
  // Получение каталога расположения файла параметров (рабочий
каталог пользователя)
  //String dir = System.getProperty("user.home") +
System.getProperty("file.separator");
  String FileName = dir + "tsn_demo.prop"; // Имя файла с настройками
  // Объект для хранения параметров со значениями (имена
параметров чувствительны к регистру)
  Properties p = new Properties();
  { System.out.println("Каталог с файлом параметров: " + dir); }

  public void WriteProps() { // Запись файла настроек
    try {
      File f = new File(FileName); // Создаем объект доступа к файлу
параметров
      if (f.exists() == false) { f.createNewFile(); } // Если нет файла, то
создаем его
        else p.load(new FileInputStream(FileName)); // Если есть, то
загрузить файл с параметрами
      p.put("date", new Date().toString()); // Записать (обновить)
параметр "date"
      p.put("name", "Сергей"); // Записать (обновить) параметр
"name"
      p.put("name.pass", "pass123"); // Записать (обновить) параметр
"name.pass"
      // Сохранить файл с настройками
```

```
            p.store(new FileOutputStream(FileName), "/* properties updated
*/");
        } catch (Exception e) { // Если ошибка
            System.out.println(e); // Вывести сообщения об ошибке
        }
    }

    public void ReadProps() { // Чтение файла настроек
        try {
            p.load(new FileInputStream(FileName)); // Загрузить файл с
параметрами
            System.out.println("date = " + p.getProperty("date")); // Считать
параметр "date"
            System.out.println("name = " + p.getProperty("name")); // Считать
параметр "name"
            System.out.println("name.pass = " + p.getProperty("name.pass")); //
Считать параметр "name.pass"
        } catch (Exception e) { // Если ошибка
            System.out.println(e); // Вывести сообщения об ошибке
        }
    }
}

public class App1 {
    public static void main(String[] args) {
        System.out.println("Информация о системе:");
        System.out.println("Версия Java: " +
System.getProperty("java.version"));
        System.out.println("Каталог, в который установлена система Java:
" + System.getProperty("java.home"));
        System.out.println("Версия OS: " + System.getProperty("os.name"));
        System.out.println("Архитектура ОС: " +
System.getProperty("os.arch"));
        // Работа с файлом параметров
        Props props = new Props(); // Создание объекта работы с файлом
параметров
        props.WriteProps(); // Записать параметры в файл
        props.ReadProps(); // Считать параметры из файла
    }
}
```

На экран будет выведено следующее:

```
Информация о системе:
Версия Java: 1.7.0_25
Каталог, в который установлена система Java: C:\Program Files\Java\jdk1.7.0_25\jre
Версия OS: Windows 7
Архитектура ОС: x86
Каталог с файлом параметров: D:\NetBeansProjects\1\TSN01_Prop\
date = Sat Oct 05 23:39:58 ALMT 2013
name = Сергей
name.pass = pass123
```

В результате работы программы получится следующий файл с параметрами «tsn_demo.prop»:

```
#/* properties updated */
#Sat Oct 05 20:28:50 ALMT 2013
name=\u0421\u0435\u0440\u0433\u0435\u0439
date=Sat Oct 05 20\:28\:50 ALMT 2013
name.pass=pass123
```

Работа с xml-файлами. Язык XML (eXtensible Markup Language) – это универсальный язык разметки, который является современным стандартом для работы с данными. Файлы XML – это текстовые файлы с данными, записанными в формате языка XML.

В файлах XML находится описание структуры хранимых данных, и сами данные. Файлы XML используются для хранения структурированных данных и для обмена информацией между программами.

Язык XML имеет следующие достоинства:

- международный промышленный стандарт;
- кросс платформенный формат;
- поддерживает Юникод (символы всех языков);
- человеко-ориентированный формат документа.

Пример XML-документа:

```
<?xml version="1.0" encoding="utf-8"?>
<pricelist>
  <book id="1">
    <title>Книга 1</title>
```

```
  <author>Автор 1</author>
  <price>Цена 1</price>
 </book>
 <book id="2">
  <title>Книга 2</title>
  <author>Автор 2</author>
  <price>Цена 2</price>
 </book>
 <book id="3">
  <title>Книга 3</title>
  <author>Автор 3</author>
  <price>Цена 3</price>
 </book>
</pricelist>
```

Пример работы с xml-файлом в консольном приложении:

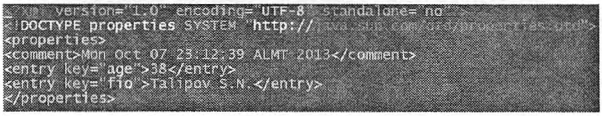
```
package tsn01.prop;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Date;
import java.util.Properties;

public class App1 {
   // Каталог запуска программы
String dir = new
File(".").getAbsoluteFile().getParentFile().getAbsolutePath()
         + System.getProperty("file.separator");
   String FileName = dir + "tsn_demo.xml"; // Имя файла с настройками
   File f = new File(FileName); // Создаем объект доступа к файлу
параметров
   // Объект для хранения параметров со значениями (имена
параметров чувствительны к регистру)
   Properties p = new Properties();

   void writeXML() { // Сохранение (обновление) данных в xml-файле
      try {
         if (f.exists() == false) { f.createNewFile(); } // Если нет XML-
файла, то создаем его
```

```
            else { p.loadFromXML(new FileInputStream(FileName)); } //
Считываем значения из XML-файла, если он есть
        p.setProperty("fio", "Talipov S.N."); // Установка параметра "fio"
        p.setProperty("age", "38"); // Установка параметра "age"
        p.storeToXML(new FileOutputStream(FileName), new
Date().toString()); // Сохраняем XML-файл
    } catch (Exception e) { // Если ошибка
        System.err.println("Error store to XML!"); // Вывести сообщения
об ошибке
    }
}

    void readXML() { // Считывание данных из xml-файла
        try {
            p.loadFromXML(new FileInputStream(FileName)); // Загружаем
значения из XML-файлв
            System.out.println(p.getProperty("fio", "???")); // Считываем и
выводим на экран параметр "fio"
            System.out.println(p.getProperty("age", "0")); // Считываем и
выводим на экран параметр "age"
    } catch (Exception e) { // Если ошибка
        System.err.println("Error load from XML!"); // Вывести
сообщения об ошибке
    }
}

public static void main(String[] args) {
    // РАБОТА С XML-ФАЙЛОМ
    App1 MyApp = new App1(); // Создание объекта работы с файлом
параметров
    MyApp.writeXML(); // Записать параметры в XML-файл
    MyApp.readXML(); // Считать параметры из XML-файла
  }
}
```

На экран будет выведено следующее:

```
Talipov S.N.
38
```

В результате работы программы получится следующий файл с параметрами «tsn_demo.xml»:



Пример загрузки xml-файла с jar-ресурса:

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    // Загрузка xml-файла из jar-ресурсов
    try { // Получаем имя xml-файла из jar-ресурсов
    InputStream XMLstream =
getClass().getResourceAsStream("tsn_demo.xml");
    Properties XML = new Properties(); // Создание объекта работы с
параметрами
        XML.loadFromXML(XMLstream); // Загрузка файла из ресурса
        // Вывод значения параметра "fio"
        JOptionPane.showMessageDialog(rootPane,
XML.getProperty("fio", "???"), "ФИО",
 JOptionPane.INFORMATION_MESSAGE);
        XMLstream.close(); // Закрываем доступ к XML-файлу
  } catch (Exception ex) { JOptionPane.showMessageDialog(rootPane,
"Ошибка чтения XML-файла", "Ошибка ввода",
JOptionPane.ERROR_MESSAGE);
  }
 }
```

3.12 Работа с мультимедиа, графикой, треем.

Рассмотрим простейший пример проигрывания звукового wav-файла из jar-ресурса:

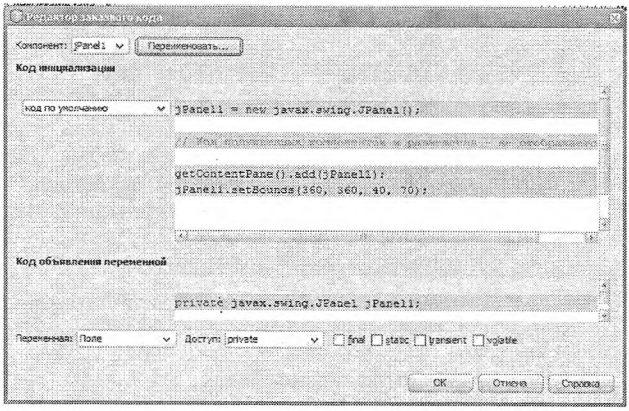
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Издаем звук файла "snd.wav" из jar-ресурса программы
    try {
      Clip c = AudioSystem.getClip(); // Подключение к звуковой
системе
      URL url = this.getClass().getResource("snd.wav"); // Находим
звук в jar-ресурсе
      // Подключение к звуковому файлу
      AudioInputStream ais = AudioSystem.getAudioInputStream(url);
      c.open(ais); // Проиграть звук
      c.loop(0); // Проиграть звук один раз, без повторов
```

```
            // Если необходимо гарантированное время для проигрывания,
то нужен следующий код:
//              try { Thread.sleep(500); // Время ожидания программы пока
играет звук, мс
//          } catch (InterruptedException ex) {
//          }
    } catch (LineUnavailableException | UnsupportedAudioFileException |
IOException ex) {
            ex.printStackTrace();
    }
}
```

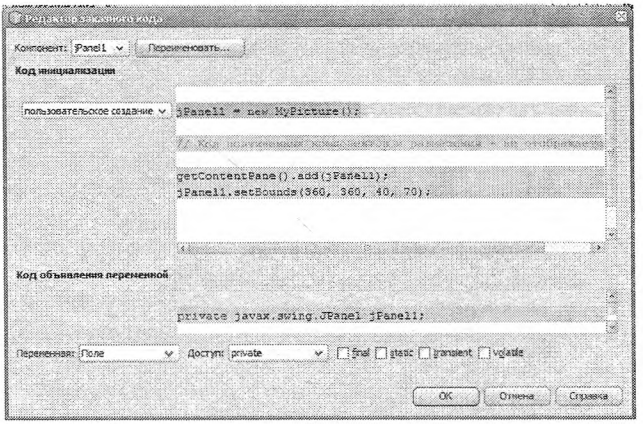
Работа с графикой. В Java имеется возможность наносить собственный рисунок на визуальные элементы программы, в частности на панели JPanel. Это позволяет создавать визуальные графики, делать графическую анимацию и многое другое.

Для того чтобы панель JPanel позволила наносить на нее рисунок, она должна быть создана не из ее родного класса javax.swing.JPanel, а через новый дочерний класс, который является потомком для javax.swing.JPanel и имеет переопределенный метод рисования «paint».

Для того чтобы сделать обычную панель JPanel на форме в пригодную для рисования, нужно правой кнопкой мыши вызвать на ней контекстное меню и выбрать пункт «Настроить код»:



В редакторе заказного кода необходимо в коде инициализации изменить значение «код по умолчанию» на значение «пользовательское создание» и изменить строчку «jPanel1 = new javax.swing.JPanel();» на значение «jPanel1 = new MyPicture();»:



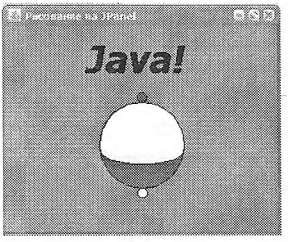
Мы указали, что панель формы jPanel1 будет создана не как обычно, а на основе нашего собственного класса, который мы назвали «MyPicture». Класс «MyPicture» нужно вписать в код программы, как например, показано далее в примере.

Пример рисования на панели JPanel с заливкой и буфером изображения:

```
public class NewJFrame extends javax.swing.JFrame {
 ...

  public class MyPicture extends JPanel { // Класс рисования своей картинки
      Graphics2D canvas;   // Класс рисования
      BufferedImage buff;  // Буферное изображение
      int x = 400; // Константа размера полотна по x
      int y = 250; // Константа размера полотна по y

      MyPicture() {
        // Создаем буферное полотно для рисования размером x-y
        buff = new BufferedImage(x, y, BufferedImage.TYPE_INT_RGB);
```

```
        // Создаем двустороннюю связь между буферным
изображением и классом рисования
        canvas = (Graphics2D) buff.getGraphics();
        canvas.setPaint(Color.GRAY); // Устанавливаем цвет рисования
серым
        canvas.fillRect(0, 0, x, y); // Заливаем полотно для рисования
        canvas.setPaint(Color.BLACK);        // Устанавливаем цвет
рисования черным
        canvas.drawOval(100, 80, 90, 90);  // Отрисовываем большой
овал
        canvas.drawArc(100, 115, 90, 30, 180, 180); // Отрисовываем
дугу по середине овала
        //рисуем два мелньких круга сверху и снизу
        canvas.drawOval(140, 70, 10, 10);
        canvas.drawOval(140, 170, 10, 10);
        //устанавливаем стиль и пишим текст
        canvas.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC,
40));
        canvas.drawString("Java!", 85, 50);
        try {
          // При помощи созданной функции заливки закрашиваем две
части главного овала
          // и два маленьких кружка
          fill(110, 110, Color.GRAY, Color.WHITE);
          fill(140, 150, Color.GRAY, Color.RED);
          fill(145, 75, Color.GRAY, Color.RED);
          fill(145, 175, Color.GRAY, Color.WHITE);
    } catch (Exception ex) {
    }
  }

    @Override
    public void paintComponent(Graphics g) {
      super.paintComponent(g); // Отрисовываем панель и компоненты
на ней
      g.drawImage(buff, 0, 0, this); // Отрисовываем буфер с нашим
изображением на панель
  }

  private void fill(int x, int y, Color bgcolor, Color color) throws
Exception {
```

```
        // Заливка фигур методом ромба, параметры: x,y- координаты начала заливки,
        // bgcolor-цвет который надо закрашивать, color-цвет которым надо закрашивать
        ArrayList<Point> point = new ArrayList<>(); // Создаем динамический массив точек
        point.add(new Point(x, y)); // Добавляем начальную точку в массив
        Color oldColor = canvas.getColor(); // Сохраняем старый цвет рисования
        canvas.setPaint(color); //ставим цвет закраски
        while (point.size() > 0) { // Пока в массиве имеются точки для закрашивания
            Point p = point.remove(0); // Считываем координаты первой точки, и удаляем ее из массива
            x = p.x; y = p.y;
            if (bgcolor.getRGB() == buff.getRGB(x, y)) { // Если ее надо нам закрасить
                canvas.drawLine(x, y, x, y); // Закрашиваем точку
                point.add(new Point(x + 1, y)); // Добавляем точку справа
                point.add(new Point(x - 1, y)); // Добавляем точку слева
                point.add(new Point(x, y + 1)); // Добавляем точку снизу
                point.add(new Point(x, y - 1)); // Добавляем точку сверху
            }
        }
        canvas.setPaint(oldColor); //ставим старый цвет рисования
        repaint(); // Перерисовываем изображение
    }
}

...

    public NewJFrame () {
        initComponents();
}
```



Пример простого рисования на панели JPanel:

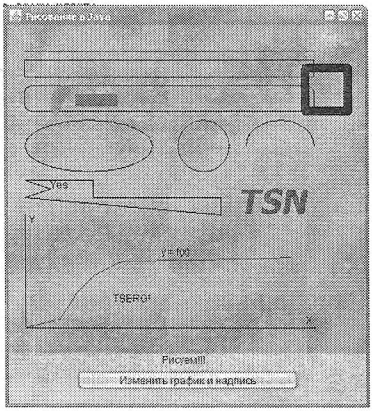
```
public class NewJFrame extends javax.swing.JFrame {
    ...

