Элементы ГИП

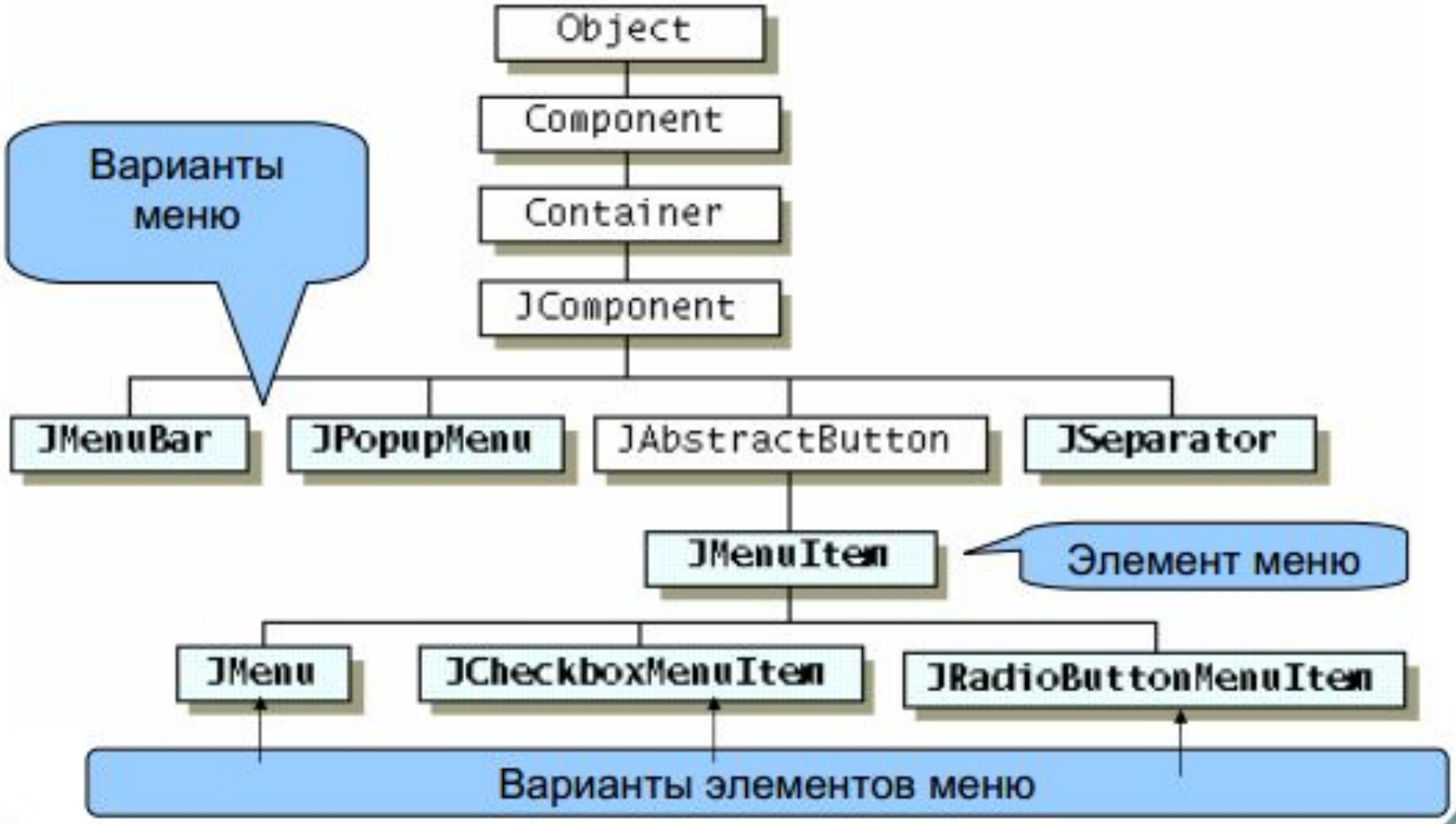
Меню – класс JMenu

- Меню – еще одна возможность выбора одного значения из многих
- Обычно элементы меню обозначают некоторые действия
- Меню, в отличие от других графических элементов, располагается особо – в верхней части окна

Меню – сопутствующие классы

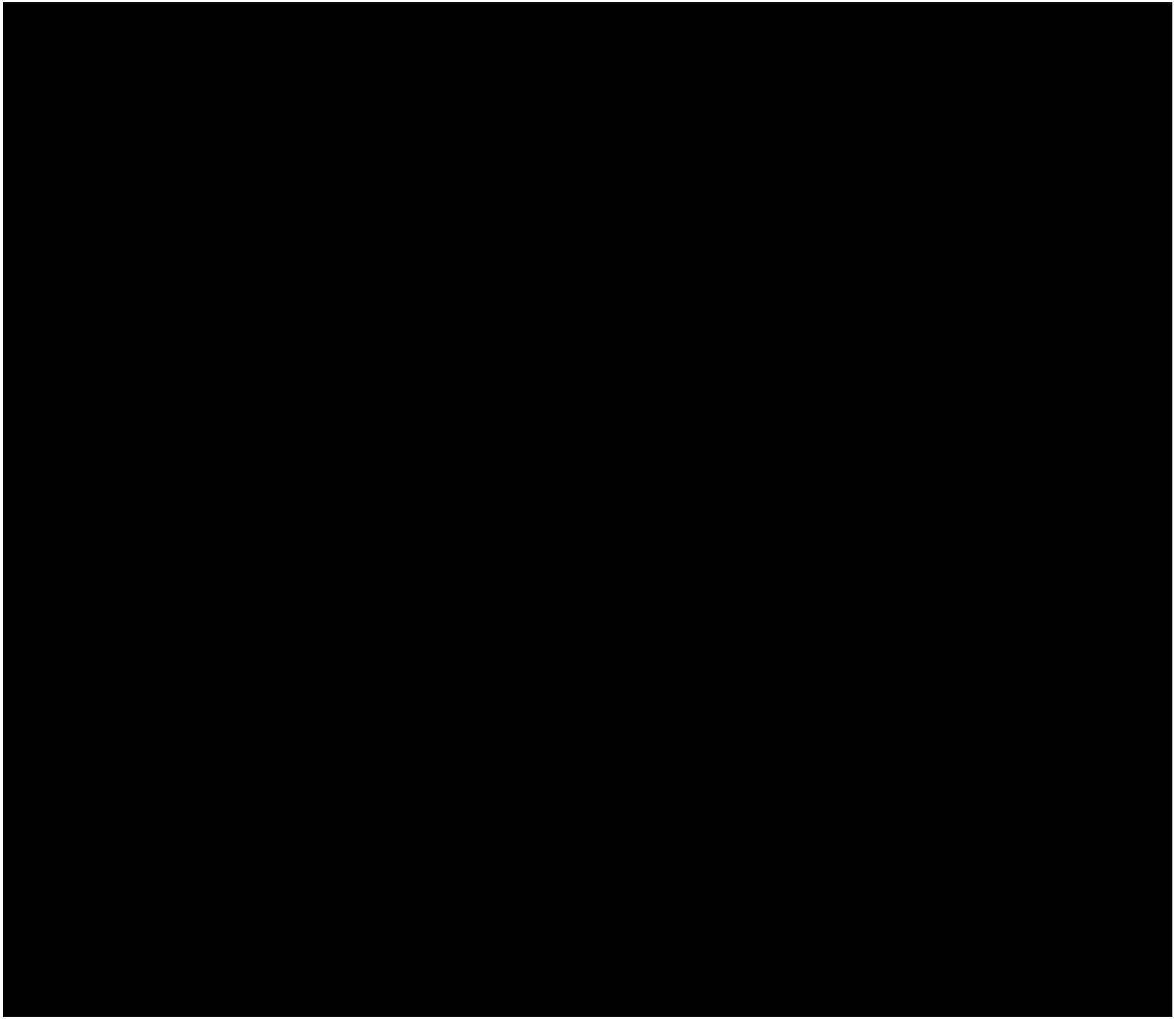
- Класс JMenuBar – полоска элементов, каждый из которых – это меню
- Класс JPopupMenu – раскрывающееся вниз ("спадающее") меню, элементы которого предоставляют возможность выбора

