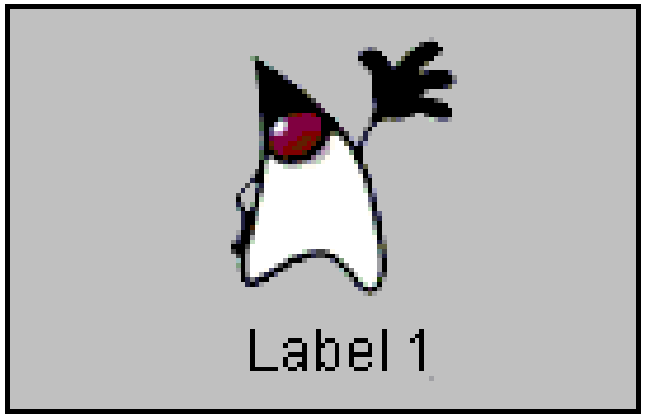
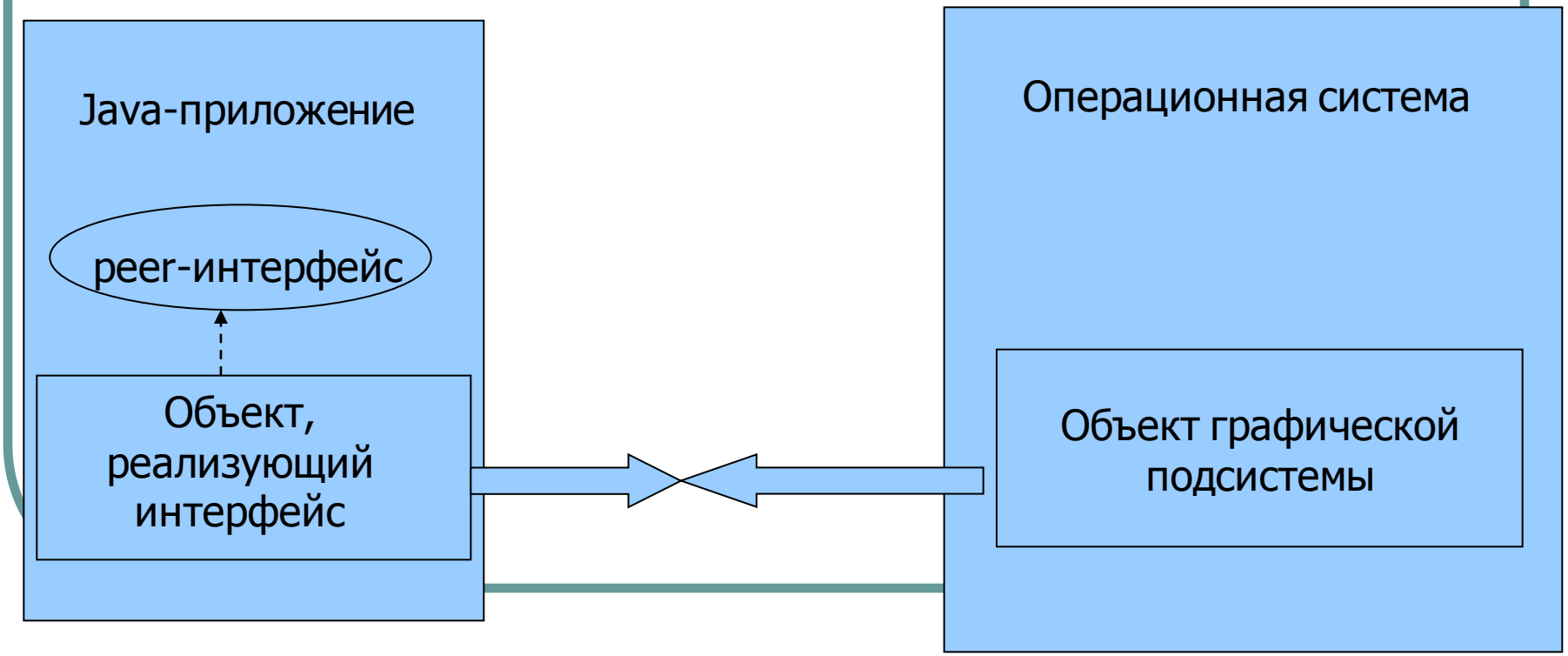
Средства создания графического интерфейса в Java



