Лабораторная работа №9. Графический интерфейс java-приложений. Swing и AWT

Цель работы

Лабораторная работа №9 посвящена построению приложений с использованием графического интерфейса библиотек java.awt и javax.swing.

Указания к работе

Графический пользовательский интерфейс (GUI) – основной способ взаимодействия конечных пользователей с java-приложением. Для разработки прикладного программного обеспечения на языке Java, а точнее графического интерфейса приложений, обычно используются пакеты AWT и Swing.

AWT (для доступа загружается пакет java.awt) содержит набор классов, позволяющих выполнять графические операции и создавать оконные элементы управления, подобно тому, как это делается в VBA и Delphi;

Swing (для доступа загружается пакет javax.swing) содержит новые классы, в основном аналогичные AWT. К именам классов добавляется J (JButton, JLabel и др).

На данный момент основные классы для построения визуальных интерфейсов содержатся в пакете Swing. Из пакета AWT используются классы для обработки сообщений. Простейшее графическое приложение приведено ниже.

```
import javax. swing.*;

public final class HelloWorld implements Runnable {

public static void main(String[] args) {

//Swing имеет собственный управляющий поток (т.н. dispatching thread),

//который работает параллельно с основным (в котором выполняется main())

//потоком. Если основной поток закончит работу (метод main завершится),
```

```
//поток отвечающий за работу Swing-интерфейса может продолжать свою работу.

//И даже если пользователь закрыл все окна, программа продолжит свою работу

//(до тех пор, пока жив данный поток). Начиная с Java 6, когда все

//компоненты уничтожены, управляющий поток останавливается автоматически.

//Запускаем весь код, работающий в управляющем потоке, даже инициализацию:

SwingUtilities.invokeLater (new HelloWorld());

}

public void run() {

// Создаем окно с заголовком "Hello, World!"

Frame f = new JFrame ("Hello, World!");

// Ранее практиковалось следующее: создавался listener и регистрировался

// на экземпляре главного окна, который реагировал на windowClosing()

// принудительной остановкой виртуальной машины вызовом System.exit()

// Теперь же есть более "правильный" способ задав реакцию на закрытие окна.

// Данный способ уничтожает текущее окно, но не останавливает приложение. Тем

// самым приложение будет работать пока не будут закрыты все окна.

f.setDefaultCloseOperation (JFrame. DISPOSE_ON_CLOSE );

// однако можно задать и так:

// f.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

// Добавляем на панель окна нередактируемый компонент с текстом.

// f.getContentPane().add (new JLabel("Hello, World!")); - старый стиль

f.add(new JLabel("Hello World"));
```

```
// pack() "упаковывает" окно до оптимального размера

// всех расположенных в нем компонентов.

f.pack();

// Показать окно

f.setVisible(true);

}}
```

Технология Swing предоставляет механизмы для управления следующими аспектами представления:

Клавиатура (Swing предоставляет способ перехвата пользовательского ввода);

Цвета (Swing предоставляет способ менять цвета, которые вы видите на экране);

Текстовое поле для ввода (Swing предоставляет текстовые компоненты для обработки всех повседневных задач).

JComponent

Базовым классом всей библиотеки визуальных компонентов Swing является JComponent. Это суперкласс других визуальных компонентов. Он является абстрактным классом, поэтому в действительности вы не можете создать JComponent, но он содержит сотни функций, которые каждый компонент Swing может использовать как результат иерархии классов. Класс JComponent обеспечивает инфраструктуру окрашивания для всех компонентов, он знает, как обрабатывать все нажатия клавиш на клавиатуре, его подклассы, следовательно, должны только прослушивать определенные клавиши. Класс JComponent также содержит метод add(), который позволяет добавить другие объекты класса JComponent, так можно добавить любой Swing-компонент к любому другому для создания вложенных компонентов (например, JPanel, содержащую JButton, или даже более причудливые комбинации, например JMenu, содержащее JButton).

JLabel

Самым простым и в то же время основным визуальным компонентом в библиотеке Swing является JLabel, или «метка». К методам этого класса относится установка текста, изображения, выравнивания и других компонентов, которые описывает метка:

get/setText() – получить/установить текст в метке;

get/setIcon() – получить/установить изображение в метке;

get/setHorizontalAlignment – получить/установить горизонтальную позицию текста;

get/setVerticalAlignment() – получить/установить вертикальную позицию текста;

get/setDisplayedMnemonic() – получить/установить мнемонику (подчеркнутый символ) для метки;

get/setLabelFor() – получить/установить компонент, к которому присоединена данная метка; когда пользователь нажимает комбинацию клавиш Alt + мнемоника, фокус перемещается на указанный компонент.

JButton

Основным активным компонентом в Swing является Jbutton.

Методы, используемые для изменения свойств JButton, аналогичны методам JLabel (вы обнаружите, что они аналогичны для большинства Swing-компонентов). Они управляют текстом, изображениями и ориентацией:

get/setText() – получить/установить текст в кнопке;

get/setIcon() – получить/установить изображение в кнопке;

get/setHorizontalAlignment() – получить/установить горизонтальную позицию текста;

get/setVerticalAlignment() – получить/установить вертикальную позицию текста;

get/setDisplayedMnenomic() – получить/установить мнемонику (подчеркнутый символ), которая в комбинации с кнопкой Alt вызывает нажатие кнопки.

JFrame

КлассJFrame является контейнером, позволяющим добавлять к себе другие компоненты для их организации и предоставления пользователю.