JMenu -- иерархия элементов

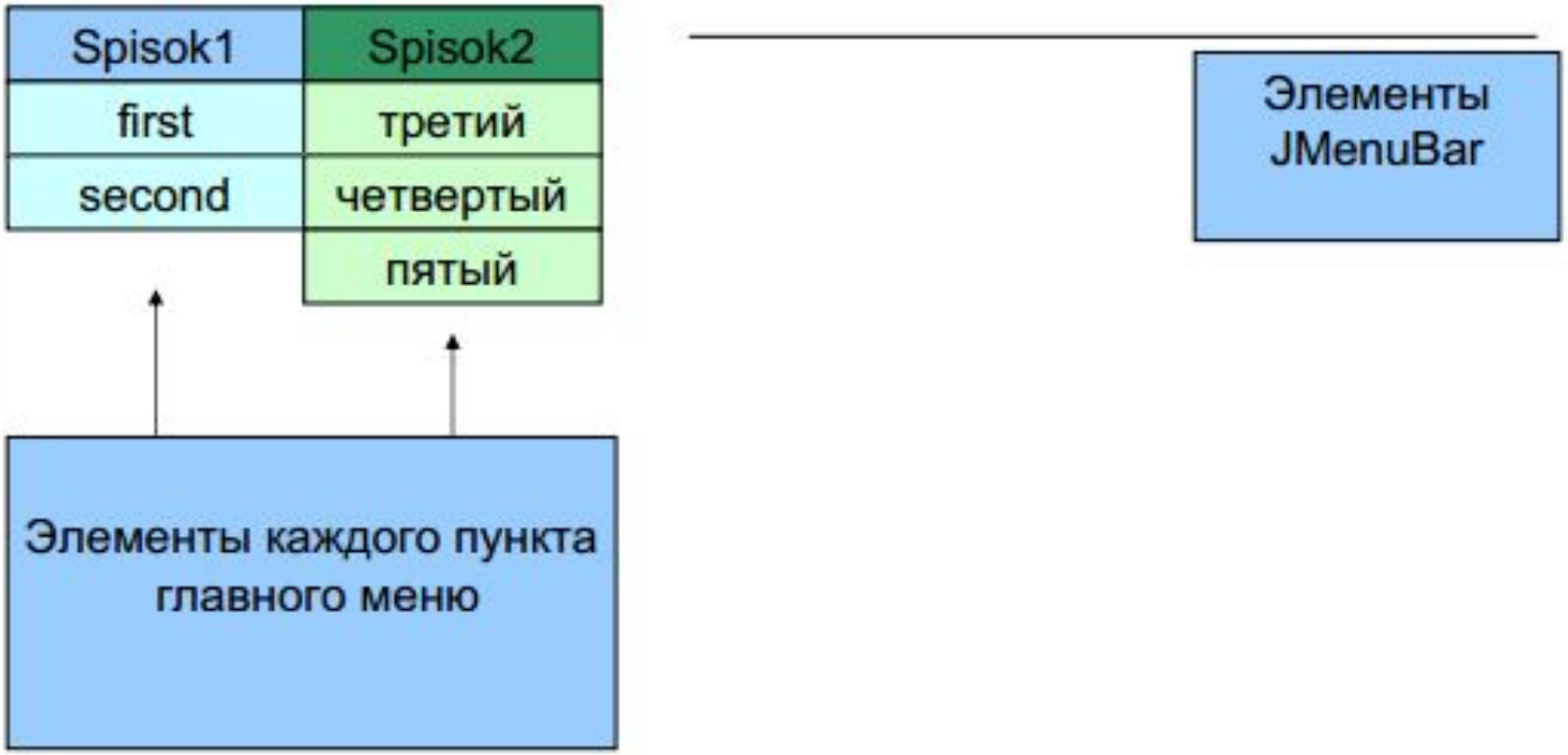


Меню

- По существу, каждый элемент типа JMenu – это кнопка, при нажатии на которую раскрывается некоторое JPopupMenu
- Несколько элементов JMenu, расположенных по горизонтали – это JMenuBar – набор кнопок, объединенных в группу