Библиотека AWT

- peer-to-peer интерфейсы («тяжелые» компоненты)



«Легкие» (lightweight) компоненты

- Сохранение заданного при создании вида (look and feel)
- Возможность изменить вид в любой момент работы приложения (PL&F)
- Библиотека «легких» компонентов Swing

Библиотека графических средств Java (JFC)

- AWT
- Swing
- Java 2D
- DnD
- Input Method Framework
- Accessibility

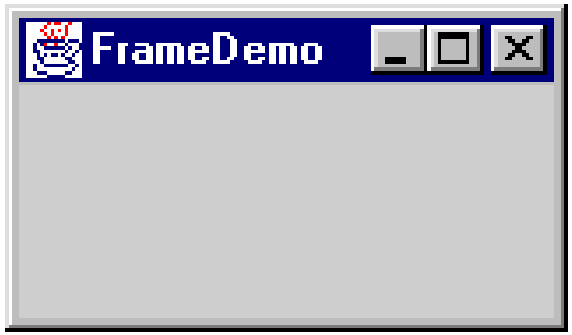
Основа ГИП в Java

- *Компонент* – отдельный, полностью определенный элемент, который можно использовать в графическом интерфейсе независимо от других элементов
- Каждый компонент перед выводом на экран помещается в *контейнер.*

Класс Frame

```
import java.awt.*;

class SimpleFrame extends Frame{
  public static void main(String[] args){
        Frame f=new SimpleFrame();
        f.setSize(400,150);
        f.setVisible(true);
}
}//для завершения Ctrl+C
```



События окна

```
import java.awt.*;
import java.awt.event.*;

class FirstFrame extends Frame{
  FirstFrame(String s){
    super(s);
    setSize(400,150);
    setVisible(true);
    addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent ev){
            System.exit(0);
        }
    });
}
  public static void main(String[] args){
      new FirstFrame( "Первое окно");
}
}
```

Управление выводом в окне

```
import java.awt.*;
import java.awt.event.*;

class SecondFrame extends Frame{
  SecondFrame(String s){
    super(s);
  }
  public void paint(Graphics g){
   g.setFont( new Font("Serif",Font.ITALIC|Font.BOLD,30));
   g.drawString(" Это второе окно!!!", 20,100);
}
public static void main(String[] args){
    Frame f= new SecondFrame( "Второе окно");
    f.setSize(400,150);
    f.setVisible(true);
    f.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent ev){
            System.exit(0);
        }
    });
}
}
```

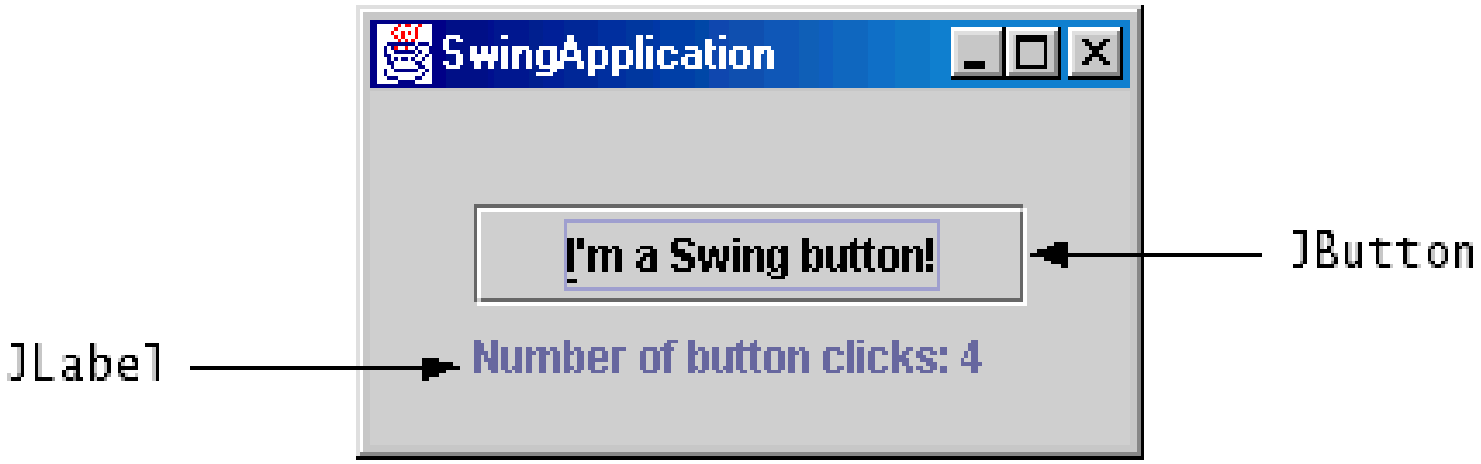
Окно библиотеки Swing

```
import javax.swing.*;

class FirstJFrame extends JFrame{
  FirstJFrame(String s){
    super(s);
    setSize(400,150);
    setVisible(true);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
  public static void main(String[ ] args){
      new FirstJFrame( "Первое Swing -окно");

}
}
```