    // Делаем общедоступные переменные-массивы для использования
в панели рисования и для всех других компонент программы
    char[] mas = {'T', 'S', 'E', 'R', 'G', '!'};
    int[] xArray = {20, 40, 60, 80, 100, 120, 130, 140, 280, 332};
    int[] yArray = {350, 345, 340, 310, 290, 280, 275, 273, 271, 269};

    public class MyPicture extends JPanel { // Класс рисования своей
картинки

        @Override
        public void paintComponent(Graphics g) { // Переменная «g»
служит для доступа к классическому Java-рисованию
            super.paintComponent(g); // // Отрисовываем панель и
компоненты на ней
            Graphics2D g2 = (Graphics2D) g; // Получаем переменную «g2
доступа к Java2D-рисованию
            jLabel1.setText("Рисуем!!!"); // Выводим в метку текст (доступ
к внешнему компоненту)
            this.setBackground(new Color(153,255,153)); // Установка цвета
фона панели
            this.setForeground(Color.darkGray); // Установка цвета
рисования на панели

            int w = getWidth(); int h = getHeight(); // Узнать ширину и
высоту области рисования
            // Создаем объект-градиент
```

```
        GradientPaint gradient = new GradientPaint(0, 0,Color.orange, w,
h, Color.green, true);
        g2.setPaint(gradient); // Установить вид закраски
        g2.fillRect(0, 0, w, h); // Нарисовать закрашенный градиентной
заливкой четырехугольник
        g.drawLine(110, 120, 110, 120); // Рисование точки
        g.drawLine(20, 20, 360, 20); // Рисование линии
        Color oldColor = g.getColor(); // Считать текущий цвет
"foreground"
        Color newColor = new Color(0, 0, 255); // Создать переменную с
цветом
        g.setColor(newColor); // Установка нового цвета для текущего
цвета
        g.drawLine(20, 30, 360, 30); // Рисование линии
        g.setColor(oldColor); // Установка старого текущего цвета
        g.drawRect(20, 40, 340, 20); // Рисование не закрашенного
четырехугольника
        newColor = new Color(100, 15, 255); // Создать переменную с
цветом
        g.setColor(newColor); // Установка нового текущего цвета
        g.fillRect(80, 80, 50, 15); // Рисование закрашенного
четырехугольника
        g.setColor(oldColor); // Установка старого текущего цвета
        // Рисование не закрашенного округлого четырехугольника
        g.drawRoundRect(20, 70, 340, 30, 20, 15);
        g.drawOval(20, 110, 150, 60); // Нарисовать овал
        g.drawOval(200, 110, 60, 60); // Нарисовать круг
        g.drawArc(280, 110, 80, 60, 0, 180); // Нарисовать дугу
        int[] arrayX = {20, 100, 100, 250, 250, 20, 20, 50}; // Координаты
x полигона
        int[] arrayY = {180, 180, 200, 200, 220, 200, 200, 190}; //
Координаты y полигона
        Polygon poly = new Polygon(arrayX, arrayY, 8); // Объект-
полигон
        g.drawPolygon(poly); // Нарисовать объект-полигон
        Point aPoint = new Point(50, 190); // Получить объект-точку
        if (poly.contains(aPoint)) { // Если точка имеется в полигоне, то
          g.drawString("Yes", 50, 190); // Нарисовать текст текущим
цветом "Yes"
    }
        g.setColor(new Color(10, 200, 55)); // Установить новый цвет
```

```
        g.fillArc(50, 70, 30, 40, 50, 190); // Нарисовать закрашенный
сектор
        newColor = new Color(10, 10, 155); // Создать переменную с
цветом
        g.setColor(newColor); // Установка нового текущего цвета
        Font font = new Font("Tahoma", Font.BOLD | Font.ITALIC, 40); //
Задать объект-шрифт
        Font oldFont = g.getFont(); // Считать текущий шрифт (фонт)
        g.setFont(font); // Установить собственный тип шрифта
        g.drawString("TSN", 270, 220); // Нарисовать текст "TSN"
        g.setFont(oldFont); // Установить старый (сохраненный) тип
шрифта
        g.setColor(oldColor); // Установка нового текущего цвета
        // Нарисовать оси графика
        g.drawLine(20, 220, 20, 350); g.drawLine(20, 350, 360, 350);
        g.drawString("Y", 25, 230);   g.drawString("X", 350, 346);
        // Нарисовать график
        g.setColor(newColor); // Установить собственный тип шрифта
        g.drawPolyline(xArray, yArray, 10); // Нарисовать объект-
полигон
        g.setColor(oldColor); // Установка нового текущего цвета
        g.drawString("y = f(x)", 180, 267); // Нарисовать текст "y = f(x)"
        // Рисование символов на экране
        g2.drawChars(mas, 0, 6, 123, 321);
        // Использование перьев в Graphics2D
        BasicStroke penl = new BasicStroke(10,
BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND, 10);
        g2.setStroke(penl);
        g2.draw(new Rectangle2D.Double(350, 50, 50, 50));
    }
}
    ...
   public NewJFrame() {
      initComponents();

    ...
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // Изменяем данные картинки
      mas[0] = 'x'; // Меняем символ «T» в надписи «TSERG!»,
получится «xSERG!»
```

```
    xArray[0] = 10; yArray[0] = 200; // Меняем координаты для
графика рисунка
    this.repaint(); // Перерисовать все компоненты формы
}
...
```

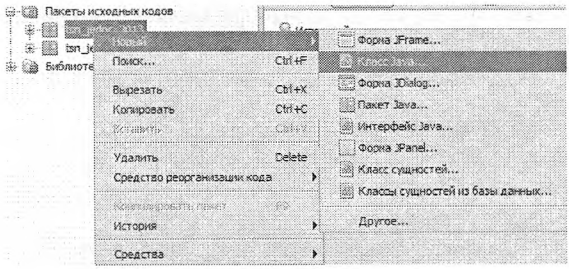
Вот что получится в результате выполнения программы:

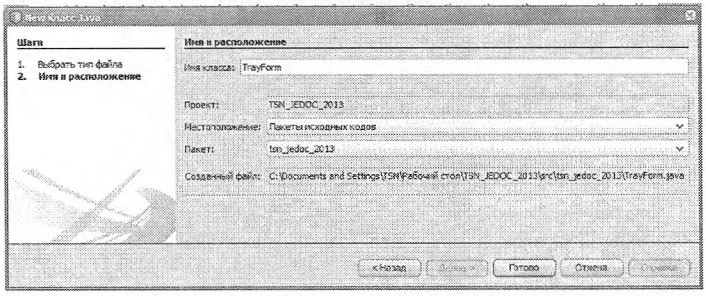


Работа с треем. Для того чтобы программа могла сворачиваться в системный трей, необходимо модифицировать главную форму программы.

Наиболее простой способ состоит в создании в программе нового дополнительного класса работы с треем, и привязке к нему основной формы программы:

Для этого заменяем к коде у формы строку «public class NewJFrame extends javax.swing.JFrame {» на значение «public class NewJFrame extends TrayForm {», после чего добавляем в программу (создаем) новый класс «TrayForm»:





После этого вписываем следующий код в созданный класс «TrayForm»:

```
package tsn_jedoc_2013;
import java.awt.AWTException;
import java.awt.Image;
import java.awt.MenuItem;
import java.awt.PopupMenu;
import java.awt.SystemTray;
import java.awt.Toolkit;
import java.awt.TrayIcon;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowStateListener;
import java.net.URL;
import javax.swing.JFrame;

public class TrayForm extends JFrame {
private SystemTray systemTray = SystemTray.getSystemTray();
```

```
private TrayIcon trayIcon;

  public  TrayForm() {
    super();
    URL resource = getClass().getResource("icon.png"); // Имя картинки
для трея (из JAR-ресурса)
    Image image = Toolkit.getDefaultToolkit().getImage(resource);
    // Надпись в трее при наведении курсора на значок программы
    trayIcon = new TrayIcon(image, "Программирование на Java
(Swing)");
    trayIcon.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) { // Восстановление
программы из иконки
        setVisible(true); setState(JFrame.NORMAL); removeTrayIcon();
    } });
    addWindowStateListener(new WindowStateListener() {
      public void windowStateChanged(WindowEvent e) { //
Сворачивание программы в иконку
        if (e.getNewState() == JFrame.ICONIFIED) {
          setVisible(false); addTrayIcon();
     } } });
    PopupMenu popupMenu = new PopupMenu(); // Создание меню для
трея
    MenuItem item1 = new MenuItem("Развернуть программу");
    MenuItem item2 = new MenuItem("Выход");
    item1.addActionListener(new ActionListener() { // Команды
контекстного меню трея
      public void actionPerformed(ActionEvent e) { // Восстановление
программы из иконки
        setVisible(true); setState(JFrame.NORMAL); removeTrayIcon();
    } });
    item2.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) { // Выход из
программы
        dispose(); System.exit(0);
    } });

    // Установка меню для трея
    popupMenu.add(item1); popupMenu.add(item2);
trayIcon.setPopupMenu(popupMenu);
}
```

```
private void removeTrayIcon() { systemTray.remove(trayIcon); }
private void addTrayIcon() {
      try {
         systemTray.add(trayIcon);
         trayIcon.displayMessage("Программирование на Java (Swing)",
              "Приложение свернуто. Двойной клик для восстановления
окна", TrayIcon.MessageType.INFO);
   } catch (AWTException ex) {
         ex.printStackTrace();
   }
  }
}
```

3.13 Работа с таймерами, потоками, реестром

Таймеры. Класс javax.swing.Timer является таймером подсистемы Swing, используется в программах с графическим интерфейсом для периодического выполнения задачи (набора команд) с заранее заданным интервалом.

Класс java.util.Timer является таймером общего назначения, используется в программах без графического интерфейса или при необходимости долгой обработки данных, а также для выполнения фоновых задач по расписанию в отдельном потоке.

У таймеров Swing имеется событие «actionPerformed», которое вызывается при срабатывании таймера, а также методы:

- setRepeats(false) – установка режима единичного срабатывания таймера;
- start – запуск таймера;
- stop – остановка таймера.

Таймеры общего назначения используют класс задания «java.util.TimerTask», в методе «run» которого реализуется код таймера. Запускаются таймеры общего назначения методом «schedule».

Пример работы с Swing-таймерами:

```
public class Form1 extends javax.swing.JFrame {
...

  // Текст таймера прописывается в данном месте кода формы
вручную!
  // Таймер будет вызываться каждую миллисекунду
```

```
  javax.swing.Timer jTimer1 = new javax.swing.Timer(1, new
ActionListener() {
    public void actionPerformed(ActionEvent evt) { // Метод таймера 1
      Point p = jLabel1.getLocation();
      if (p.x <= 300) { ++p.x; } else { jTimer1.stop();jTimer2.start(); }
      jLabel1.setLocation(p);
  }
});

  // Текст таймера прописывается в данном месте кода формы
вручную!
  // Таймер будет вызываться каждую миллисекунду
  javax.swing.Timer jTimer2 = new javax.swing.Timer(1, new
ActionListener() {
    public void actionPerformed(ActionEvent evt) { // Метод таймера 2
      Point p = jLabel1.getLocation();
      if (p.x >= 30) { --p.x; } else { jTimer2.stop();jTimer1.start(); }
      jLabel1.setLocation(p);
  }
});

...
  public Form1() {
    initComponents();
...
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Запуск таймера
    jTimer1.start();
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Остановка таймеров
    jTimer1.stop(); jTimer2.stop();
}
```

Пример работы с таймером общего назначения:

```
  public Form1() { // ---------- Вписывается в конструктор формы
после вызова «initComponents()»
    initComponents();
...
```

```
    // Текст таймера прописывается в данном месте кода формы вручную!
    // Таймер запускается с задержкой «0» миллисекунд, и будет вызываться раз в секунду (1000мс)
    java.util.Timer uTimer1 = new java.util.Timer();
    uTimer1.schedule(new java.util.TimerTask() {
      public void run() {
        jLabel3.setText(new java.text.SimpleDateFormat(
          "dd-MMM-yy
hh:mm:ss").format(java.util.Calendar.getInstance().getTime()));
    } }, 0, 1000);
} // ---------- Конец конструктора формы
```

Потоки. Практически все операционные системы поддерживают концепцию процессов – независимо работающих программ, до некоторой степени изолированных друг от друга. Потоками называются средства, позволяющие нескольким операциям сосуществовать в рамках одного процесса. Потоки представляют собой независимые параллельные пути исполнения программ.

Потоки используют в программах для того, чтобы:

- повысить отзывчивость интерфейса пользователя;
- использовать преимущества многопроцессорных систем;
- упростить моделирование;
- выполнять асинхронную или фоновую обработку данных.

Пример работы с потоком в программе с графическим интерфейсом:

```
...

public class Form1 extends javax.swing.JFrame {
  // Текст потока прописывается в данном месте кода формы вручную
  class TThread1 extends Thread { // Пример потока

    public void run() { // Главный метод потока
      try { while (true) { // Выполняем поток постоянно до прерывания
          jLabel2.setText(String.valueOf(Math.random()));
          if (Math.random()>0.999) break; // Случайное само прерывание потока
          sleep(1); // Задержка потока для анализа прерывания
      } } catch (InterruptedException e) { // Прерывание потока
```

```
                jLabel2.setText("Поток принудительно остановлен!!!");
                return; // Выход из потока
            } jLabel2.setText("Поток остановился");
    }

} Thread tThread1; // Переменная для доступа к потоку

...

