Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Чувашский государственный университет имени И.Н. Ульянова»

ГРАФИЧЕСКИЕ ИНТЕРФЕЙСЫ ПОЛЬЗОВАТЕЛЯ

Лекция 15

Назарова Ольга Васильевна

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.

Пакет AWT (загружается java.awt) содержит набор классов, позволяющих выполнять графические операции и создавать элементы управления. Этот пакет поддерживается последующими версиями языка, однако считается весьма ограниченным и недостаточно эффективным.

Пакет Swing (загружается javax.swing, имя javax обозначает, что пакет не является основным, а только расширением языка) содержит улучшенные и обновленные классы, по большей части аналогичные AWT. К именам этих классов добавляется J (JButton, JLabel и т.д.). Пакет является частью библиотеки JFC (Java Foundation Classes), которая содержит большой набор компонентов JavaBeans, предназначенных для создания пользовательских интерфейсов.

Апплеты – это небольшие программы, встраиваемые в Web-документ и использующие для своей визуализации средства Web-браузера. Графические приложения сами отвечают за свою прорисовку. Апплеты используют окна, производные от класса Panel, графические приложения используют окна, производные от класса Frame, порожденного от класса Window.

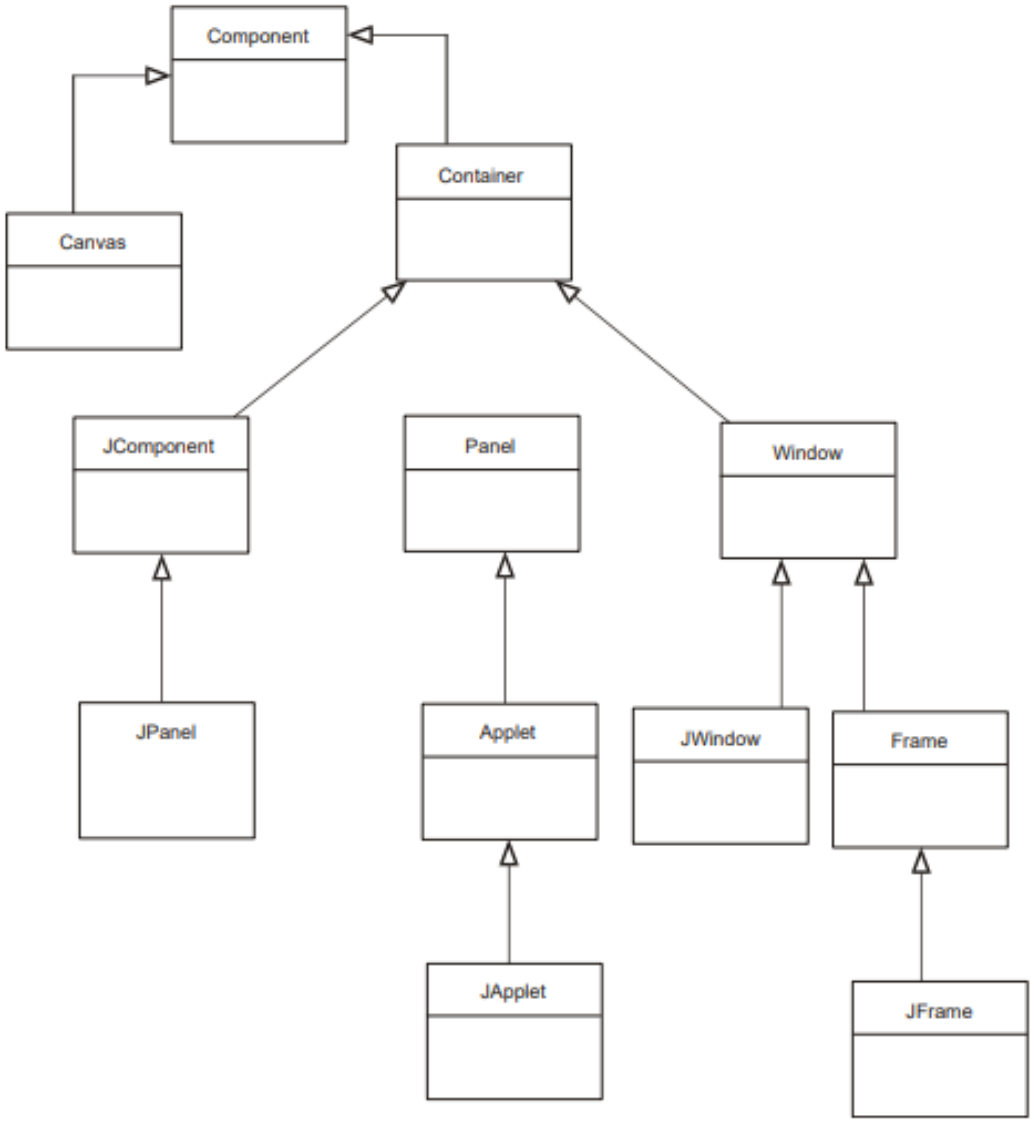

Рис. 1. Иерархия классов основных графических компонентов AWT и Swing

Суперкласс **java.awt.Component** является абстрактным классом, инкапсулирующим все атрибуты визуального компонента. Класс содержит большое число методов для создания компонентов управления и событий, с ними связанных.

Порожденный от него подкласс **Container** содержит методы типа **add()**, которые позволяют вкладывать в него другие компоненты (объекты) и отвечает за размещение любых компонентов, которые он содержит. Класс **Container** порождает классы **Panel** и **Window** – фундаментальные классы при создании апплетов и фреймов.

Класс **Panel** используется апплетом, графический вывод которого рисуется на поверхности объекта **Panel** - окна, не содержащего области заголовка, строки меню и обрамления. Основу для его отображения предоставляет браузер.

Графические приложения используют класс **Window** (окно верхнего уровня), который является основой для организации фрейма, но сам непосредственно для вывода компонент не используется. Для этого применяется его подкласс **Frame**. С помощью объекта типа **Frame** создается стандартное окно со строкой заголовка, меню, размерами.

Аналогично классы из пакета **Swing** используют для вывода графических изображений окна **JPanel** и **JFrame**. Еще один используемый для вывода класс **Canvas**, представляющий пустое окно, в котором можно рисовать, является наследником класса **Component**.

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.

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).

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();
}
}
```