Вид Swing-окна



MVC-схема в компонентах Swing

- *Модель* –один или несколько классов, в которых хранится и вырабатывается вся информация, связанная с компонентом, состояние объектов, созданных этим компонентом.

Для ввода и изменения информации используются методы

setXXX()

MVC-схема в компонентах Swing

- *Вид* – один или несколько классов, определяющих способ представления на экране или другом устройстве результатов, полученных моделью, в определенном виде

Получают информацию из модели методами getXXX() и isXXX()

Модель сообщает видам об изменении состояния объекта методами fireXXX()

MVC-схема в компонентах Swing

- *Контроллер* – классы, создающие интерфейс для ввода информации и изменения состояния объекта. Они реагируют на события ввода клавиатуры, действия мыши и пр. и обращаются к методам setXXX() модели

MVC-схема в компонентах Swing

Вид регистрируется (подписывается) у модели. *Контроллер*, реагируя на события меняет *модель*. *Модель* сообщает (рассылает) видам об изменении состояния. *Вид* забирает измененную информацию у *модели*.

Контроллер и *вид* не взаимодействуют.

MVC-схема в компонентах Swing

- Модели описаны интерфейсами, в которых перечислены необходимые методы (ButtonModel)
- У каждого интерфейса есть хотя бы одна стандартная реализация (DefaultButtonModel)
- Возможны реализации в виде абстрактных классов (AbstractListModel)

MVC-схема в компонентах Swing

Для реализации модели используется делегирование полномочий.

В качестве модели данных используется представитель – экземпляр класса с именем вида xxxModel. Обычно это защищенное или закрытое поле.

Доступ к модели осуществляется через метод getModel(). Если необходимо заменить модель используется метод setModel(xxxModel).

Компонент JTable

- В качестве модели используются классы-делегаты
  - TableModel
  - TableColumnModel
  - JTableHeader

Компонент JTable

- Для отображения элементов таблицы используется класс-делегат
  - TableCellRenderer

Компонент JTable

- За выделение ячеек и строк отвечает
  - ListSelectionModel
- Редактирование ячеек передается класссу
  - TableCellEditor

Пример таблицы (модель по умолчанию)

```
private Object[ ][ ] cells = {
    {"tt",234.5,10,false,Color.red},
    {"trr",24.675,-1000,true,Color.yellow}
   };
 private String[ ] columnNames = {"one","2","III","****","5"};

JTable tabl = new JTable (cells, columnNames);
 add( new JScrollPane(tabl), BorderLayout.CENTER);
```