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // Создание и запуск потока
    tThread1 = new TThread1(); tThread1.start();
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // Принудительная остановка потока
    tThread1.interrupt();
}
```

Работа с системным реестром. Системный реестр – это иерархическая централизованная база данных в операционных системах.

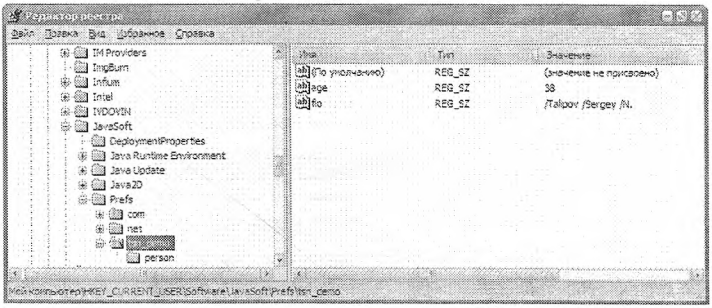
В реестре хранятся данные, необходимы для правильного функционирования ОС: профили всех пользователей, сведения об установленном программном обеспечении и типах документов, которые могут быть созданы каждой программой, информация о свойствах папок и значках приложений, а также установленном оборудовании и используемых портах.

Пример работы с системным реестром в консольном приложении:

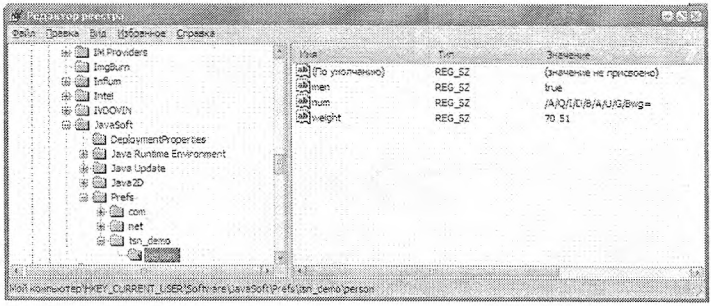
```
package tsn01.pref;
import java.util.prefs.Preferences;
import java.util.Arrays;

public class App1 {
   public static void main(String[] args) {
      // Работа с реестром windows
      // Создается (открывается ) раздел в реестре:
      // HKEY_CURRENT_USER\Software\JavaSoft\Prefs\tsn_demo и
вложенный узел "person"
```

```
        // Для Linux: $HOME/.java/.userPrefs/.tsn_demo
        Preferences node = Preferences.userRoot().node("tsn_demo");
        Preferences node2 =
Preferences.userRoot().node("tsn_demo").node("person");
        // Создается раздел в реестре:
HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Prefs\tsn_demo
        // и вложенный узел «"person"»
        // Preferences node = Preferences.systemRoot().node("tsn_demo");
        // Preferences node2 =
Preferences.systemRoot().node("tsn_demo").node("person");
        byte k[] = new byte[]{1, 2, 3, 4, 5, 6, 7, 8}; // Массив байт
        // Запись данных в реестр
        node.put("fio", "Talipov Sergey N."); // Записать (обновить)
строковый параметр "fio"
        node.putInt("age", 38); // Записать (обновить) целый параметр
"age"
        node2.putFloat("weight", 70.51f);  // Записать (обновить)
вещественный параметр "weight"
        node2.putBoolean("men", true);  // Записать (обновить) логический
параметр "men"
        node2.putByteArray("num", k);  // Записать (обновить) массив байт
"num"
        // Считывание данных из реестра
        String s = node.get("fio", "???"); // Считать строковый параметр
"fio"
        int n = node.getInt("age", 0); // Считать целый параметр "age"
        float f = node2.getFloat("weight", 0); // Считать вещественный
параметр "weight"
        boolean b = node2.getBoolean("men", true); // Считать логический
параметр "men"
        byte[] kk = node2.getByteArray("num", null); // Считать массив байт
"num"
        String s2 = Arrays.toString(kk); // Преобразовать массив байт в
строку
        // Вывод данных на экран
        System.out.println(s + " - " + n); // Выведется на экран "Talipov
Sergey N. - 38"
        System.out.println(f + " - " + b); // Выведется на экран "70.51 - true"
        System.out.println(s2); // Выведется на экран "[1, 2, 3, 4, 5, 6, 7, 8]"
    }
}
```

Записанные значения в реестре «HKEY_CURRENT_USER\Software\JavaSoft\Prefs\tsn_demo»:



Записанные значения в реестре «HKEY_CURRENT_USER\Software\JavaSoft\Prefs\tsn_demo\person»:



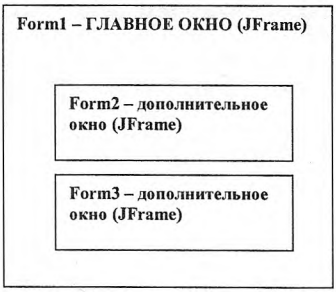
3.14 Многооконные программы

Постановка задачи. Необходимо сделать программу из трех окон с переходом от одного к другому. В первом и втором окне реализовать запрос данных из списков, в третьем окне выдать выбранные пользователем значения.

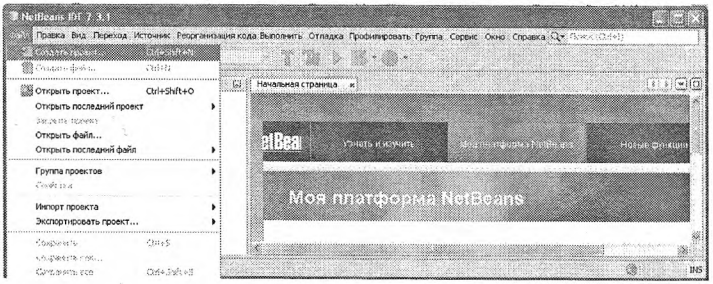
Принцип реализации. Для реализации многооконных программ имеются разные подходы. Рассмотрим наиболее простой и удобный вариант, предоставляющий системой NetBeans.

Основу программы будет составлять обычное окно JFrame. Это окно создастся при создании простейшей программы, рассмотренной ранее в теме «NetBeans. Простейшая программа». Назовем его Frame1.

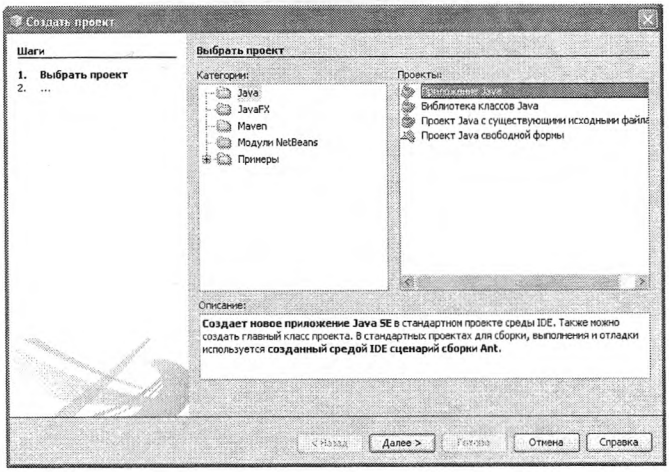
В основное окно Frame1 будут помещены дополнительные два окна, тоже типа JFrame. Эти дополнительные окна Frame1 и Frame2 будут добавлены в основное окно через компонент «Фрейм» из палитры «Диалоговые окно Swing». Получится следующая структура программы:



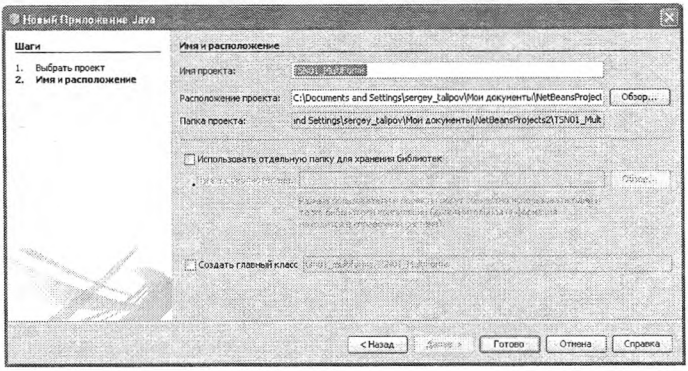
Создание проекта многооконной программы. Выбираем «Файл» – «Создать проект»:



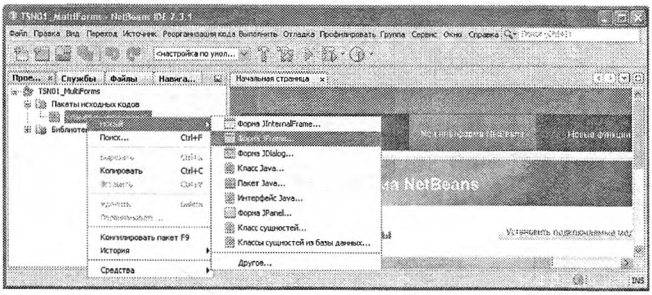
Указываем тип проекта «Java» – «Приложение Java»:



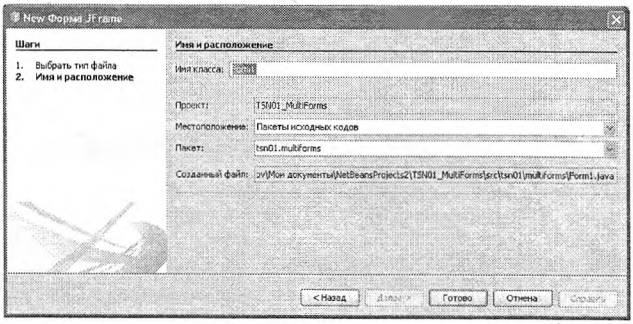
Задаем имя проекту и указываем, что главный класс создавать ненужно:



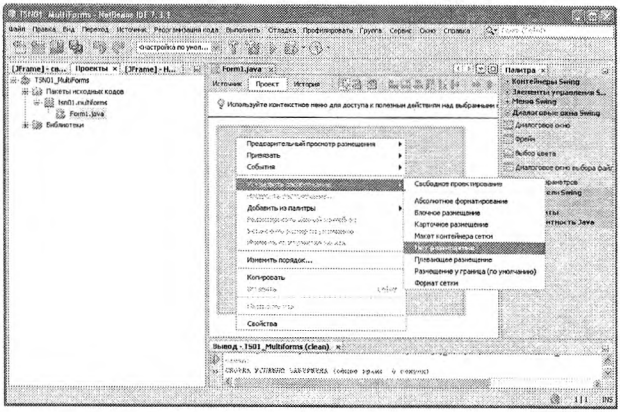
Создание первой формы. Добавляем в проект первую (основную) форму:



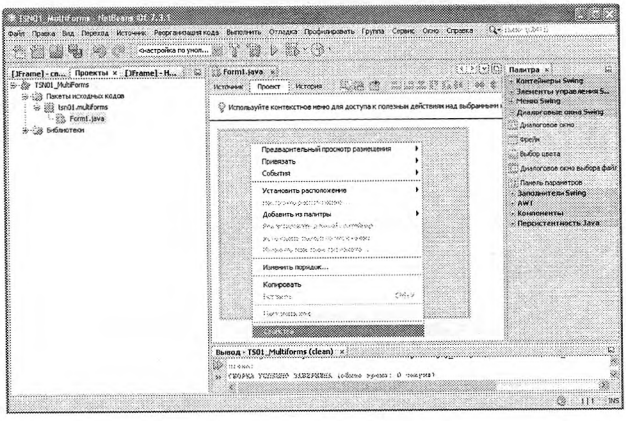
Задаем форме имя класса и пакет:

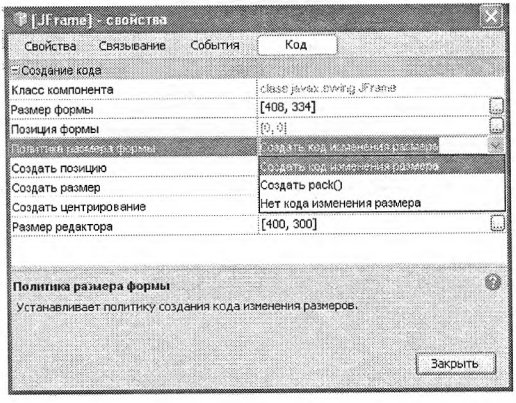


Настраиваем форму на свободное размещение компонент:



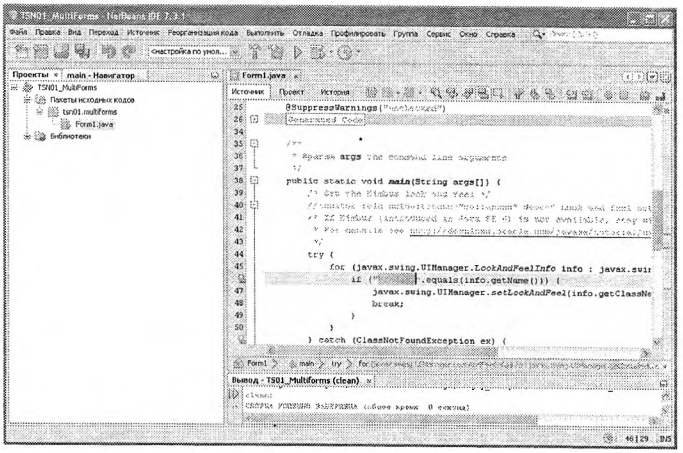
Задаем окну наличие размера и центрирование на экране:



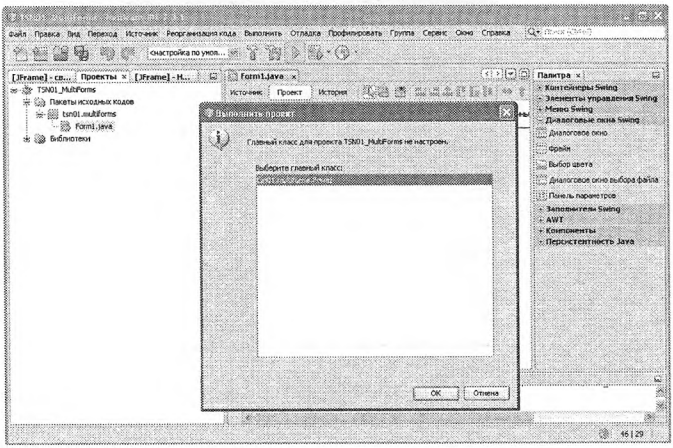


Запрещаем главному окну изменять размеры (убираем галочку со свойства «resizable»):

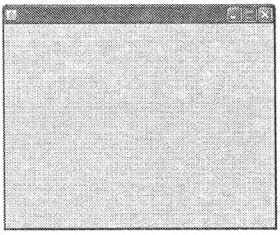
Задаем окну стиль оформления «Windows»:



Запускаем программу для указания ее главного класса:



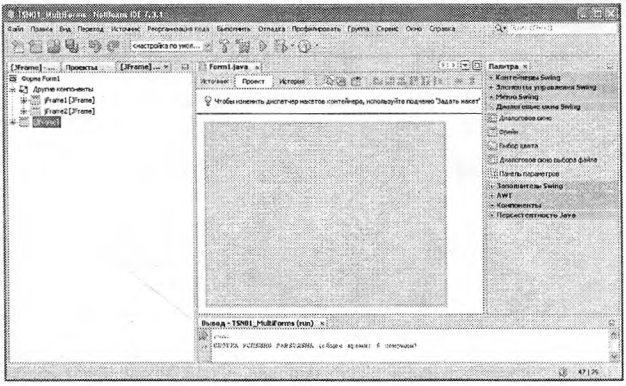
Закрываем запущенную программу:



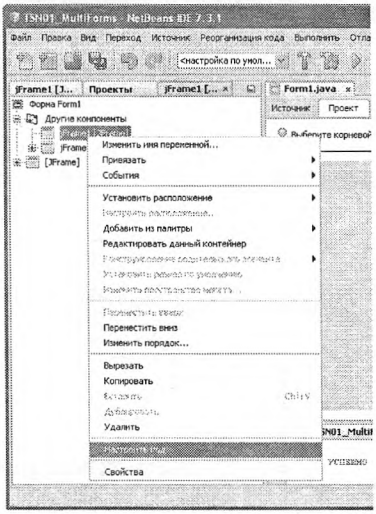
Добавление в проект двух дополнительных окон. Добавляем на форму Form1 дополнительно два окна из палитры «Диалоговые окна Swing» – «Фрейм».

Перетаскиваем компонент «Фрейм» на главную форму – в программу добавляется второе окно.

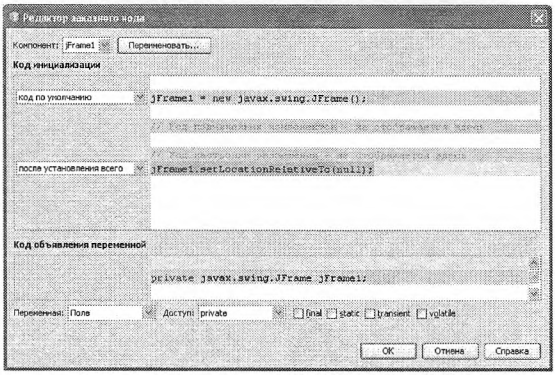
Перетаскиваем еще один компонент «Фрейм» на главную форму – в программу добавляется третье окно.



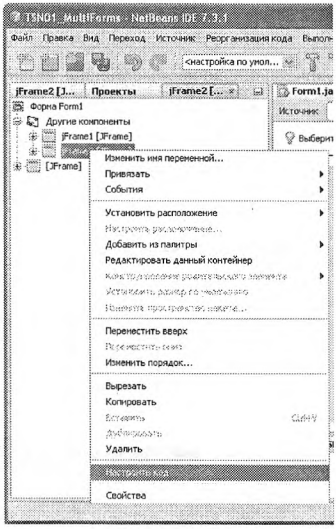
Выбираем второе окно jFrame1 и настраиваем его для расположения по центру экрана:



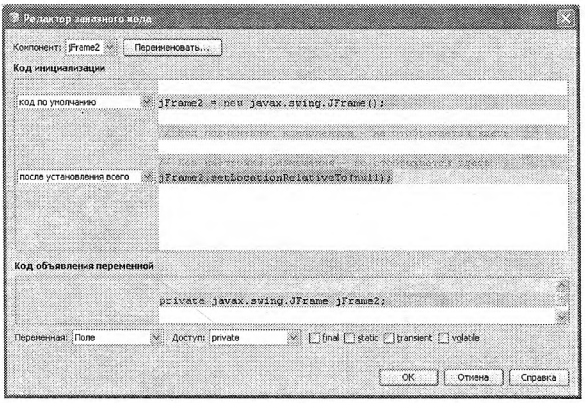
Для этого вписываем в указанное на рисунке место строку «jFrame1.setLocationRelativeTo(null);»:



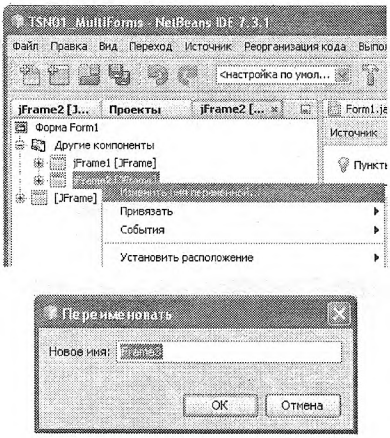
Выбираем третье окно jFrame2 и настраиваем его для расположения по центру экрана:



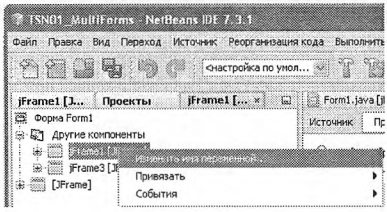
Для этого вписываем в указанное на рисунке место строку «jFrame2.setLocationRelativeTo(null);»:

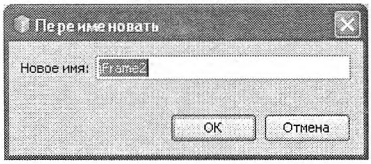


Переименовываем окно jFrame2 в «jFrame3»:

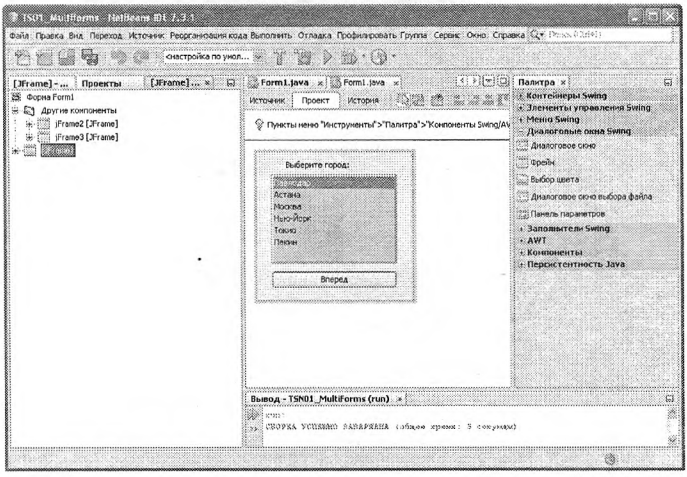


Переименовываем окно jFrame1 в «jFrame2»:

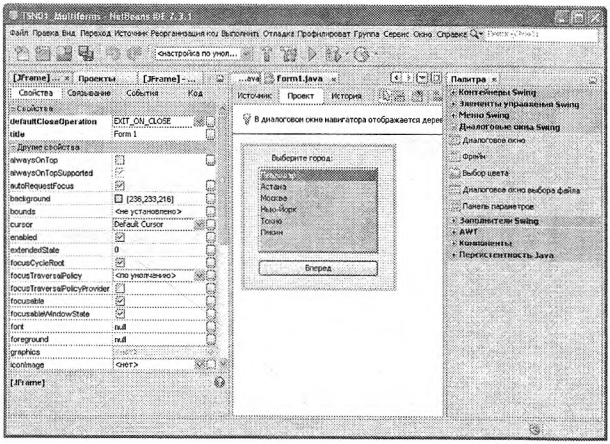




Настройка и установки главного (первого) окна. Наносим и настраиваем компоненты в главном окне:



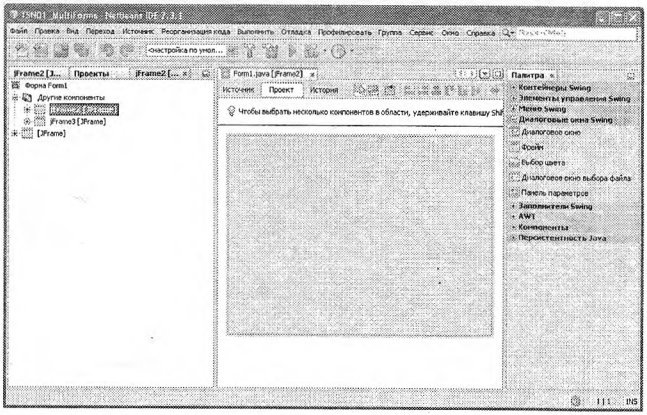
Задаем название окну название «Form1» в свойстве «title»:



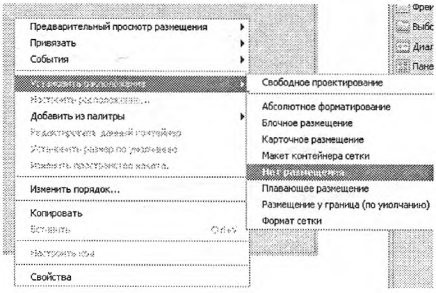
Делаем в главном окне обработчик нажатия кнопки:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Переход на 2 окно, кнопка «Вперед»
    this.setVisible(false); // Спрятать главное окно 1
    jFrame2.setVisible(true); // Показать окно 2
}
```