Пример – создание меню



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

public class GUI_Menu1 extends JComponent
        implements MenuListener, ActionListener {
    JMenu m1, m2;
    JMenuBar mb1;
    public GUI_Menu1(){ //Конструктор
        m1 = new JMenu("Пункт 1");
        m2 = new JMenu("Пункт 2");
        JMenuItem first = new JMenuItem("First");
        m1.add(first);
        first.addActionListener(this);
        JMenuItem second = new JMenuItem("Second");
        m1.add(second);
        second.addActionListener (this);
```

```
JMenuItem третий = new JMenuItem("Третий");
m2.add(третий);
третий.addActionListener(this);
JMenuItem четвертый = new JMenuItem("Четвертый");
m2.add(четвертый); //без "ушей"

m2.addSeparator(); //разделяем подпункты
                   // горизонтальной чертой

JMenuItem пятый = new JMenuItem("Пятый");
пятый.setEnabled(false);
m2.add(пятый); //без "ушей"
m1.addMenuListener(this); //добавляем слушателей
m2.addMenuListener(this);  //пунктов меню
mb1 = new JMenuBar(); //строка меню
mb1.add(m1);  // добавляем пункты
mb1.add(m2);  //в строку меню
}
```

Пояснения к конструктору (1)

- Для создания пунктов главного меню применен конструктор JMenu с параметром – названием пункта
- Для создания подпункта меню применен конструктор JMenuItem
- Подпункт добавляется к пункту (add…)
- Для подпункта регистрируется слушатель события ActionEvent
- Создается элемент JMenuBar и к нему добавляются два пункта меню

```
// действия для пунктов меню – пока «заглушки» (лучше оформить
// в виде отдельных методов, которые будут вызываться из
// обработчиков событий (будет более читабельный код)
private void action_1 () {
  System.out.println ("Выполнение программы для подпункта 1 пункта 1");
  System.out.println();
}
private void action_2 () {
  System.out.println ("Выполнение программы для подпункта 2 пункта 1");
  System.out.println();
}
private void action_3 () {
  System.out.println ("Выполнение программы для подпункта 1 пункта 2");
  System.out.println();
}
private void action_4 () {
  System.out.println ("Выполнение программы для подпункта 2 пункта 2");
  System.out.println();
}
private void action_5 () {
  System.out.println ("Выполнение программы для подпункта 3 пункта 2");
  System.out.println(); }
```

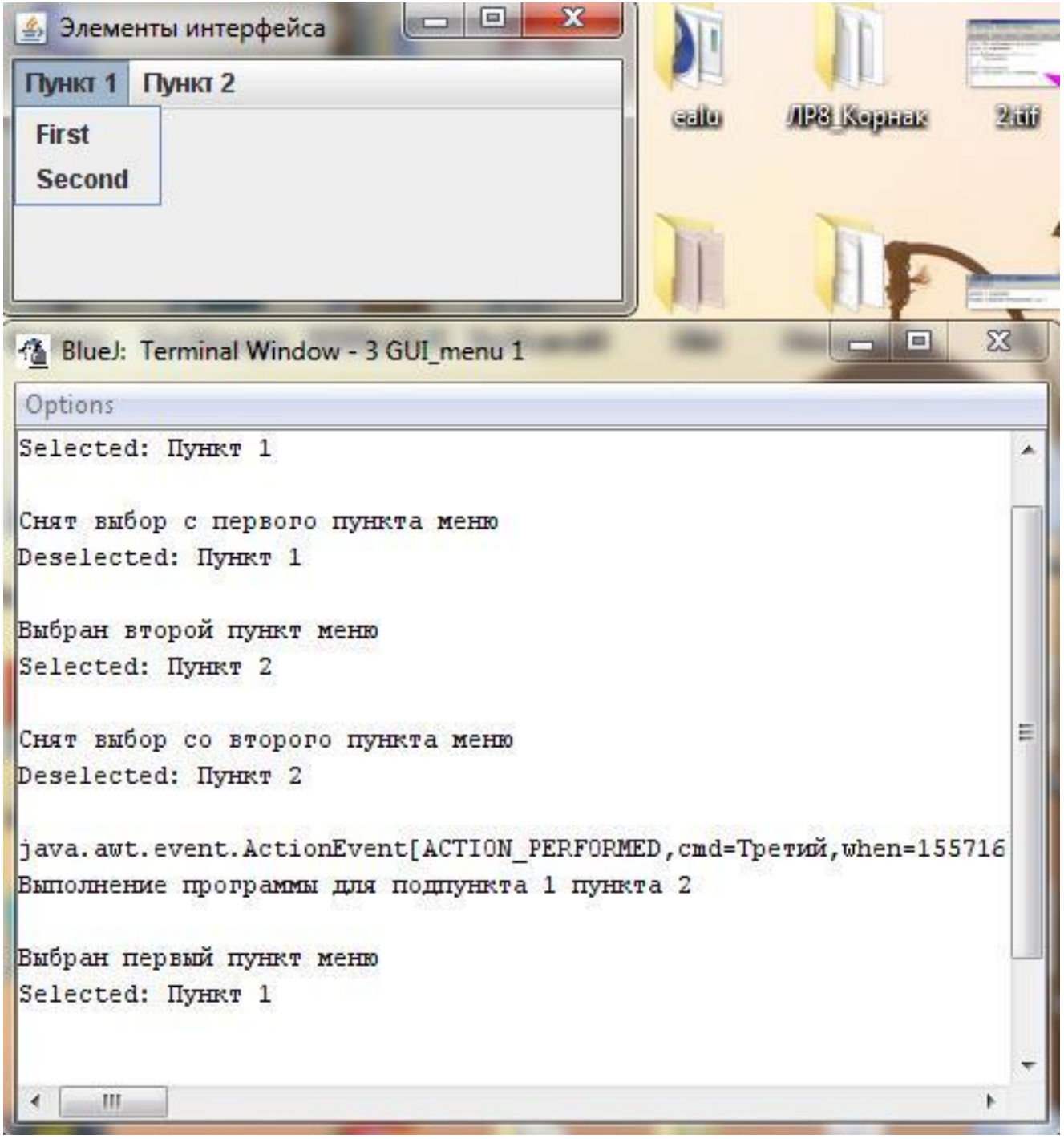
```
public void actionPerformed (ActionEvent e){
    // обработчик события подпункта меню
    System.out.println(e);
    // по умолчанию название команды равно
    // тексту элемента (подпункта меню)
    if ("First".equals(e.getActionCommand())) action_1();
    else if ("Second".equals(e.getActionCommand())) action_2();
    else if ("Третий".equals(e.getActionCommand())) action_3();
    else if ("Четвертый".equals(e.getActionCommand()))
                                                    action_4();
    else  action_5(); //четвертый и пятый не сработают –
                      // без "ушей"
}
```