Пример

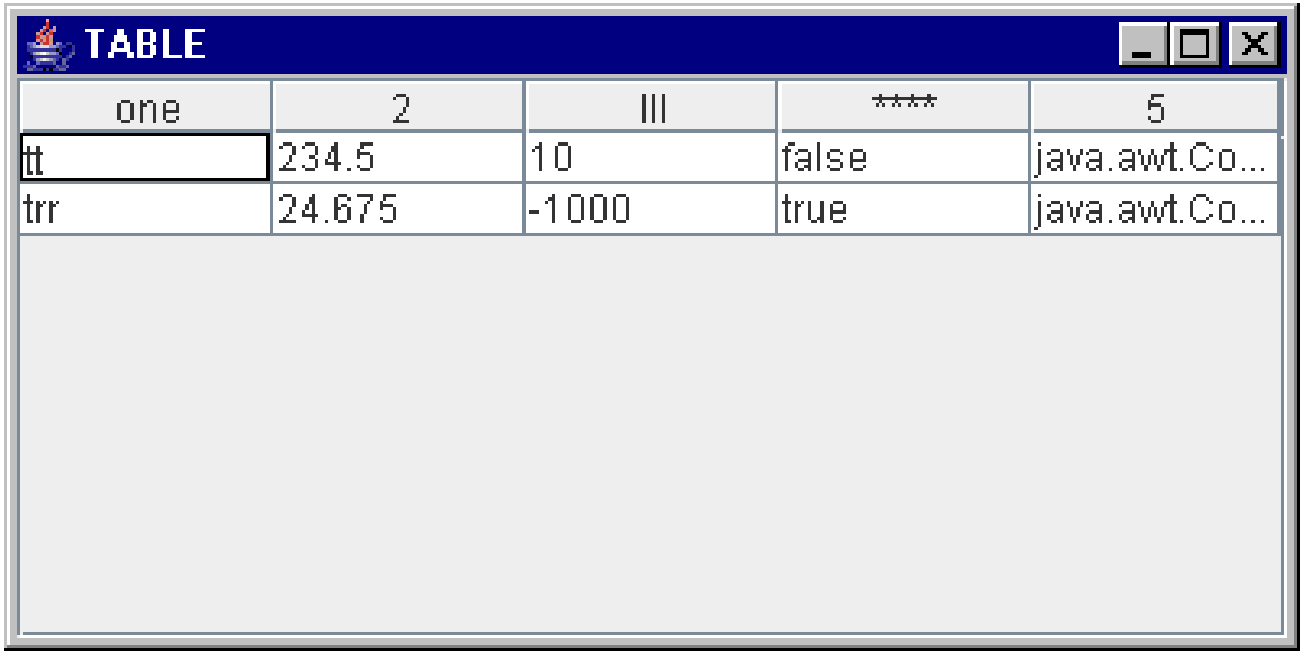


TABLE
one
2
III
****
5
tt
234.5
10
false
java.awt.Co...
trr
24.675
-1000
true
java.awt.Co...