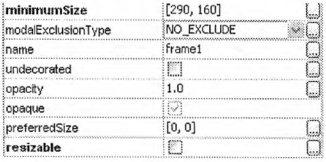
Настройка и установки второго окна. Выбираем второе окно:



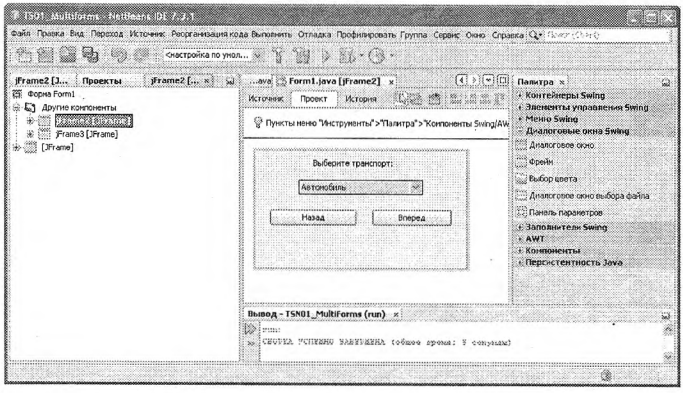
Указываем второму окну свободное размещение компонент:



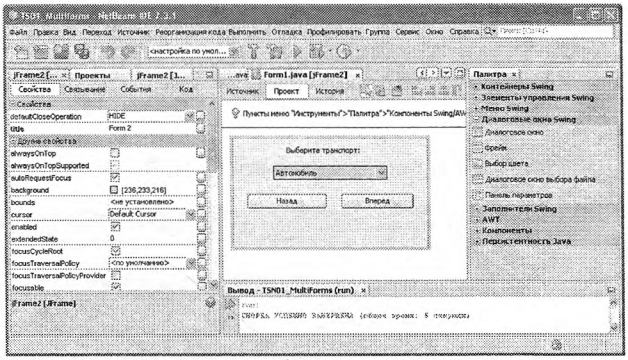
Запрещаем второму окну изменять размеры – «resizable» и задаем нужный размер в свойстве «minimumSize». Без задания данного размера форма не будет нормально отображаться.



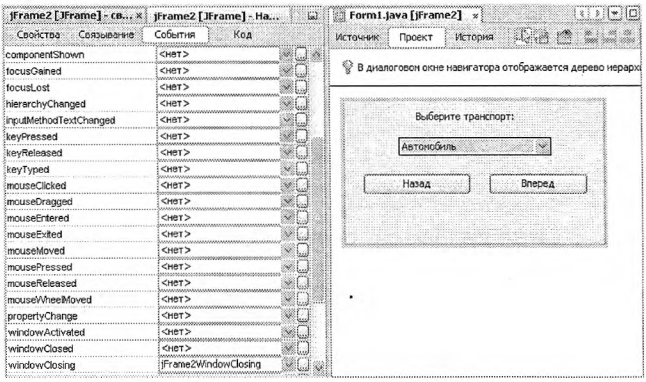
Наносим и настраиваем компоненты во втором окне:



Задаем второму окну название «Form2» в свойстве «title»:



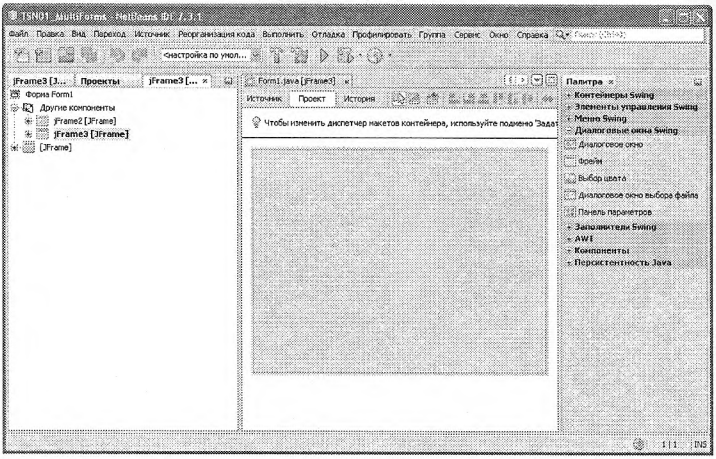
Делаем во втором окне jFrame2 процедуру на закрытие и обработчики нажатия кнопок:



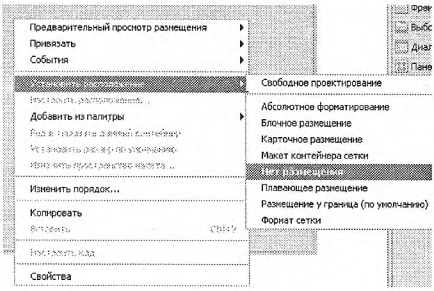

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Переход на главное окно 1 из 2 окна, кнопка «Назад»
    jFrame2.setVisible(false); // Спрятать окно 2
    this.setVisible(true); // Показать главное окно 1
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // Переход на 3 окно из 2 окна, кнопка «Вперед»
    jFrame2.setVisible(false); // Спрятать окно 2
    jFrame3.setVisible(true); // Показать окно 3
}
```

```
private void jFrame2WindowClosing(java.awt.event.WindowEvent evt) {
    // Выход из программы
    System.exit(0); // Выход из программы
}
```

Настройка и установки третьего окна. Выбираем третье окно:



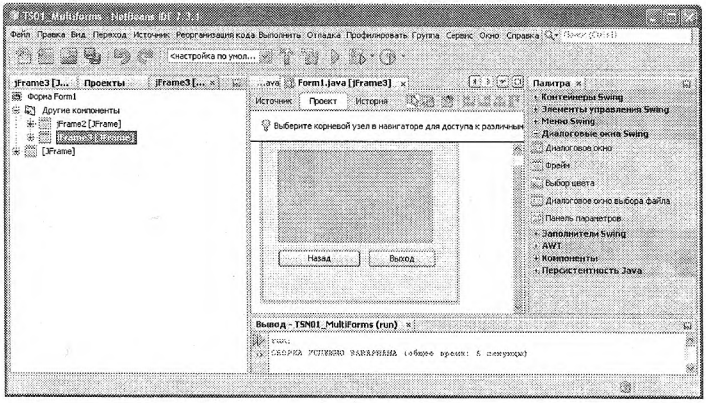
Указываем третьему окну свободное размещение компонент:



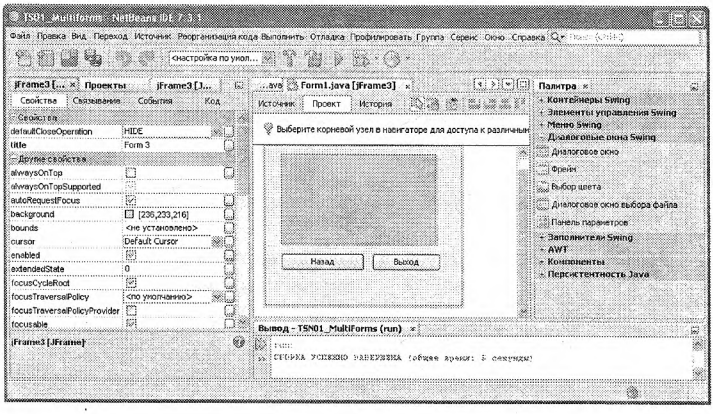
Запрещаем третьему окну изменять размеры – «resizable» и задаем нужный размер в свойстве «minimumSize». Без задания данного размера форма не будет нормально отображаться.

minimumSize	[255, 210]
modalExclusionType	NO_EXCLUDE
name	frame3
undecorated	
opacity	1.0
opaque	
preferredSize	[0, 0]
resizable	

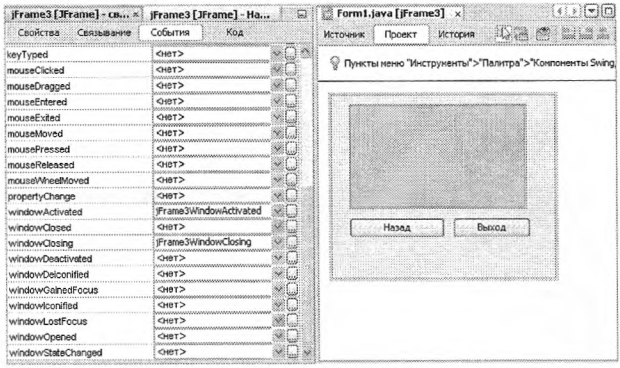
Наносим и настраиваем компоненты в третьем окне:



Задаем третьему окну название «Form3» в свойстве «title»:



Делаем в третьем окне процедуру на закрытие и открытие окна и обработчики нажатия кнопок:




```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // Переход на 2 окно из 3 окна, кнопка «Назад»
    jFrame3.setVisible(false); // Спрятать окно 3
    jFrame2.setVisible(true); // Показать окно 2
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // Кнопка выхода
    System.exit(0); // Выход из программы
}

private void jFrame3WindowClosing(java.awt.event.WindowEvent evt) {
    // Выход из программы
    System.exit(0); // Выход из программы
}

private void jFrame3WindowActivated(java.awt.event.WindowEvent evt)
{
    // Выдача результатов при открытии окна 3
    jTextArea1.setText(""); // Очистка компонента
    jTextArea1.append("Ваш выбор:" + "\n");
    jTextArea1.append("_____" + "\n");
    jTextArea1.append("Город: " + jList1.getSelectedValue().toString() +
"\n");
```

```
    jTextArea1.append("Транспорт: " +
jComboBox1.getSelectedItem().toString() + "\n");
}
```

3.15 Многоклассовые программы

Класс – структура, состоящая из полей и методов. Поля хранят какие-либо значения, а методы позволяют к ним обращаться и проводить над ними необходимые манипуляции.

Объект – это созданный на основе класса экземпляр класса. Программа работает с созданными из классов объектами.

Одни классы могут создаваться на основе других, ранее созданных классов. Класс потомок, созданный на основе родительского класса (называемого суперклассом), наследует поля и методы родителя. В классе потомка родительские методы могут быть переопределены или перегружены.

Переопределенный (полиморфный) метод – метод с таким же названием и параметрами, но разным программным кодом. Перегруженный метод – метод с таким же названием и другими параметрами, разным программным кодом.

Класс имеет один или несколько методов-конструкторов, один из которых автоматически вызывается при создании объекта из класса. Класс может иметь секцию инициализации, также автоматически исполняемой при создании объекта из класса.

Пример консольной программы с одним классом решения квадратного уравнения:

```
package tsn01.demo_sc;
import java.util.Scanner;

public class App1 {
  public static void main(String[] args) {
    // Вычисление квадратного уравнения
    double a, b, c; // Входные переменные
    QuEq eq = new QuEq(); // Создаем объект, решающий квадратное
уравнение
    try {
      Scanner sc = new Scanner(System.in); //Создаем объект для ввода
данных с консоли
      System.out.println("Решение квадратного уравнения");
      System.out.print("Введите a="); a = sc.nextDouble(); // Ввод
значения a
```

```
        System.out.print("Введите b="); b = sc.nextDouble(); // Ввод
значения b
        System.out.print("Введите c="); c = sc.nextDouble(); // Ввод
значения c
        if (eq.Calc(a, b, c) == true) { // Если есть решение, то
           System.out.format("x1=%.3f%nx2= %.3f%n", eq.x1, eq.x2); //
Вывод ответа
     } else { System.out.println("Нет решения!"); } // Если нет решения,
то сообщение об этом
   } catch (Exception e) {}
 }
}

class QuEq { // Класс "Квадратное уравнение"
private double a, b, c; // Входные переменные
   public double x1, x2; // Искомые значения
private double d; // Дискриминант

   QuEq() { a = 0; b = 0; c = 0; d = 0; x1 = 0;  x2 = 0; } // Конструктор
класса

   public boolean Calc(double a, double b, double c) { // Метод класса для
расчета
      this.a = a; this.b = b; this.c = c;
      d = (b * b) - 4 * a * c;
      try {
         x1 = (-b + Math.sqrt(d)) / (2 * a); // x1
         x2 = (-b - Math.sqrt(d)) / (2 * a); // x2
         if (!(Double.isNaN(x1)) && (!Double.isInfinite(x1))
               && (!(Double.isNaN(x2)) && (!Double.isInfinite(x2)))) {
            return true; } else { return false; }
   } catch (Exception e) { return false; }
 }
}
```

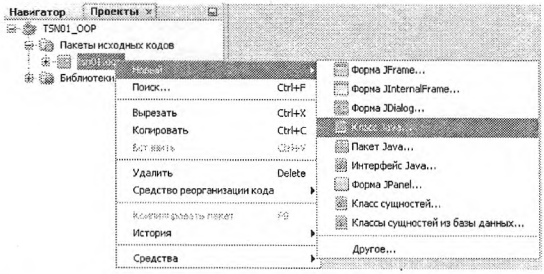
Различие между конструкторами и методами в классе:

```
package tsn.utils;

 class MyClass {
   public MyClass() { // Это конструктор
```

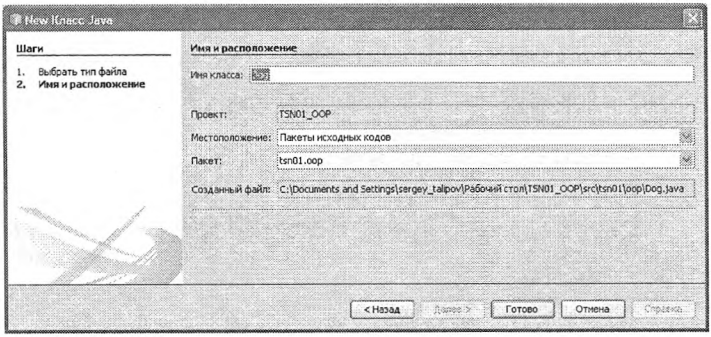
```
    System.out.println("This is constructor");
}

  public void MyClass() { А это метод с таким же названием, разница в
возвращаемом значении (void)
    System.out.println("This is metod");
 }
}

public class MainClass {
    public static void main(String[] args) {
    MyClass mc = new MyClass(); // Создание объекта класса и вызов
его конструктора
    mc.MyClass(); // Вызов метода класса
    }
}
```

Создание дополнительных классов в программах. В многоклассовых программах несколько классов могут храниться в одном файле с расширением «java», но более правильно хранить каждый класс в отдельном файле. Рассмотрим создание новых классов в отдельных файлах программы.

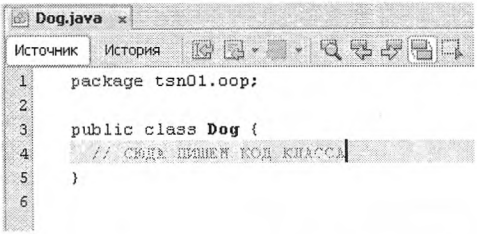
Для добавления в программу новых классов, необходимо вызвать правой кнопкой мыши контекстное меню на имени пакета, выбрать опцию «Новый» - «Класс Java»:



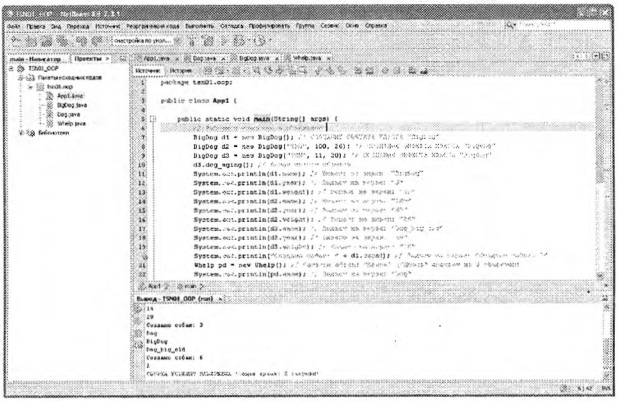
Далее необходимо указать желаемое имя для класса, который нужно создать (имя пакета изменять не нужно):



Заготовка класса готова, теперь нужно вписать в нее нужный код:



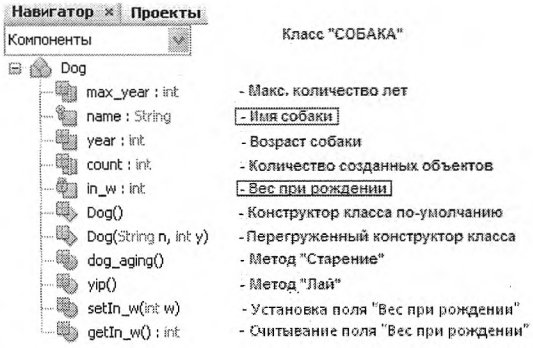
Пример консольной программы с несколькими классами (модель «собака – большая собака – щенок»)



Главный класс «App1»:

```
package tsn01.oop;
public class App1 {
   public static void main(String[] args) {
      // Работа с классами и объектами
      BigDog d1 = new BigDog(); // Создание объекта класса "BigDog"
      BigDog d2 = new BigDog("TSN", 100, 26); // Создание объекта
класса "BigDog"
      BigDog d3 = new BigDog("TSN", 11, 28); // Создание объекта
класса "BigDog"
      d3.dog_aging(); // Вызов метода объекта
      System.out.println(d1.name); // Выдаст на экран: "BigDog"
      System.out.println(d1.year); // Выдаст на экран: "5"
      System.out.println(d1.weight); // Выдаст на экран: "10"
      System.out.println(d2.name); // Выдаст на экран: "TSN"
      System.out.println(d2.year); // Выдаст на экран: "50"
      System.out.println(d2.weight); // Выдаст на экран: "26"
      System.out.println(d3.name); // Выдаст на экран: "Dog_big_old"
      System.out.println(d3.year); // Выдаст на экран: "14"
      System.out.println(d3.weight); // Выдаст на экран: "28"
      System.out.println("Создано собак: " + d1.count); // Выдаст на
экран: "Создано собак: 3"
      Whelp pd = new Whelp(); // Создаем объект "Щенок" ("Щенок"
состоит из 3 объектов)
      System.out.println(pd.name); // Выдаст на экран: "Dog"
      pd.dg.dog_aging(); // "Состарим отца щенка"
      System.out.println(pd.bt.name); // Выдаст на экран: "BigDog"
      System.out.println(pd.dg.name); // Выдаст на экран: "Dog_big_old"
      System.out.println("Создано собак: " + d1.count); // Выдаст на
экран: "Создано собак: 6"
      //pd.in_w = 2; // Так нельзя, поле скрытое от прямого доступа
      pd.setIn_w(2); // Установка поля "Вес при рождении"
      System.out.println(pd.getIn_w()); // Выдаст на экран: "2"
      // d1 = ("BigDog", 5, 10) - все параметры по умолчанию
      // d2 = ("TSN", 50, 26) - все параметры явно указаны, возраст
автоограничен
      // d3 = ("Dog_big_old", 14, 28) - все параметры явно указаны, но
первые два после переопределены
}}
```

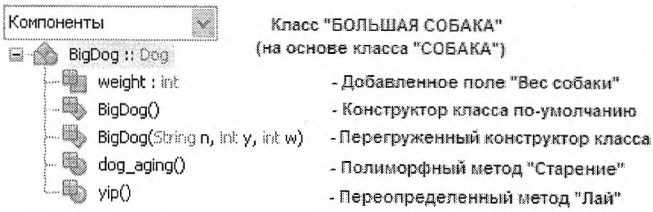
Класс « Dog» – «Собака»:




```
package tsn01.oop;

public class Dog { // Класс "собака"
   final static int max_year = 50; // Константа «Максимальное
количество лет»
   protected  String name; // Поле "Имя собаки"
   int year; // Поле "Количество лет собаки"
   static int count = 0; // Статическое поле, количество созданных
объектов собак
private int in_w = 1; // Инкапсулированное (скрытое) поле "Вес при
рождении"

   Dog() { // Метод-конструктор по-умолчанию
      name = "Dog"; // Задаем значение в поле "Имя собаки"
      year = 0; // Задаем значение в поле "Количество лет собаки"
      count++; // Увеличиваем число созданных объектов в поле "count"
}

   Dog(String n, int y) { // Перезагрузка метода-конструктора
      name = n; // Задаем значение в поле "Имя собаки"
      if (y > max_year) { // Проверяем год
         year = max_year; // Задаем значение в поле "Количество лет
собаки"
   } else {
         year = y; // Задаем значение в поле "Количество лет собаки"
   }
```

```
        count++; // Увеличиваем число созданных объектов в поле "count"
    }

    void dog_aging() { // Метод класса: делаем собаку "старой", меняя ей
имя и возраст (вес не изменяем)
        name = "Dog_old"; // Задаем значение в поле "Имя собаки"
        year = 10; // Задаем значение в поле "Количество лет собаки"
    }

    void yip() { // Метод "лай"
        System.out.println("yip"); // Вывод на экран фразы "yip"
    }
    void setIn_w(int w) { // Установка поля "Вес при рождении"
        in_w = w;
    }

    int getIn_w() { // Считывание поля "Вес при рождении"
        return in_w;
    }

}
```

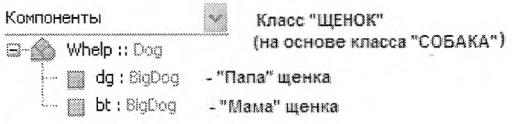
Класс «BigDog» – «Большая собака»:




```
package tsn01.oop;

// Класс "большая собака"
public class BigDog extends Dog { // Наследование класса от другого
класса

 int weight; // Новое поле "Вес собаки"
```

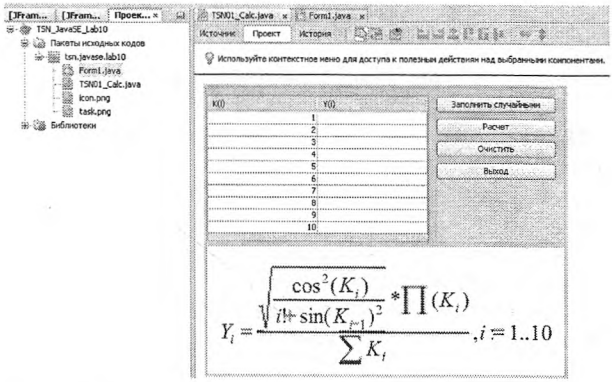
```
  BigDog() { // Метод-конструктор по-умолчанию
     name = "BigDog"; // Задаем значение в поле "Имя собаки"
     year = 5; // Задаем значение в поле "Количество лет собаки"
     weight = 10; // Задаем значение в поле "Вес собаки"
}