```
  //обработчики событий пункта меню
  public void menuCanceled(MenuEvent e){
     System.out.println("Canceled");
     System.out.println();
  }
  public void menuDeselected(MenuEvent e){
    JMenu c = (JMenu) e.getSource();
    if (c==m1)System.out.println ("Снят выбор с первого пункта меню");
    else System.out.println ("Снят выбор со второго пункта меню");
    System.out.println("Deselected: "+ c.getText());
    System.out.println();
  }
  public void menuSelected(MenuEvent e){
    JMenu c = (JMenu) e.getSource();
    if (c==m1)System.out.println ("Выбран первый пункт меню");
    else System.out.println ("Выбран второй пункт меню");
    System.out.println("Selected: "+ c.getText());
    System.out.println();
  }
} // GUI_Menu1
```

Пояснения к обработчикам

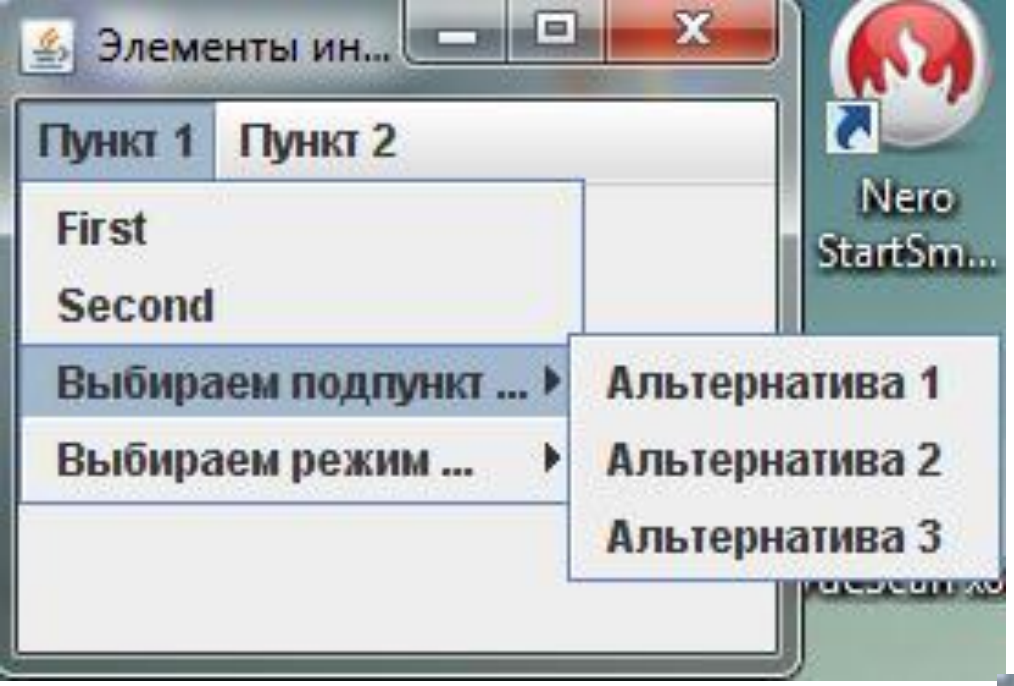
- при изменении выбора в главном меню возникают события (в окне терминала Selected, Deselected)
- при выборе пункта меню возникает такое же событие, как и при нажатии кнопки. По умолчанию "команда", связанная с пунктом, равна тексту пункта
- по команде обработчик может выполнить нужное действие

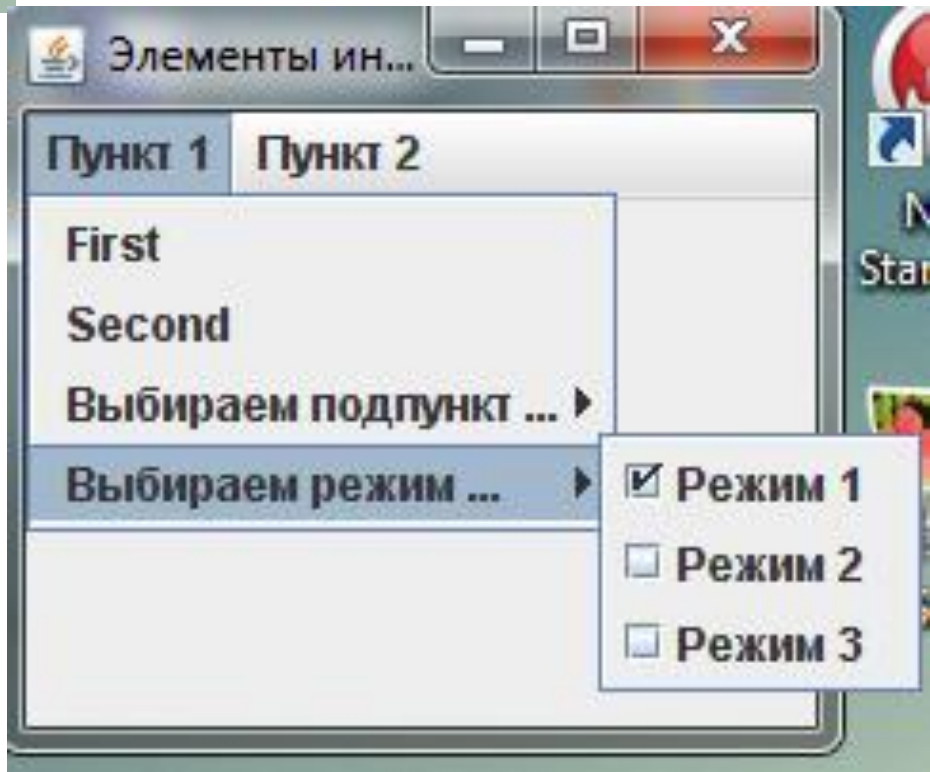
```
import java.awt.*;
import javax.swing.*;
public class MyFrame{
   private static void createAndShowGUI(){
      JFrame frame = new JFrame("Элементы интерфейса");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       GUI_Menu1 s = new GUI_Menu1();
      frame.setJMenuBar(s.mb1);
      frame.setSize(300,170);
      frame.setLocation(10,10);
      frame.setVisible(true);
  }
  public static void main (String[ ] args){
      javax.swing.SwingUtilities.invokeLater(new Runnable(){
          public void run(){createAndShowGUI();}});
  }}
```


Элементы интерфейса
Пункт 1
Пункт 2
First
Second
ealu
ЛР8_Корнак
2.tif
BlueJ: Terminal Window - 3 GUI_menu 1
Options
Selected: Пункт 1
Снят выбор с первого пункта меню
Deselected: Пункт 1
Выбран второй пункт меню
Selected: Пункт 2
Снят выбор со второго пункта меню
Deselected: Пункт 2
java.awt.event.ActionEvent[ACTION_PERFORMED,cmd=Третий,when=155716
Выполнение программы для подпункта 1 пункта 2
Выбран первый пункт меню
Selected: Пункт 1

Элементом меню может быть меню.
Допишем (в конце) в конструкторе класса GUI_Menu1 следующие операторы:

```
//Проект - 4 GUI_menu 2
JMenu change1 = new JMenu("Выбираем подпункт ..."); //создали новое меню
m1.add(change1); //добавили его в качестве пункта в меню m1
JMenuItem alter1 = new JMenuItem("Альтернатива 1"); //добавляем подпункт
change1.add(alter1);
JMenuItem alter2 = new JMenuItem("Альтернатива 2"); //добавляем подпункт
change1.add(alter2);
JMenuItem alter3 = new JMenuItem("Альтернатива 3"); //добавляем подпункт
change1.add(alter3);

JMenu change2 = new JMenu("Выбираем режим ..."); //создали новое меню
m1.add(change2); //добавили его в качестве пункта в меню m1
//создаем подпункты
JCheckBoxMenuItem mode1 = new JCheckBoxMenuItem("Режим 1",true);
change2.add(mode1);
JCheckBoxMenuItem mode2 = new JCheckBoxMenuItem("Режим 2");
change2.add(mode2);
change2.add(new JCheckBoxMenuItem("Режим 3"));
```


Элементы ин...
Пункт 1
Пункт 2
First
Second
Выбираем подпункт ...
Выбираем режим ...
Альтернатива 1
Альтернатива 2
Альтернатива 3
Nero
StartSm...


Элементы ин...
Пункт 1
Пункт 2
First
Second
Выбираем подпункт ...
Выбираем режим ...
Режим 1
Режим 2
Режим 3

Всплывающее меню JPopupMenu

- "раскрывает" свои пункты вниз, если в главном меню
- "раскрывается" вправо, если не в главном меню
- обычно используется для вывода контекстной помощи (по правой кнопке мышки)

JTable

JTable – компонент ГИП, позволяющий организовать данные в виде таблицы.
https://jakeroid.com/ru/blog/primeryi-ispolzovaniya-jtable.html

Разработчики *JTable* создали его по парадигме «модель-вид-контроллер».

Другими словами, это разделило *JTable* на части таким образом, что одна отвечает за способы отображения информации, другая за внешний вид, третья за получение данных.

За организацию вывода данных из какой-либо структуры в таблицу отвечает модель таблицы.

Самый простой способ создания *JTable* — на основе массива. Конструктору передается массив названий столбцов и массив значений. Все остальное таблица делает сама.

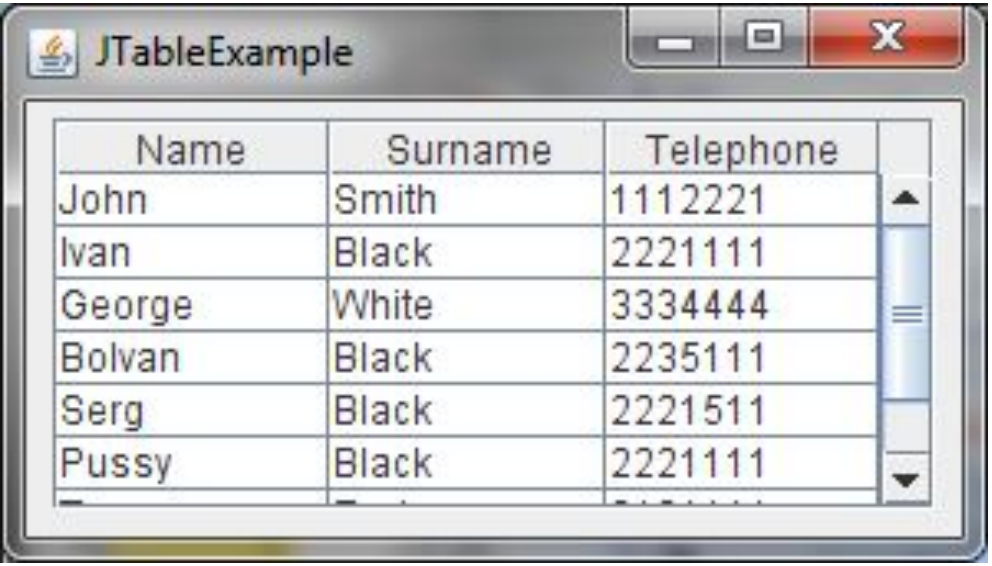
```
import java.awt.Dimension; //Проект Использование JTable 1
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.SwingUtilities;

public class JTableExample {
   //Массив содержащий заголовки таблицы
   Object[ ] headers = { "Name", "Surname", "Telephone" };
    //Массив (двумерный), содержащий информацию для таблицы
   Object[ ][ ] data = {
      { "John", "Smith", "1112221" },
      { "Ivan", "Black", "2221111" },
      { "George", "White", "3334444" },
      { "Bolvan", "Black", "2235111" },
      { "Serg", "Black", "2221511" },
      { "Pussy", "Black", "2221111" },
      { "Tonya", "Red", "2121111" },
      { "Elise", "Green", "2321111" },
   };
   //Ссылка на объект-таблицу
   Jtable  jTabPeople;
```

Пример 1

```
JTableExample() { //конструктор
    //Создаем новый контейнер JFrame
    JFrame jfrm = new JFrame("JTableExample");
    //Устанавливаем диспетчер компоновки
    jfrm.getContentPane().setLayout(new FlowLayout());
    //Устанавливаем размер окна
    jfrm.setSize(300, 170);
    //Устанавливаем завершение программы при закрытии окна
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    //Создаем новую таблицу на основе двумерного массива данных
    // и одномерного массива заголовков
    jTabPeople = new JTable(data, headers);
    //Создаем панель прокрутки и включаем в ее состав таблицу
    JScrollPane jscrlp = new JScrollPane(jTabPeople);
    //Устаналиваем размеры прокручиваемой области
    jTabPeople.setPreferredScrollableViewportSize(new Dimension(250, 100));
    //Добавляем в контейнер панель прокрутки
    //и таблицу вместе с ней
    jfrm.getContentPane().add(jscrlp);
    //Отображаем контейнер
    jfrm.setVisible(true); }
```

Внимание! Заголовок таблицы отображается только если для нее создана панель прокрутки .