Пример

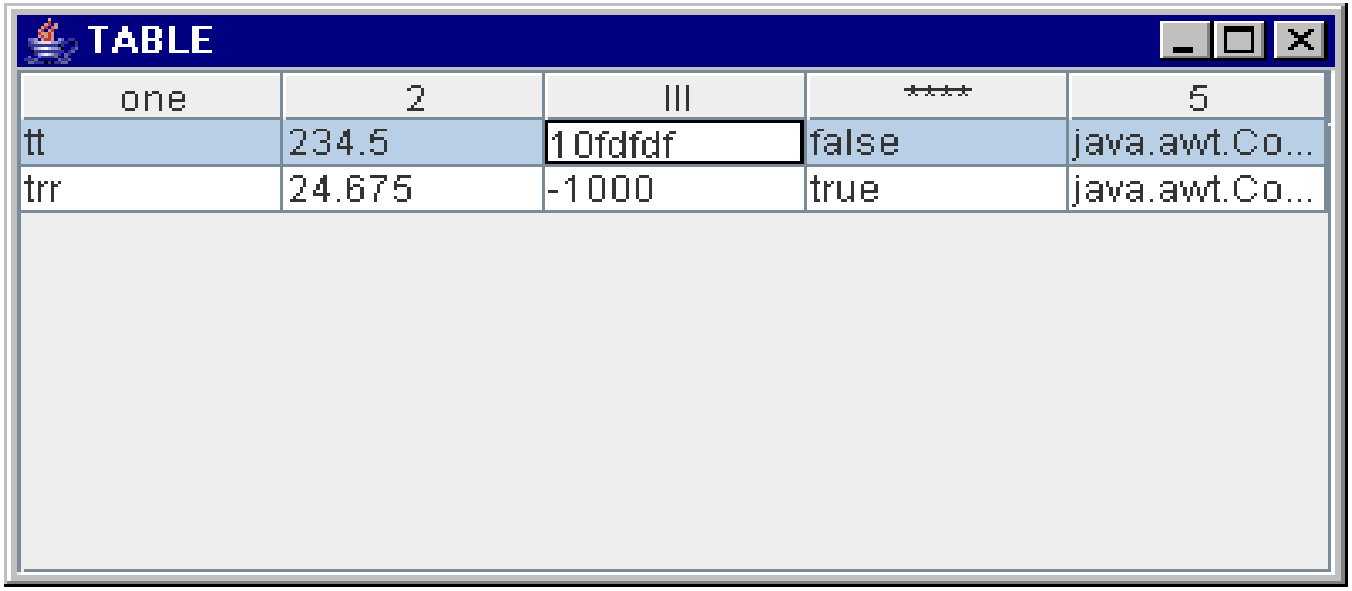


TABLE
one
2
III
****
5
tt
234.5
10fdfdf
false
java.awt.Co...
trr
24.675
-1000
true
java.awt.Co...

Собственная табличная модель

```
class FirstModel extends AbstractTableModel {
private Object[][] cell = {    {"tt",234.5,10,false,Color.red},
   {"trr",24.675,-1000,true,Color.yellow}   };
 private String[] columnNames = {"one","2","III","****","5"};
 public FirstModel() { super();}
 public int getRowCount() {return cell.length;}
 public int getColumnCount() {return cell[0].length;}
 public Object getValueAt (int row, int col) {return cell[row][col];}
 public void setValueAt (Object value, int row, int col) {
  if (col!=0) cell[row][col] = value;
 }
```

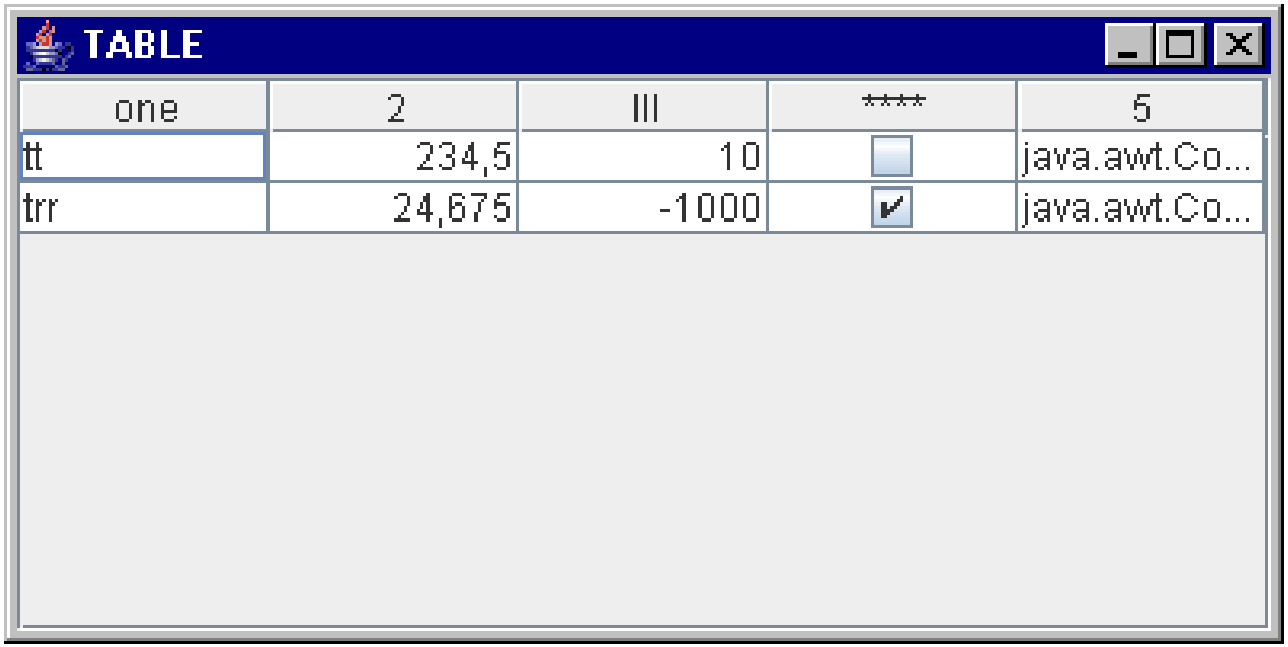
Собственная табличная модель(продолжение)

```
public boolean isCellEditable (int row, int col) {
   return col !=0;
}
 public Class getColumnClass(int col) {
   return cell[0][col].getClass();
}
public String getColumnName(int col) {
  return columnNames[col];
}
}
```