JFrame выступает в качестве моста между независимыми от конкретной операционной системы Swing-частями и реальной операционной системой, на которой они работают. JFrame регистрируется как окно и таким образом получает многие свойств окна операционной системы: минимизация/максимизация, изменение размеров и перемещение. Хотя выполнении лабораторной работы достаточно считать JFrame палитрой, на которой вы размещаете компоненты. Перечислим некоторые из методов, которые вы можете вызвать в JFrame для изменения его свойств:

get/setTitle() – получить/установить заголовок фрейма;

get/setState() – получить/установить состояние фрейма (минимизировать, максимизировать и т.д.);

is/setVisible() – получить/установить видимость фрейма, другими словами, отображение на экране;

get/setLocation() – получить/установить месторасположение в окне, где фрейм должен появиться;

get/setSize() – получить/установить размер фрейма;

add() – добавить компоненты к фрейму.

Схемы, модели и события

При построении визуальных приложений в Java нельзя просто случайно разместить их на экране и ожидать от них немедленной работы. Компоненты необходимо разместить в определенные места, реагировать на взаимодействие с ними, обновлять их на основе этого взаимодействия и заполнять данными. Для эффективной работы с визуальными компонентами необходима установка следующих трех архитектурных составляющих Swing.

Схемы (layout). Swing содержит множество схем, которые представляют собой классы, управляющие размещением компонентов в приложении и тем, что должно произойти с ними при изменении размеров окна приложения или при удалении или добавлении компонентов.

События (event). Программа должна реагировать на нажатия клавиш, нажатия кнопки мыши и на все остальное, что пользователь может сделать.

Модели (model). Для более продвинутых компонентов (списки, таблицы, деревья) и даже для некоторых более простых, например, JComboBox, модели – это самый эффективный способ работы с данными. Они удаляют большую часть работы по обработке данных из самого компонента (вспомните MVC) и предоставляют оболочку для общих объектных классов данных (например, Vector и ArrayList).

Особое внимание, в связи с необходимостью изображения динамических сцен на визуальных компонентах необходимо уделить классу Graphics 2D.

Пример

Задание: Отобразить вращение треугольника вокруг своего центра тяжести

Решение:

```
import java.awt.*;

import java.awt.event.*;

import java.awt.geom.*;

import javax.swing.*;

public class Main {

public static void main(String[] args) {

JFrame fr=new JFrame("Вращение треугольника вокруг своего центра тяжести");
```

```
fr.setPreferredSize( new Dimension(300,300));
final JPanel pan= new JPanel();
fr.add(pan);
fr.setVisible(true);
fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
fr.pack();
Timer tm= new Timer(500, new ActionListener(){
int i=0;
@Override
public void actionPerformed(ActionEvent arg0) {
Graphics2D gr=(Graphics2D)pan.getRootPane().getGraphics();
pan.update(gr);
GeneralPath path=new GeneralPath();
path.append(new Polygon(new int []{60,-80,50},new int[]{-60,-50,40},3),true);
int x=(60-80+50)/3,y=(-60-50+40)/3;
gr.translate(150, 150);
AffineTransform tranforms = AffineTransform.getRotateInstance((i++)*0.07, x,
y);
gr.transform(tranforms);
gr.draw(path);
}});
tm.start();
}
}
```

Задания к лабораторной работе

1. Задать движение по экрану строк (одна за другой) из массива строк. Направление движения по апплету и значение каждой строки выбирается случайным образом.
2. Задать движение окружности по апплету так, чтобы при касании границы окружность отражалась от нее с эффектом упругого сжатия.
3. Изобразить в апплете приближающийся издали шар, удаляющийся шар. Шар должен двигаться с постоянной скоростью.
4. Изобразить в окне приложения отрезок, вращающийся в плоскости экрана вокруг одной из своих концевых точек. Цвет прямой должен изменяться при переходе от одного положения к другому.
5. Изобразить в окне приложения отрезок, вращающийся в плоскости фрейма вокруг точки, движущейся по отрезку.
6. Изобразить четырехугольник, вращающийся в плоскости апплета вокруг своего центра тяжести.
7. Создать фрейм с областью для рисования «пером». Создать меню для выбора цвета и толщины линии.
8. Составить программу для управления скоростью движения точки по апплету. Одна кнопка увеличивает скорость, другая – уменьшает. Каждый щелчок изменяет скорость на определенную величину.
9. Изобразить в окне гармонические колебания точки вдоль некоторого горизонтального отрезка. Если длина отрезка равна q, то расстояние от точки до левого конца в момент времени t можно считать равным $q(1 + \cos(wt))/2$, где w – некоторая константа. Предусмотреть поля для ввода указанных величин и кнопку для остановки и пуска процесса.
10. Создать апплет со строкой движущейся по диагонали. При достижении границ апплета все символы строки случайным образом меняют регистр. При этом шрифт меняется на шрифт, выбранный из списка.
11. Создать апплет со строкой движущейся горизонтально, отражаясь от границ апплета и меняя при этом свой цвет, на цвет выбранный из выпадающего списка.

12. Промоделировать вращение спутника вокруг планеты по эллиптической орбите. Когда скрывается за планетой – спутник не виден.

13. Промоделировать аналоговые часы (со стрелками) с кнопками для увеличения/уменьшения времени на час/минуту.

14. Задать движение множества снежинок с верхней части апплета вниз с разной скоростью. При достижении нижней границы снежинки исчезают и появляются в случайной позиции сверху. Снежинки должны быть представлены в виде кругов разного радиуса. Цвет каждой снежинки выбирается случайным образом.

15. Изобразить Солнце и планеты, вращающиеся вокруг него по круговым орбитам с различной скоростью. Размеры орбит и планет пропорциональны их реальным размерам. Планеты должны быть окрашены в разные цвета. Предусмотреть кнопку для ускорения/замедления движения.