  BigDog(String n, int y, int w) { // Перезагрузка метода-конструктора
     super(n, y); // Вызов конструктора суперкласса (т.е. "Dog(n,y)")
     weight = w; // Установка значения у поля
}

  void dog_aging() { // Полиморфизм (переопределение) метода
"Старение"
     name = "Dog_big_old"; // Задаем значение в поле "Имя собаки"
     year = 14; // Задаем значение в поле "Количество лет собаки"
}

  void yip() { // Переопределенный (полиморфный) метод "Лай"
     System.out.println("yip-yip-yip"); // Вывод на экран фразы "yip"
}

}
```

Класс «Whelp» – «Щенок»:




```
package tsn01.oop;
public class Whelp extends Dog { // класс "щенок"
  public BigDog dg; // Композиция объектов - Объект "Папа"
  public BigDog bt; // Композиция объектов - Объект "Мама"
  { // Секция инициализации класса
     dg = new BigDog(); // Создаем объект "Папа"
     bt = new BigDog();  // Создаем объект "Мама"
  }
}
```

Пример визуальной программы с несколькими классами.



Методы окна программы:

```
private void jButton_RandomActionPerformed(java.awt.event.ActionEvent
evt) {
    // Заполнение случайными числами
    for (int i = 0; i < jTable1.getRowCount(); i++) {
      jTable1.setValueAt((int) (Math.random() * 11), i, 0);
  }
}

private void jButton_TaskActionPerformed(java.awt.event.ActionEvent
evt) {
    // Выполнение задания
    // Проверка заполнения таблицы
    for (int i = 0; i < jTable1.getRowCount(); i++) {
      if (jTable1.getValueAt(i, 0) == null) {
        JOptionPane.showMessageDialog(rootPane, "Проверьте
правильность заполнения столбца K(i)", "Ошибка ввода",
JOptionPane.ERROR_MESSAGE);
        return;
      }
  }
    int rowCount = jTable1.getRowCount(); // Количество строк в
таблице
```

```
    int[] inArray = new int[rowCount]; // Создаем исходный массив
целых чисел
    for (int i = 0; i < rowCount; i++) { // Заполняем исходный массив из
первого столбца таблицы
      inArray[i] = (int) jTable1.getModel().getValueAt(i, 0);
  }
    TSN01_Calc ma = new TSN01_Calc(); // Создаем объект-решатель
задания
    ma.SetMass(inArray); // Передаем в объект-решатель исходный
массив
     double[] outArray = ma.GetMass(); // Получаем с объекта решателя
значение результирующего массива
    for (int i = 0; i < rowCount; i++) { // Заносим во второй столбец
таблицы значения из результирующего массива
      jTable1.getModel().setValueAt(String.format("%.3f", outArray[i]),
i, 1);
  }
}

private void jButton_ClearActionPerformed(java.awt.event.ActionEvent
evt) {
    // Очистка таблицы
    for (int i = 0; i < jTable1.getRowCount(); i++) {
      for (int j = 0; j < jTable1.getColumnCount(); j++) {
        jTable1.setValueAt(null, i, j);
    }
  }
}

private void jButton_ExitActionPerformed(java.awt.event.ActionEvent
evt) {
  System.exit(0); // Выход из программы
}
```

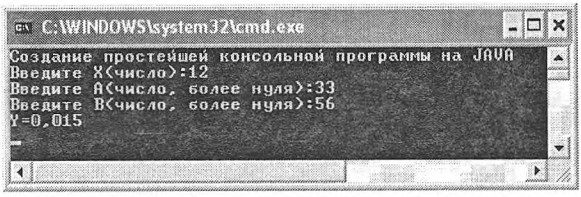
Методы класса-решателя задания:

```
package tsn.javase.lab10;

import java.util.Arrays; // Библиотека работы с массивами
import static java.lang.Math.*; // Импорт всех математических функций
```

```
public class TSN01_Calc { // Класс вычисления значений
   int[] InMass; // Переменная для входного массива целых чисел
   double[] OutMass; // Переменная для выходного массива
вещественных чисел
   int length_OrigMass; // Переменная для длины исходного массива

   public void SetMass(int OrigMass[]) { // Метод для задания исходных
данных
      length_OrigMass = OrigMass.length; // Определяем длину
исходного массива
      // Копируем значения из исходного массива во входной массив
      InMass = Arrays.copyOf(OrigMass, length_OrigMass);
      OutMass = new double[length_OrigMass]; // Создаем выходной
массив
}