```
//метод main, запускающийся при старте приложения
   public static void main(String[ ] args) {
      //Создаем фрейм в потоке обработки событий
      SwingUtilities.invokeLater(new Runnable() {
          public void run() {
             new JTableExample();
          }
      });  } }
```



Все ячейки будут редактируемыми (используется модель по умолчанию), измененные данные будут вводится в ячейку при выборе другой ячейки (событие)

В рассмотренной программе сначала создаются два массива. Один содержит заголовки, второй данные для таблицы. Массив с данными, в каждой строке должен содержать такое же количество элементов, как и массив с заголовками. Таблица создается конструктором, принимающим два элемента, в качестве которых выступают указанные массивы. При этом таблица создает модель данных сама. Потом мы создаем *JScrollPane*, которая нужна для возможности прокрутки содержимого таблицы. Добавляем таблицу на объект *JScrollPane*, а его на панель контента фрейма. Отображаем окно.

Но использование массивов для задания значений неудобно. Во первых данные могут лежать в базе данных, тогда будет необходимо их преобразовать в массив. Это неудобно и неэффективно. Кроме того, при изменении данных, придется заново создавать таблицу, с новыми массивами.

Решением этой проблемы является создание своей модели таблицы. Модель таблицы можно создать разными способами.

Можно пойти более сложным путем, создав класс, реализующий интерфейс *TableModel*. Тогда придется описывать все необходимые методы интерфейса и разбираться, как они работают. Кроме того, нужно будет хранить слушателя событий для модели данных.

Рассмотрим более простой способ.

Класс *AbstractTableModel* содержит реализацию всех методов интерфейса *TableModel* за исключением

***getValueAt()*, *getRowCount()* и *getColumnCount()*.**

Создадим свой класс, унаследовав его от *AbstractTableModel* и реализуем эти методы.

Пример 2.

Данные о людях предполагается хранить в таблице

```
//Проект Использование JTable
public class Human { // Сущность - человек
    private String name;
    private String surname;
    private String telephone;
    public Human(String name, String surname, String telephone) {
        this.name = name;
        this.surname = surname;
        this.telephone = telephone;
    }
    public String getName() {
        return name;
    }
    public String getSurname() {
        return surname;
    }
    public String getTelephone() {
        return telephone;
    }
}
```

```
import java.util.*;
import javax.swing.table.AbstractTableModel;
public class MyTableModel extends AbstractTableModel{
   ArrayList <Human> humans; //переменная
                                    // экземпляра
   public MyTableModel(ArrayList <Human> humans) {
      //конструктор
      super();
      this.humans = humans;
   }
   @Override
   public int getRowCount() {
      //возвращает число строк в таблице
      return humans.size();
   }
   @Override
   public int getColumnCount() {
      // возвращает число колонок в таблице
      return 3;  }
```

Внимание! Источником данных для таблицы теперь является не массив, а список объектов класса Human.

```
@Override
public String getColumnName(int c) {
    //возвращает название колонки с номером c
        String result = "";
        switch (c) {
            case 0:
                result = "Name";
                break;
            case 1:
                result = "Surname";
                break;
            case 2:
                result = "Telephone";
                break;
        }
        return result;  }
```

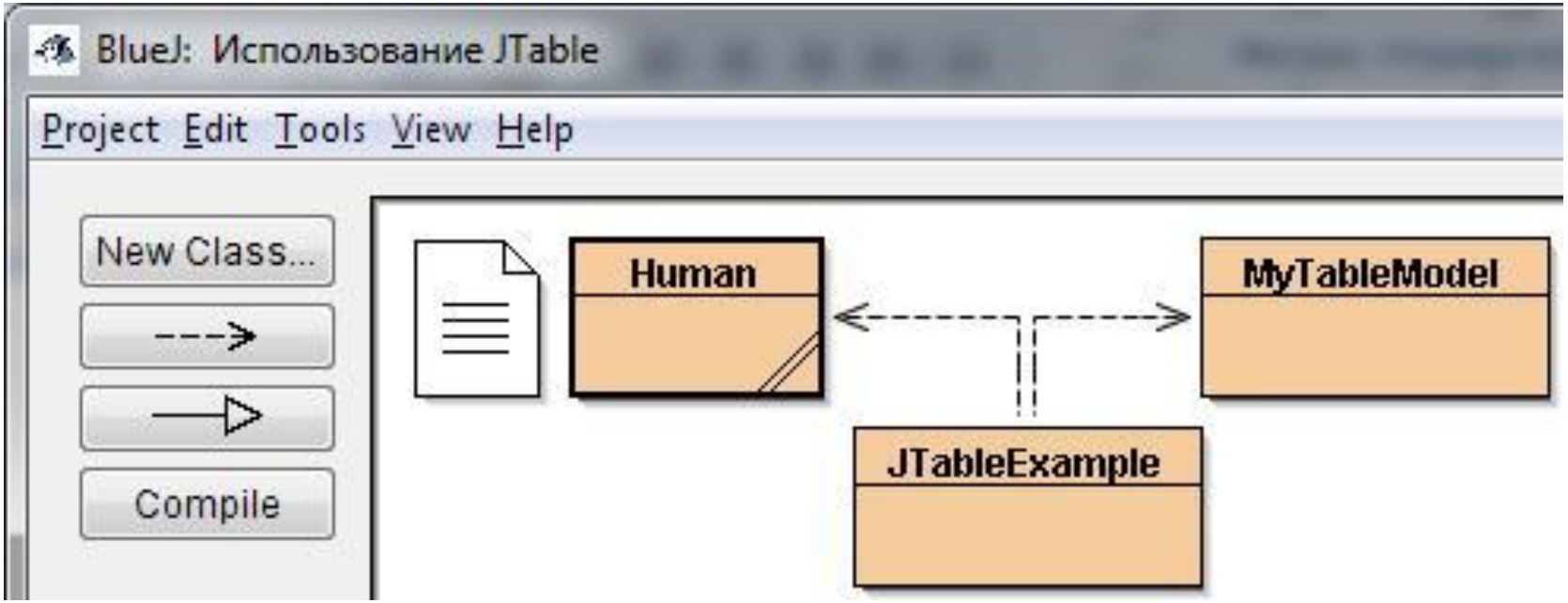
```
@Override
public Object getValueAt(int r, int c) {
        //возвращает значение в колонке с строки r
        switch (c) {
            case 0:
                return humans.get(r).getName();
            case 1:
                return humans.get(r).getSurname();
            case 2:
                return humans.get(r).getTelephone();
            default:
                return "";
        }
    }
} // MyTableModel
```