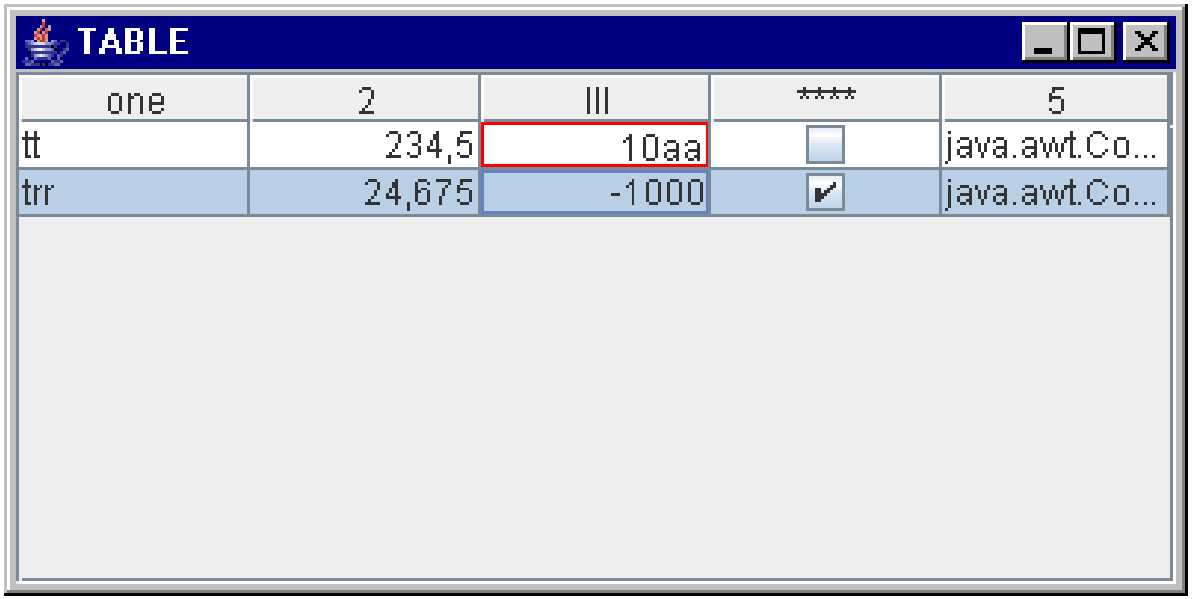
Собственная табличная модель(продолжение)

```
JTable tabl = new JTable (new FirstModel());
```

Собственная табличная модель(продолжение)



Собственная табличная модель(продолжение)



Изменение вида ячеек

```
class ColorRenderer extends DefaultTableCellRenderer {
  public void setValue(Object value) {
  setBackground((Color) value);
}
}

 JTable tabl = new JTable (new FirstModel());
 tabl.setDefaultRenderer(Color.class, new ColorRenderer());
```

Изменение вида ячеек