   public double[] GetMass() { // Метод вычисления значения
      int ki, ki1;  double pr = 1, sum = 0, fk = 1;
      for (int i = 0; i < length_OrigMass; i++) { // Цикл по элементам
входного массива
         try {
            fk *= (i + 1); // Расчет факториала
            ki = InMass[i]; // Текущее значение элемента входного
массива
            pr *= ki; // Расчет произведения элементов входного массива
            sum += ki; // Расчет суммы элементов входного массива
            ki1 = InMass[i - 1]; // Предыдущее значение элемента
входного массива
            // Вычисление элемента для выходного массива по
заданному алгоритму
            OutMass[i] = (sqrt(pow(cos(ki), 2)
                  / (fk + sin(pow(ki1, 2)))) * pr)  / sum;
      } catch (Exception ex) {
            OutMass[i] = Double.NaN; // Присваиваем значение элементу
"нет решения"
      }
   }
      return OutMass; // Возвращаем результирующий массив в
программу
   }
}
```

4 Задания к практическим работам

4.1 Практическая работа № 1 – Создание консольной программы



Сделать консольную программу расчета математической величины по заданному в варианте алгоритму. В программе предусмотреть защиту от неверно введенных данных и защиту от ошибки при отсутствии решения.

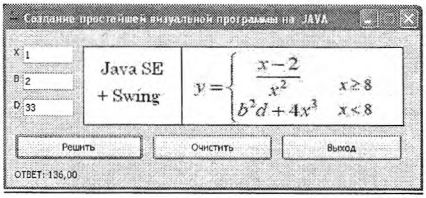
Варианты:

№	Задание
1	$y=\begin{cases}\dfrac{5x^2}{6(a+b)^2} & x\geq 5\\ x^3(a+b) & x<5\end{cases}$
2	$y=\begin{cases}\dfrac{x^2+a^2+b^2}{a+b} & x<7\\ x^3(a+b)^2 & x\geq 7\end{cases}$
3	$y=\begin{cases}\dfrac{a^2}{x^2}+6x & x\leq 4\\ b^2(4+x)^2 & x>4\end{cases}$
4	$y=\begin{cases}\dfrac{a^2+5x+b^2}{ab} & x\geq 4\\ x(a-b) & x<4\end{cases}$
5	$y=\begin{cases}\dfrac{a^2+4x^2+b}{2x} & x\geq 8\\ a^2-2x^2 & x<8\end{cases}$
6	$y=\begin{cases}\dfrac{6x^2-ab}{2x^2} & x>6\\ 4(x+a^2+b^2) & x\leq 6\end{cases}$
7	$y=\begin{cases}\dfrac{6(a^2+x+b^2)}{abx} & x<8\\ 4(a^2-x+b^2) & x\geq 8\end{cases}$
8	$y=\begin{cases}\dfrac{4(a^2+2x+b^2)}{ab} & x\geq 6\\ x^3(a-b)^2 & x<6\end{cases}$
9	$y=\begin{cases}\dfrac{5ab}{x^2+a^2} & x>5\\ 4(a+b-x)^2 & x\leq 5\end{cases}$
10	$y=\begin{cases}\dfrac{5x^2+b^2}{a^2+b^2} & x>4\\ 6(x^2-a^2) & x\leq 4\end{cases}$
11	$y=\begin{cases}\dfrac{2abx}{(a+b)^2} & x<7\\ x(a^2+4b) & x\geq 7\end{cases}$
12	$y=\begin{cases}\dfrac{4a^2+bx}{(b+x)} & x>4\\ 3(a+b+x)^2 & x\leq 4\end{cases}$
13	$y=\begin{cases}\dfrac{a^2c+b^2-d}{x} & x\leq 5\\ x^2+5 & x>5\end{cases}$
14	$y=\begin{cases}\dfrac{(a+b)^2}{x-2} & x<6\\ xd^3+b^2 & x\geq 6\end{cases}$
15	$y=\begin{cases}\dfrac{x-2}{x^2} & x\geq 8\\ b^2d+4x^3 & x<8\end{cases}$
16	$y=\begin{cases}\dfrac{(x^2+a^2)c}{2b} & x<4\\ x^3(a-b) & x\geq 4\end{cases}$

17	$y=\begin{cases}\dfrac{x(a^2+b^2)}{6a} & x\ge 3\\ x(1-ab) & x<3\end{cases}$
18	$y=\begin{cases}\dfrac{5(a^2+b^2)}{x-4} & x\ge 5\\ 6ab-5x & x<5\end{cases}$
19	$y=\begin{cases}\dfrac{x+4a}{a^2b^2} & x\ge 4\\ x^3-ab & x<4\end{cases}$
20	$y=\begin{cases}\dfrac{5a^2-2}{x^2+b^2} & x>6\\ x+8a^2b & x\le 6\end{cases}$

21	$y=\begin{cases}\dfrac{(a+b)^2}{x^2} & x\le 9\\ x(a^2+b^2) & x>9\end{cases}$
22	$y=\begin{cases}\dfrac{a}{x}+\dfrac{b}{x^2} & x>6\\ a^2(x+b) & x\le 6\end{cases}$
23	$y=\begin{cases}\dfrac{x+4}{a^2+b^2} & x\le 7\\ x(a+b)^2 & x>7\end{cases}$
24	$y=\begin{cases}\dfrac{10(x+a^2)}{b+a} & x\ge 4\\ 5(x+a^2+b) & x<4\end{cases}$

4.2 Практическая работа № 2 – Создание простейшей визуальной программы



Сделать программу расчета математической величины по заданному в варианте алгоритму. Программа должна состоять из компонентов «*JTextField*» – для ввода и вывода значений, компонент «*JLabel*» – для поясняющих надписей на форме и для отображения картинки с вариантом задания и кнопки «*JButton*» с надписью «Решить» для расчета значения.

Варианты:

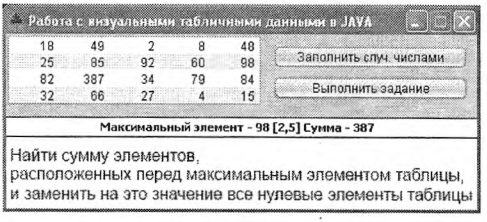
№	Задание
1	$y=\begin{cases}\dfrac{a^2c+b^2-d}{x} & x\le 5\\ x^2+5 & x>5\end{cases}$
2	$y=\begin{cases}\dfrac{(a+b)^2}{x-2} & x<6\\ xd^3+b^2 & x\ge 6\end{cases}$
3	$y=\begin{cases}\dfrac{(x^2+a^2)c}{2b} & x<4\\ x^3(a-b) & x\ge 4\end{cases}$

4	$y=\begin{cases}\dfrac{x(a^2+b^2)}{6a} & x\ge 3\\ x(1-ab) & x<3\end{cases}$
5	$y=\begin{cases}\dfrac{5(a^2+b^2)}{x-4} & x\ge 5\\ 6ab-5x & x<5\end{cases}$
6	$y=\begin{cases}\dfrac{x+4a}{a^2b^2} & x\ge 4\\ x^3-ab & x<4\end{cases}$

7	$y=\begin{cases}\dfrac{5a^2-2}{x^2+b^2} & x>6\\ x+8a^2b & x\le 6\end{cases}$
8	$y=\begin{cases}\dfrac{(a+b)^2}{x^2} & x\le 9\\ x(a^2+b^2) & x>9\end{cases}$
9	$y=\begin{cases}\dfrac{a}{x}+\dfrac{b}{x^2} & x>6\\ a^2(x+b) & x\le 6\end{cases}$
10	$y=\begin{cases}\dfrac{x+4}{a^2+b^2} & x\le 7\\ x(a+b)^2 & x>7\end{cases}$
11	$y=\begin{cases}\dfrac{10(x+a^2)}{b+a} & x\ge 4\\ 5(x+a^2+b) & x<4\end{cases}$
12	$y=\begin{cases}\dfrac{5x^2}{6(a+b)^2} & x\ge 5\\ x^3(a+b) & x<5\end{cases}$
13	$y=\begin{cases}\dfrac{x^2+a^2+b^2}{a+b} & x<7\\ x^3(a+b)^2 & x\ge 7\end{cases}$
14	$y=\begin{cases}\dfrac{x}{a^2}+\dfrac{x}{b^2} & x\ge 8\\ x(a+b)^2 & x<8\end{cases}$
15	$y=\begin{cases}\dfrac{a^2}{x^2}+6x & x\le 4\\ b^2(4+x)^2 & x>4\end{cases}$

16	$y=\begin{cases}\dfrac{a^2+5x+b^2}{ab} & x\ge 4\\ x(a-b) & x<4\end{cases}$
17	$y=\begin{cases}\dfrac{a^2+4x^2+b}{2x} & x\ge 8\\ a^2-2x^2 & x<8\end{cases}$
18	$y=\begin{cases}\dfrac{6x^2-ab}{2x^2} & x>6\\ 4(x+a^2+b^2) & x\le 6\end{cases}$
19	$y=\begin{cases}\dfrac{6(a^2+x+b^2)}{abx} & x<8\\ 4(a^2-x+b^2) & x\ge 8\end{cases}$
20	$y=\begin{cases}\dfrac{4(a^2+2x+b^2)}{ab} & x\ge 6\\ x^3(a-b)^2 & x<6\end{cases}$
21	$y=\begin{cases}\dfrac{5ab}{x^2+a^2} & x>5\\ 4(a+b-x)^2 & x\le 5\end{cases}$
22	$y=\begin{cases}\dfrac{5x^2+b^2}{a^2+b^2} & x>4\\ 6(x^2-a^2) & x\le 4\end{cases}$
23	$y=\begin{cases}\dfrac{2abx}{(a+b)^2} & x<7\\ x(a^2+4b) & x\ge 7\end{cases}$
24	$y=\begin{cases}\dfrac{4a^2+bx}{(b+x)} & x>4\\ 3(a+b+x)^2 & x\le 4\end{cases}$

4.3 Практическая работа № 3 – Работа с визуальными табличными данными



Имеется двухмерный массив 4x5 в виде компонента «*JTable*», кнопка «*JButton*» с надписью «Выполнить задание», кнопка «*JButton*»

с надписью «Заполнить случайными числами» и компонент «*JLabel*» для отображения текста варианта задания.

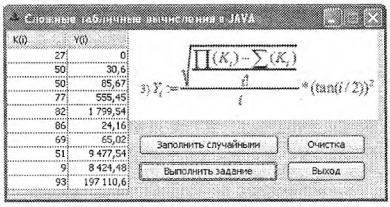
Кнопка «Заполнить случайными числами» заполняет массив случайными значениями от -0 до 100. Кнопка «Выполнить задание» выполняет обработку таблицы «*JTable*» по заданному в варианте алгоритму.

Варианты:

№	Задание
1	Подсчитать количество нулей в таблице и заменить на это значение все нечетные целые элементы таблицы
2	Поменять местами максимальный и минимальный элементы таблицы, если сумма значений таблицы больше ста
3	Подсчитать количество отрицательных элементов в таблице и увеличить на это значение минимальный и максимальный элементы таблицы
4	Определить, имеется ли в таблице хотя бы один нулевой элемент. Если такой элемент есть, то заменить все вещественные значения таблицы единицами
5	Если максимальный элемент в таблице расположен после минимального, то поменять значения элементов первой и второй строки между собой
6	Подсчитать количество четных целых элементов, расположенных перед максимальным элементом таблицы и увеличить на это значение максимальный элемент
7	Если после максимального элемента таблицы расположена хотя бы одна единица, то увеличить все положительные элементы таблицы в два раза
8	Если в таблице сумма значений не равна нулю, то уменьшить максимальный элемент таблицы в два раза, а минимальный элемент уменьшить в три раза
9	Если перед максимальным элементом таблицы расположены все единицы, то заменить максимальный элемент таблицы на количество этих единиц
10	Если максимальный элемент в таблице больше минимального в 10 или более раз, то все нули заменить единицами, а отрицательные числа заменить на их значения по модулю
11	Если в таблице количество нулей больше пяти, и количество положительных чисел больше трех, то увеличить максимальный элемент в два раза
12	Найти максимальный элемент второй строки таблицы и заменить его на сумму элементов второго столбца
13	Найти количество нулей во второй строке таблицы и заменить на это значение максимальный элемент второго столбца
14	Если максимальный элемент находится в последней строке таблицы, то увеличить все элементы первого столбца в 2 раза
15	Если минимальный элемент стоит во втором столбце, то заменить элементы этого столбца нулями
16	Если все элементы второго столбца равны 1, то заменить максимальный

	элемент таблицы на 100, а минимальный элемент на 5.
17	Если в третьей строке стоят все единицы, то увеличить максимальный элемент первого столбца в два раза, а максимальный элемент второго столбца в три раза.
18	Найти максимальный элемент второй строки. Если он больше первого элемента третьей строки, то поменять элементы местами
19	Если сумма элементов первой строки больше минимального элемента, то заменить этот минимальный элемент на найденную сумму
20	Если максимальный элемент находится в первом столбце, то увеличить все элементы третьей строки на 10
21	Поменять местами максимальные элементы первой и второй строк таблицы, если сумма элементов больше 100
22	Найти сумму элементов первой и второй строк и заменить на это значение максимальный элемент первого столбца, если он больше 100
23	Если сумма элементов первой строки не равна максимальному элементу таблицы, то увеличить этот максимальный элемент в два раза
24	Если во втором столбце стоят две единицы, то уменьшить максимальный элемент первой строки в два раза, а все единицы в таблице заменить нулями

4.4 Практическая работа № 4 – Сложные табличные вычисления



Имеется массив в два столбца и 10 строк в виде компонента «*JTable*», кнопки «Заполнить случайными числами» и «Выполнить задание». В первый столбец вводятся исходные данные – K_i .

Нажимая на кнопку «Выполнить задание» выполняется расчет значений во втором столбце – Y_i. Каждое значения Y_i зависит от соответствующего значения K_i и предыдущих значений K_i по заданному алгоритму в варианте.

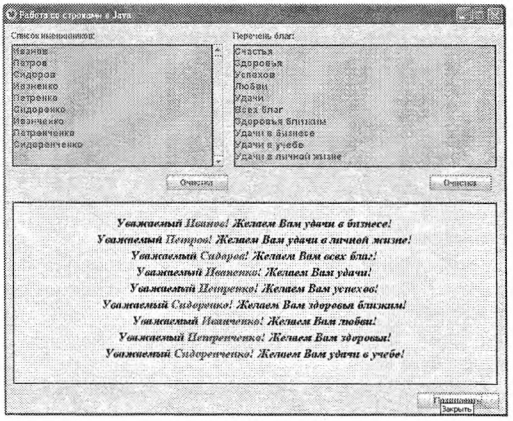
В программе необходимо сделать отображение варианта задания в компоненте ««*JLabel*». Кнопка «Заполнить случайными числами» служит для заполнения случайными числами первого столбца таблицы, т.е. значениями K_i.

Варианты:

№	Задание
1	$Y_i = \sqrt{\frac{\sin(K_i) - \sin(K_{i-1})}{\frac{\tan(K_i - K_{i-1})}{2}}}$, $i = 1..10$
2	$Y_i = \sqrt{\frac{\sin(K_i^2)}{i!}} + \frac{tg(\cos^3(K_{i-1}))}{3.5 * \sin(K_i)}, i = 1..10$
3	$Y_i = \begin{cases} \sin(\sum K_i) / K_i \text{ , } K_i < 0 \\ \cos^2(\sqrt{K_i}) \text{ , } K_i = 0, i = 1..10 \\ \prod(K_i) - \tan(K_{i-1}) \text{ , } K_i > 0 \end{cases}$
4	$Y_i = \sqrt{\frac{\tan(K_i - K_{i-1})}{2}} \div (\sqrt{\frac{\prod(K_i)}{i!}})$, $i = 1..10$
5	$Y_i = \begin{cases} \frac{\sqrt{\sum(K \text{ четных до i})}}{\prod(\text{K нечетных до i})} \text{ , } K_i >= 0 \text{ , } i = 1..10 \\ 0 \text{ , } K_i < 0 \end{cases}$
6	$Y_i = \frac{\sqrt{\frac{ax^3 - bx^2 + x}{\sqrt{x}}}}{\sin(K_i)}$, $x = \sum(K_i > 0)$, a и b задаются в JTextField, $i = 1..10$
7	$Y_i = \left[\frac{\prod(K_i) - \tan(K_{i-1})}{\sqrt{\frac{\sin^3(K_i^2)}{i!}}} \right]^3$, $i = 1..10$
8	$Y_i = \frac{\sqrt{\frac{\tan(K_i)^2 - a^2}{i! + \sin(K_{i-1})^b}}}{\cos^3(K_i)}$, $i = 1..10$, a и b задаются в JTextField
9	$Y_i = \frac{\sqrt{\frac{\prod(K_i) - \sum(K_{i-1})}{\sin^2(K_i + K_{i-1})}}}{\lvert K_i \rvert}$, $i = 1..10$
10	$Y_i = \sqrt{\frac{\cos^2(K_i)}{(a^2 + b^2) - \sin(K_i)}} * \sum[(K_{i-1})]$, $i = 1..10$; a и b задаются в JTextField
11	$Y_i = \begin{cases} \prod(\text{K нечетных до i}) - \sum(\text{K четных до i}), K_i \geq 0 \\ \sqrt[3]{\sum(K_i)^2}, K_i < 0 \end{cases}$, $i = 1..10$
12	$Y_i = \frac{\sqrt{\frac{\cos^2(K_i)}{i! + \sin(K_{i-1})^2}} * \prod(K_i)}{\sum K_i}$, $i = 1..10$

13	$Y_i = \left(\frac{ax^4 + bx^3}{a^2 - b^3}\right)^3$ / [$\prod(K_i) - \sum$K(нечетных до i)] , i=1..10; a, b, x задаются в JTextField
14	$Y_i = \frac{\sqrt{\prod(K_i) - \sum(K_{i-1})}}{i! + \sin(K_i)}$, $i = 1..10$
15	$Y_i = \left(\frac{\sqrt{\sin^3(K_i^2)}}{i! + 5}\right)^{1/3}$, $i = 1..10$
16	$Y_i = \sqrt{\frac{\prod(K_i < 0) + \sum(K_i > 0)}{\sqrt{\sum[\sin(K_i) - \cos(K_i)]}}}$, $i = 1..10$
17	$Y_i = \frac{\sqrt[3]{(K_i)^2 + (K_{i-1})^5}}{tg(\sum K_i)} + i!$, $i = 1..10$
18	$Y_i = \frac{\sqrt[3]{\cos(K_{i-1})} + \sqrt[5]{tg(\frac{K_i}{K_{i-1}})}}{\sum K_i}$, $i = 1..10$
19	$Y_i = \frac{\sqrt[3]{(\sum K_i)^2}}{\prod K_{i-1}} * \left[\sin^2(K_i) - \cos^2(K_{i-1})\right]^3$, $i = 1..10$
20	$Y_i = \sqrt[3]{\frac{(a^2 + b^2)^3 * \cos^2(K_i)}{i! - \prod(K_i)}}$, $i = 1..10$, a и b задаются в JTextField
21	$Y_i = \sqrt{\frac{tg(\cos^2(K_{i-1}))}{\sin^2(K_i)}} * \sqrt{\sum(K_i > 0)}$, $i = 1..10$
22	$Y_i = \frac{\sqrt{\sum(K_i > 0)} + i!/2}{\prod(K_i < 0) - \sum(K_{i-1})}$, $i = 1..10$
23	$Y_i = \frac{\sqrt{a^2 + x^4}}{i!} * \left[\sin(K_i) - \cos(K_{i-1})\right]$, i=1..10; a и x задаются в JTextField
24	$Y_i = \frac{\sum(K$ нечетных до i$) + \prod($K четных до i$)}{\sqrt[3]{i! - \sin(K_i) / \cos(K_{i-1})}}$, $i = 1..10$

4.5 Практическая работа № 5 – Работа со строками



Имеется текст в «JTextArea»-компоненте. Данный текст необходимо преобразовать в соответствии с заданным в варианте алгоритмом и результат поместить в другой «JTextArea»-компонент или «JEditorPane»-компонент.

Варианты:

№	Задание
1	Текст записан одной длинной строкой. Признаком красной строки служит символ $. Переформатировать текст в 60-символьные строки, формируя абзацы
2	Дан текст программы на языке «Pascal» и список зарезервированных слов языка (begin, end, for и др.) в английской транскрипции. Преобразовать текст, записав все зарезервированные слова прописными буквами, а остальной текст – строчными. Русские буквы не изменять
3	В заданном тексте подсчитать частоту использования каждого слова (словосочетания) из заданного списка. Отсортировать полученные данные по-возрастанию
4	В имеющемся словаре найти группы слов, записанных одними и теми же буквами и отличающиеся только их порядком, т.е. перестановкой, например, «КОМАР» и «КОРМА»
5	В заданном тексте определить 5 наиболее часто встречающихся слов с указанием количества использования каждого из них
6	Имеется зашифрованный текст, где в каждом слове каждый второй символ – ненужный. Расшифровать текст
7	Из имеющегося словаря выбрать наиболее длинное слово, в котором все буквы разные, например: ЛЕЙКОПЛАСТЫРЬ, НЕРЯШЛИВОСТЬ,

	ЧЕТЫРЕХДЮЙМОВКА
8	Для заданного текста построить табличную гистограмму распределения длин слов
9	В заданном русском тексте выбрать слова, которые без искажения могут быть написаны латинскими буквами, например: СВЕТА РОЕТ РОВ, ВОВКА СЕЕТ ОВЕС