```
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.event.*;

public class JTableExample {

    JTable  jTabPeople; //ссылка на таблицу
    ArrayList <Human>  humans; //ссылка на источник данных
    MyTableModel   tModel ; // ссылка на модель таблицы

    JTableExample() { //конструктор
        //Создаем новый контейнер JFrame
        JFrame jfrm = new JFrame("JTableExample");
        //Устанавливаем диспетчер компоновки
        jfrm.getContentPane().setLayout(new FlowLayout());
        //Устанавливаем размер окна
        jfrm.setSize(300, 300);
        //Устанавливаем завершение программы при закрытии окна
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
//Создаем новую таблицу на основе списка объектов
// класса Human:
//1.Создадим список из сущностей класса Human
 humans = new ArrayList <Human> ();
 humans.add(new Human("John", "Smith", "1231231"));
 humans.add(new Human("George", "White", "321321312"));
 humans.add(new Human("Olga", "Bregneva", "7171711"));
 //2.Создадим модель таблицы
 tModel = new MyTableModel (humans);
 //3. На основе модели, создадим новую JTable
 jTabPeople = new JTable(tModel);
 //4.Создаем панель прокрутки и включаем в ее состав
 //таблицу
 JScrollPane jscrlp = new JScrollPane(jTabPeople);
 //5.Устаналиваем размеры прокручиваемой области
 jTabPeople.setPreferredScrollableViewportSize(
                                        new Dimension(250, 100));
 //Добавляем в контейнер панель прокрутки
 //и таблицу вместе с ней
 jfrm.getContentPane().add(jscrlp);
```

```
    //Создаем кнопку
    JButton btnPress = new JButton("Click!");
     // Добавляем к источнику (кнопке) анонимного
     // слушателя событий
    btnPress.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent  ae) {
            //добавить человека в список:
            humans.add(new Human("Elena", "Ivanova", "12300123"));
            //изменить данные в таблице
            tModel.fireTableDataChanged();
        }
    });
    //добавить кнопку на панель контента фрейма:
    jfrm. getContentPane().add(btnPress);
    //отобразить фрейм:
    jfrm.setVisible(true);
}
```

```
//метод main, запускающийся при старте приложения
    public static void main(String[ ] args) {
        //Создаем фрейм в потоке обработки событий
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new JTableExample();
            }
        });
    }
}
```



После запуска

После 1 клика

После 4 кликов

JTableExample

Name	Surname	Telephone
John	Smith	1231231
George	White	321321312
Olga	Bregneva	7171711

Click!

JTableExample

Name	Surname	Telephone
John	Smith	1231231
George	White	321321312
Olga	Bregneva	7171711
Elena	Ivanova	12300123

Click!

JTableExample

Name	Surname	Telephone
John	Smith	1231231
George	White	321321312
Olga	Bregneva	7171711
Elena	Ivanova	12300123
Elena	Ivanova	12300123
Elena	Ivanova	12300123

Click!

Ячейки утратили возможность редактирования! Что делать?

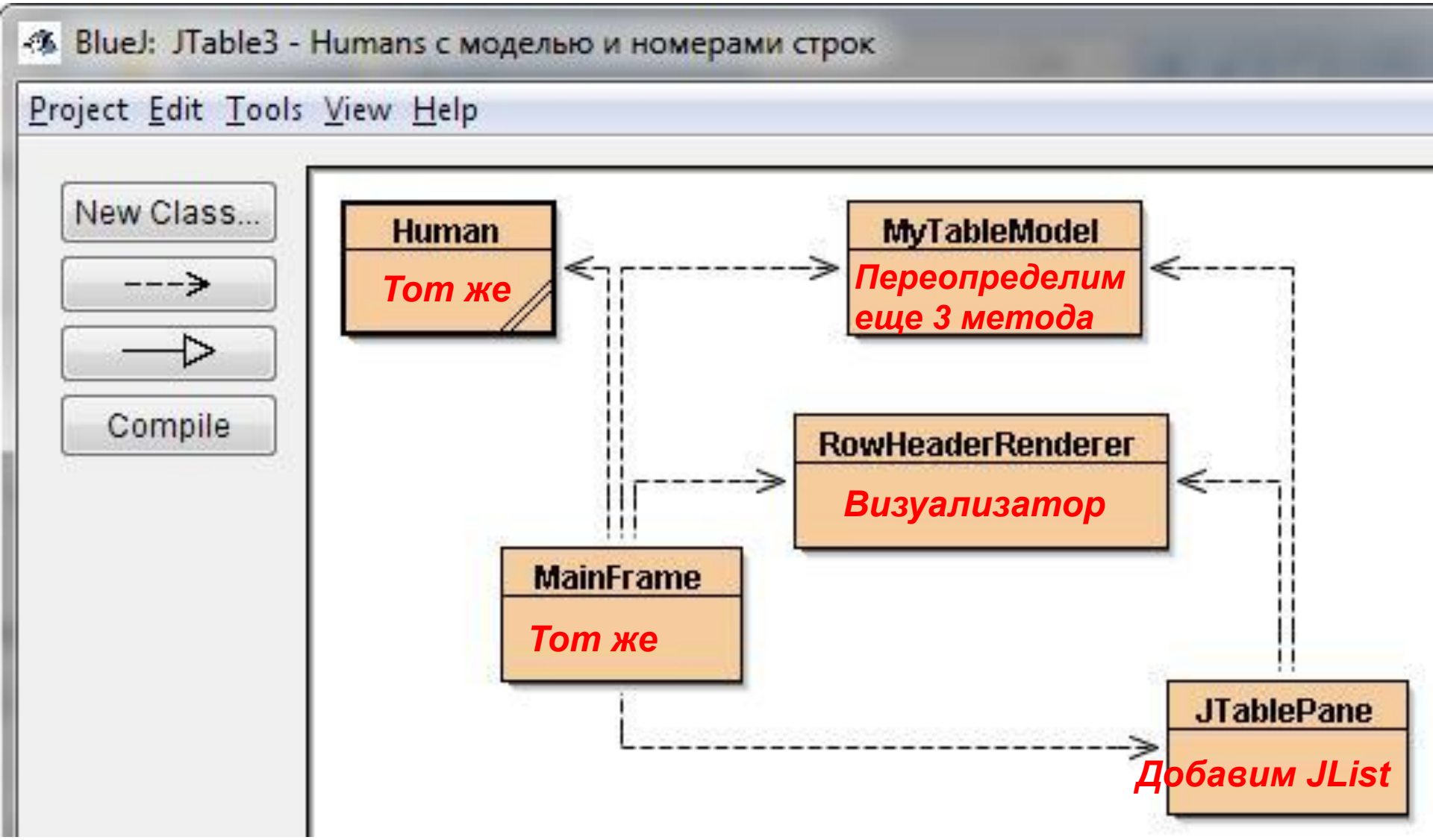
Чтобы в таблицу вставлялись разные люди, нужно брать данные, например, из JTextField.

1) **Ячейки утратили возможность редактирования, т. к. мы теперь используем собственную модель таблицы, в которой не предусмотрели эту возможность.**

Нужно переопределить методы
public Class<?> getColumnClass(int c)
public boolean isCellEditable(int r, int c) **и**
public void setValueAt(Object value, int r,int c)
класса AbstractTableModel для нашей модели.

2) **Как сделать, чтобы для таблицы выводились не только названия столбцов, но и названия строк (например, номера строк)?**

Нужно задать список (Jlist) для названий строк и визуализатор элемента (Jlabel) для этого списка.


BlueJ: JTable3 - Humans с моделью и номерами строк
Project Edit Tools View Help
New Class...
Compile
Human
Тот же
MyTableModel
Переопределим
еще 3 метода
RowHeaderRenderer
Визуализатор
MainFrame
Тот же
JTablePane
Добавим JList