10	В тексте пропущены некоторые слова и словосочетания. Эти слова и словосочетания представлены отдельным списком в том порядке, в каком должны быть вставлены. Места вставки отмечены в тексте символом «$». Откорректировать текст
11	Дан текст. Найти количество слов, начинающихся с буквы «а» и заканчивающихся на «я»
12	Дан текст. Найти длину самого короткого слова и самого длинного слова
13	Дан текст. Подсчитать, сколько различных символов встречаются в нем. Вывести их на экран
14	Дан текст. Подсчитать самую длинную последовательность подряд идущих букв «а»
15	Дан текст. Удалить из нее те слова, которые содержат хотя бы одну букву «к»
16	Дан текст. Найти в нем те слова, которые начинаются и заканчиваются одной и той же буквой
17	В тексте слова зашифрованы – каждое из них записано наоборот. Расшифровать сообщение
18	Дан тест. Найти в ней те слова, которые заканчиваются буквой, следующей за первой в алфавите
19	Определить, сколько раз в тексте встречается заданное слово
20	Дан текст на русском языке со словом-буквой. Найти слово, содержащее наибольшее количество указанных букв
21	Дан текст. Удалить из него все лишние пробелы, оставив между словами не более одного пробела.
22	Дан тест. Необходимо преобразовать тест так, чтобы все слова в нем были не более трех символов (остальные символы слов удалить)
23	Дан тест. Удалить из него все слова, содержащие символы, отличные от русских букв
24	Дан тест. Удалить из него все слова, содержащие нечетное количество символов

4.6 Практическая работа № 6 – Работа с графикой

Необходимо сделать программу, которая создает на панели формы заданный рисунок. Рисунок необходимо создавать на компоненте «JPanel» программно, только из команд рисования по канве, и в точности соответствовать заданному в варианте. На рисунке необходимо дополнительно нарисовать инициалы автора программы и номер варианта через тире.

Варианты:

№	Задание	№	Задание
1		13	
2		14	
3		15	
4		16	
5		17	
6		18	
7		19	
8		20	
9		21	
10		22	
11		23	
12		24	

4.7 Практическая работа № 7 – Многооконная программа

Необходимо создать программу из пяти окон, с возможностью последовательного перехода из одного окна в другое – вперед и назад. Данная программа должна выдавать справочную информацию об авторе в виде вопросов и ответов. Каждое окно программы должно быть посвящено определенному жизненному этапу.

Название и тематика окон следующая:

- «Приветствие» (окно 1) – окно с краткой анкетной информацией об авторе программы;
- «Мое детство» (окно 2) – окно с вопросами и ответами про детство (до 7 лет);
- «Мое отрочество» (окно 3) – окно с вопросами и ответами про отрочество (от 7 до 16 лет);
- «Моя юность» (окно 4) – окно с вопросами и ответами про юность (от 16 до 23 лет);
- «Спасибо!» (окно 5) – окно с перечнем последних вопросов, которые пользователь задал в каждом из окон 2, 3, 4.

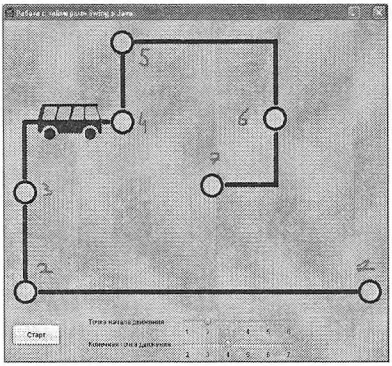
Каждое из окон 2, 3 и 4 должно содержать тот компонент для выбора вопросов, который указан в варианте. Ответ на выбранный вопрос должен отображаться в том же окне в компоненте «*JTextField*», в компоненте «*JTextArea*» или в компоненте «*JLabel*».

В пятом окне программа должна отобразить в указанном компоненте последний выбранный вопрос в каждой из форм 2, 3 и 4.

Варианты:

№	Компонент для выбора вопроса			Компонент для отображения списка заданных вопросов
	Окно 2	Окно 3	Окно 4	Окно 5
1	JList	JComboBox	JRadioButton	JTable
2	JComboBox	JRadioButton	JCheckBox	JTable
3	JCheckBox	JTable	JList	JTable
4	JTable	JList	JComboBox	JTable
5	JComboBox	JList	JRadioButton	JTextArea
6	JRadioButton	JComboBox	JCheckBox	JTextArea
7	JCheckBox	JRadioButton	JTable	JTextArea
8	JTable	JCheckBox	JList	JTextArea
9	JList	JTable	JComboBox	JTextArea
10	JRadioButton	JList	JComboBox	JTable
11	JCheckBox	JComboBox	JRadioButton	JTable
12	JTable	JRadioButton	JCheckBox	JTable
13	JList	JCheckBox	JTable	JTable
14	JComboBox	JTable	JList	JTable
15	JList	JRadioButton	JComboBox	JTextArea
16	JComboBox	JCheckBox	JRadioButton	JTextArea
17	JRadioButton	JTable	JCheckBox	JTextArea
18	JCheckBox	JList	JTable	JTextArea
19	JTable	JComboBox	JList	JTextArea
20	JComboBox	JRadioButton	JList	JTable
21	JRadioButton	JCheckBox	JComboBox	JTable
22	JCheckBox	JTable	JRadioButton	JTable
23	JTable	JList	JCheckBox	JTable
24	JList	JComboBox	JTable	JTable

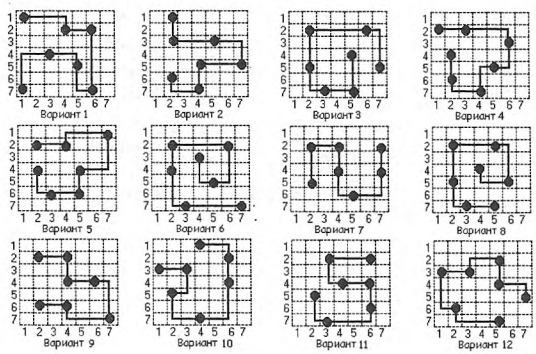
4.8 Практическая работа № 8 – Работа с таймерами

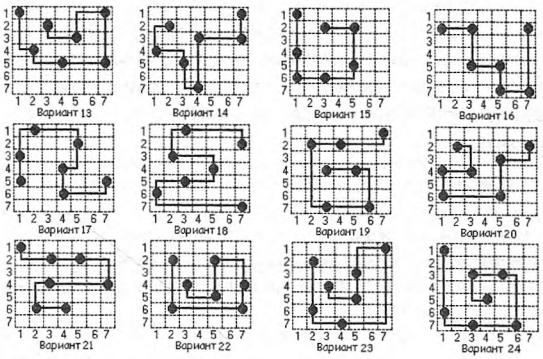


Имеется форма, на которой нарисован заданный автобусный маршрут с остановками и светофорами. Маршрут состоит из семи остановок и шести отрезков дорог. Пользователь выбирает любую начальную и конечную остановки из списка и нажимает кнопку «Движение». После нажатия на кнопку «Движение» на экране моделируется движения автобуса от начальной до конечной остановки.

Автобусный маршрут и «Автобус» представляет из себя компонент «JLabel» с картинкой. В программе необходимо использовать таймеры Swing с переключением активности между ними.

Варианты:





4.9 Практическая работа № 9 – Работа с массивами и файлами

. Имеется двухмерный массив 5х6 целых чисел, загружаемых из файла. Неоходимо создать программу, реализующую указанное в варианте задание. Программа должна:

- считать из входного файла «input.txt» данные (значения массива) и вывести их на экран – кнопка «Загрузить исходные данные из файла»;

- выполнить задание варианта: получить измененный массив (выходные данные) – кнопка «Обработать данные»;

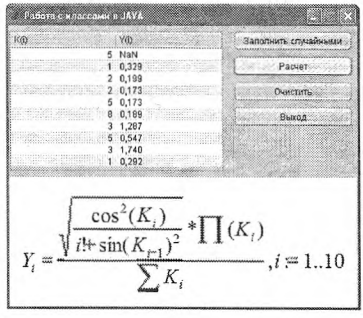
- вывести выходные данные (содержимое измененного массива) на экран и сохранить их в выходной файл «output.txt» – кнопка «Сохранить обработанные данные в файл».

Варианты:

№	Задание
1	Если максимальный элемент находится в последней строке таблицы, то увеличить все элементы первого столбца в 2 раза
2	Если минимальный элемент стоит во втором столбце, то заменить элементы этого столбца нулями
3	Если в третьей строке стоят все единицы, то увеличить максимальный элемент первого столбца в два раза, а максимальный элемент второго столбца в три раза.
4	Найти максимальный элемент второй строки. Если он больше первого элемента третьей строки, то поменять элементы местами
5	Если сумма элементов первой строки больше минимального элемента, то заменить этот минимальный элемент на найденную сумму
6	Если максимальный элемент находится в первом столбце, то увеличить все элементы третьей строки на 10
7	Поменять местами максимальные элементы первой и второй строк таблицы, если сумма элементов больше 100
8	Найти сумму элементов первой и второй строк и заменить на это значение максимальный элемент первого столбца, если он больше 100
9	Если сумма элементов первой строки не равна максимальному элементу таблицы, то увеличить этот максимальный элемент в два раза
10	Если во втором столбце стоят две единицы, то уменьшить максимальный элемент первой строки в два раза, а все единицы в таблице заменить нулями
11	Подсчитать количество нулей в таблице и заменить на это значение все нечетные целые элементы таблицы
12	Поменять местами максимальный и минимальный элементы таблицы, если сумма значений таблицы больше ста
13	Найти сумму элементов, расположенных перед максимальным элементом таблицы, и заменить на это значение все нулевые элементы таблицы
14	Подсчитать количество отрицательных элементов в таблице и увеличить на это значение минимальный и максимальный элементы таблицы
15	Определить, имеется ли в таблице хотя бы один нулевой элемент. Если такой элемент есть, то заменить все вещественные значения таблицы единицами
16	Если максимальный элемент в таблице расположен после минимального, то поменять значения элементов первой и второй строки между собой
17	Подсчитать количество четных целых элементов, расположенных перед максимальным элементом таблицы и увеличить на это значение максимальный элемент
18	Если после максимального элемента таблицы расположена хотя бы одна единица, то увеличить все положительные элементы таблицы в два раза

19	Если в таблице сумма значений не равна нулю, то уменьшить максимальный элемент таблицы в два раза, а минимальный элемент уменьшить в три раза
20	Если перед максимальным элементом таблицы расположены все единицы, то заменить максимальный элемент таблицы на количество этих единиц
21	Если максимальный элемент в таблице больше минимального в 10 раз, то все нули заменить единицами, а отрицательные числа заменить на их значения по модулю
22	Если в таблице количество нулей больше пяти, и количество положительных чисел больше трех, то увеличить максимальный элемент в два раза
23	Найти максимальный элемент второй строки таблицы и заменить его на сумму элементов второго столбца
24	Найти количество нулей во второй строке таблицы и заменить на это значение максимальный элемент второго столбца

4.10 Практическая работа № 10 – Работа с классами



Имеется таблица вещественных чисел в два столбца и 10 строк. В первом столбце имеются исходные данные – K_i. Необходимо выполнить расчет значений для второго столбце – Y_i. Каждое значения Y_i зависит от соответствующего значения K_i и предыдущих значений K_i по заданному в варианте алгоритму.

Необходимо написать программу, выполняющую данное задание. В программе необходимо сделать класс, обрабатывающий массив по алгоритму, указанному в варианте. Таблицу с данными реализовать через компонент JTable.

Варианты:

№	Задание
1	$Y_i = \frac{\sqrt{\tan(K_i)^2 - a^2}}{\cos^3(K_i)}$, $i = 1..10$, a и b задаются в JTextField
2	$Y_i = \frac{\sqrt{\frac{\prod(K_i) - \sum(K_{i-1})}{\sin^2(K_i + K_{i-1})}}}{\lvert K_i \rvert}$, i = 1..10
3	$Y_i = \sqrt{\frac{\cos^2(K_i)}{(a^2 + b^2) - \sin(K_i)}} * \sum[(K_{i-1})]$, i=1..10; a и b задаются в JTextField
4	$Y_i = \begin{cases} \prod(\text{K нечетных до i}) - \sum(\text{K четных до i}),\ K_i \geq 0 \\ \sqrt[3]{\sum(K_i)^2},\ K_i < 0 \end{cases}$, $i = 1..10$
5	$Y_i = \frac{\sqrt{\frac{\cos^2(K_i)}{i! + \sin(K_{i-1})^2}} * \prod(K_i)}{\sum K_i}$, $i = 1..10$
6	$Y_i = \left(\frac{ax^4 + bx^3}{a^2 - b^3}\right)^3$ / [$\prod(K_i) - \sum$K(нечетных до i)] , i=1..10; a, b, x задаются в JTextField
7	$Y_i = \frac{\sqrt{\frac{\prod(K_i) - \sum(K_{i-1})}{i! + \sin(K_i)}}}{\sin^2(K_i + K_{i-1})}$, $i = 1..10$
8	$Y_i = \left(\frac{\sqrt{\frac{\sin^3(K_i^2)}{i! + 5}}}{tg(\cos^2(\cos^2(K_i)))} \right)^{1/3}$, $i = 1..10$
9	$Y_i = \left\{ \frac{\prod(K_i < 0) + \sum(K_i > 0)}{\sqrt{\sum[\sin(K_i) - \cos(K_i)]}}, i = 1..10 \right.$
10	$Y_i = \frac{\sqrt[3]{(K_i)^2 + (K_{i-1})^5}}{tg(\sum K_i)} + i!$, $i = 1..10$
11	$Y_i = \frac{\sqrt[3]{\cos(K_{i-1})} + \sqrt[5]{tg(\frac{K_i}{K_{i-1}})}}{\sum K_i}$, $i = 1..10$
12	$Y_i = \frac{\sqrt[3]{(\sum K_i)^2}}{\prod K_{i-1}} * \left[\sin^2(K_i) - \cos^2(K_{i-1})\right]^3$, $i = 1..10$

13	$Y_i = \sqrt[3]{\dfrac{(a^2+b^2)^3 * \cos^2(K_i)}{i! - \prod(K_i)}}$, $i = 1..10$, a и b задаются в JTextField
14	$Y_i = \sqrt{\dfrac{tg(\cos^2(K_{i-1}))}{\sin^2(K_i)}} * \sqrt{\sum(K_i > 0)}$, $i = 1..10$
15	$Y_i = \dfrac{\sqrt{\sum(K_i > 0)} + i!/2}{\prod(K_i < 0) - \sum(K_{i-1})}$, $i = 1..10$
16	$Y_i = \dfrac{\sqrt{a^2 + x^4}}{i!} * \left[\sin(K_i) - \cos(K_{i-1})\right]$, i=1..10; a и x задаются в JTextField
17	$Y_i = \dfrac{\sum(K \text{ нечетных до i}) + \prod(\text{K четных до i})}{\sqrt[3]{i! - \sin(K_i)/\cos(K_{i-1})}}$, $i = 1..10$
18	$Y_i = \sqrt{\dfrac{\sin(K_i) - \sin(K_{i-1})}{\dfrac{\tan(K_i - K_{i-1})}{2}}}$, $i = 1..10$
19	$Y_i = \sqrt{\dfrac{\sin(K_i^2)}{i!}} + \dfrac{tg(\cos^3(K_{i-1}))}{3.5 * \sin(K_i)}$, $i = 1..10$
20	$Y_i = \begin{cases} \sin(\sum K_i)/K_i, & \mathrm{K_i} < 0 \\ \cos^2(\sqrt{K_i}), & \mathrm{K_i} = 0, i = 1..10 \\ \prod(K_i) - \tan(K_{i-1}), & \mathrm{K_i} > 0 \end{cases}$
21	$Y_i = \sqrt{\dfrac{\tan(K_i - K_{i-1})}{2}} \div \left(\sqrt{\dfrac{\prod(K_i)}{i!}}\right)$, $i = 1..10$
22	$Y_i = \begin{cases} \dfrac{\sqrt{\sum(K \text{ четных до i})}}{\prod(\text{K нечетных до i})}, & K_i >= 0, i = 1..10 \\ 0, & K_i < 0 \end{cases}$
23	$Y_i = \dfrac{\sqrt{\dfrac{ax^3 - bx^2 + x}{\sqrt{x}}}}{\sin(K_i)}$, $\mathrm{x} = \sum(\mathrm{K_i} > 0)$, a и b задаются в JTextField, $i = 1..10$
24	$Y_i = \left[\dfrac{\prod(K_i) - \tan(K_{i-1})}{\sqrt{\dfrac{\sin^3(K_i^2)}{i!}}}\right]^3$, $i = 1..10$

Литература

1 Аккуратов Е. Е. Знакомьтесь: Java [самоучитель N 1 по языкам Java 2 и Java Script] / Е. Е. Аккуратов. – М. : Вильямс, 2006. – С. 247.

2 Берд Б. Java 9 для чайников / Барри Берд [перевод с английского и редакция А. П. Сергеева]. – 7-е изд. – М. : Диалектика, 2018. – С. 616.

3 Блинов И. Н., Романчик В. С. Java. Методы программирования. – Минск: Четыре четверти, 2013. – С. 896.

4 Бруга, Любош. Java по-быстрому: практический экспресс-курс / Л. Бруга. – СПб. : Наука и Техника, 2006. – С. 369.

5 Васильев А. Н. Java. Объектно-ориентированное программирование: для магистров и бакалавров: базовый курс по объектно-ориентированному программированию / В. Н. Васильев. – М. : Питер, 2014. – С. 395.

6 Васильев А. Н. Самоучитель Java: с примерами и программами / Васильев А. Н. – 4-е изд. – СПб. : Наука и Техника, 2017. – С. 365.

7 Джошуа Блох. Java. Эффективное программирование: практическое пособие / Блох Джошуа. – Саратов: Профобразование, 2017. – С. 310.

8 Канель Е. Г., Фрайман В. М. Основы программирования на Java: для школьников и не только / Е. Г. Канель, З. Фрайман. – М. : URSS ЛЕНАНД, 2018. – С. 196 с.

9 Монахов В. В. Язык программирования Java и среда NetBeans. – СПб. : БХВ-Петербург, 2012. – С. 703.

10 Сеттер Р. Изучаем Java на примерах и задачах. – СПб. : Наука и Техника, 2016. – С. 240.

11 Хабибуллин И. Java 7. – СПб. : БХВ-Петербург, 2012. – С. 768.

12 Хорстманн К. С. Java SE 9: базовый курс: [перевод с английского] / Кей С. Хорстманн. – 2-е изд. – М. : Вильямс, 2018. С. – С. 573.

13 Шилдт Г. Java 8: руководство для начинающих: [современные методы создания, компиляции и выполнения программ на Java] / Герберт Шилдт. – 6-е изд. – М. : Диалектика, 2018. – С. 712.

14 Шилдт Г. Java: полное руководство [исчерпывающее описание языка программирования Java] / Герберт Шилдт. – 10-е изд. – М. : Диалектика, 2018. – С. 1488.

Содержание

С. Н. Талипов

СОВРЕМЕННОЕ ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ НА JAVA В СРЕДЕ NETBEANS

Учебное пособие

Технический редактор З. Ж. Шокубаева
Ответственный секретарь З. С. Искакова

Подписано в печать 14.01.2019 г.
Гарнитура Times.
Формат 60x90/16. Бумага офсетная.
Усл.печ. л. 11,16 Тираж 300 экз.
Заказ № 3343

Издательство «КЕРЕКУ»
Павлодарского государственного университета
имени С. Торайгырова
140008, г. Павлодар, ул. Ломова, 64