```
import java.util.*;
import javax.swing.table.AbstractTableModel;
public class MyTableModel  extends AbstractTableModel {

    ArrayList <Human> humans; //переменная  экземпляра
    public MyTableModel(ArrayList <Human> humans) {
        //конструктор
        super();
        this.humans = humans;
    }
    @Override
    public int getRowCount() {
        //возвращает число строк в таблице
        return humans.size();
    }
    @Override
    public int getColumnCount() {
        // возвращает число колонок в таблице
        return 3;
     }
```

```
@Override
 public String getColumnName(int c) {
//возвращает название колонки с номером c
    String result = "";
    switch (c) {
        case 0:  result = "Имя"; break;
        case 1:  result = "Фамилия"; break;
        case 2:  result = "Телефон"; break;
    }
    return result;  }

  @Override
  public Object getValueAt(int r, int c) {
    //возвращает значение в колонке с строки r
    switch (c) {
        case 0:  return humans.get(r).getName();
        case 1:  return humans.get(r).getSurname();
        case 2:  return humans.get(r).getTelephone();
        default:  return "";
    }
}
```

```
public Class<?> getColumnClass(int c){
    //Возвращает класс столбца
    return String.class;
    // Для String редактор по умолчанию - JTextField
  }
public boolean isCellEditable(int r, int c){
    //Возвращает  true, если значение в ячейке
    // можно редактировать
    return true;
}
public void setValueAt (Object value, int r, int c){
    // Устанавливает в ячейке, находящейся
    //в столбце с строки r значение value
    String str = (String)value;
    switch(c){
      case 0:  humans.get(r).setName(str); break;
      case 1:  humans.get(r).setSurname(str); break;
      case 2:  humans.get(r).setTelephone(str);   break;
    }
  } } // MyTableModel
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.*;
public class RowHeaderRenderer extends JLabel
                                    implements ListCellRenderer {
  //элемент-визуализатор для списка заголовков строк
  RowHeaderRenderer (JTable table){ //конструктор
    //Получаем ссылку на объект, управляющий  заголовком JTable
    JTableHeader header = table.getTableHeader();
    setOpaque(true); //непрозрачный фон
    setBorder(UIManager.getBorder("TableHeader.cellBorder"));
    //Класс UIManager управляет текущим внешним видом,
    // набором доступных вариантов внешнего вида
    setHorizontalAlignment(CENTER);
    setForeground(header.getForeground());
    setBackground(header.getBackground());
    setFont(header.getFont());
  }
  public Component getListCellRendererComponent( JList list, Object value,
                          int index,  boolean isSelected, boolean cellHasFocus) {
    setText ((value == null) ? "" : value.toString());
    return this;
  } } // RowHeaderRenderer
```

```
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.event.*;

public class JTablePane extends JPanel{
  //Сущность - панель с визуализированной таблицей
   JTable  jTab; //Ссылка на объект-таблицу
   ArrayList<Human>  human; //ссылка на список людей
                                    // (для модели)
   MyTableModel   tModel ; //ссылка на модель для JTable
   DefaultListModel lm; //ссылка на модель списка
                              //названий строк таблицы
   int rowNum = 0; //число строк в таблице

   JTablePane() { //конструктор
      //Установить рамку с названием панели:
      setBorder (
        new  javax.swing.border.TitledBorder ("База данных"));
      //Устанавливаем диспетчер компоновки
      setLayout(new BoxLayout (this, BoxLayout.Y_AXIS));   //сложит
       //элементы один под другим (в линию по оси Y)
```

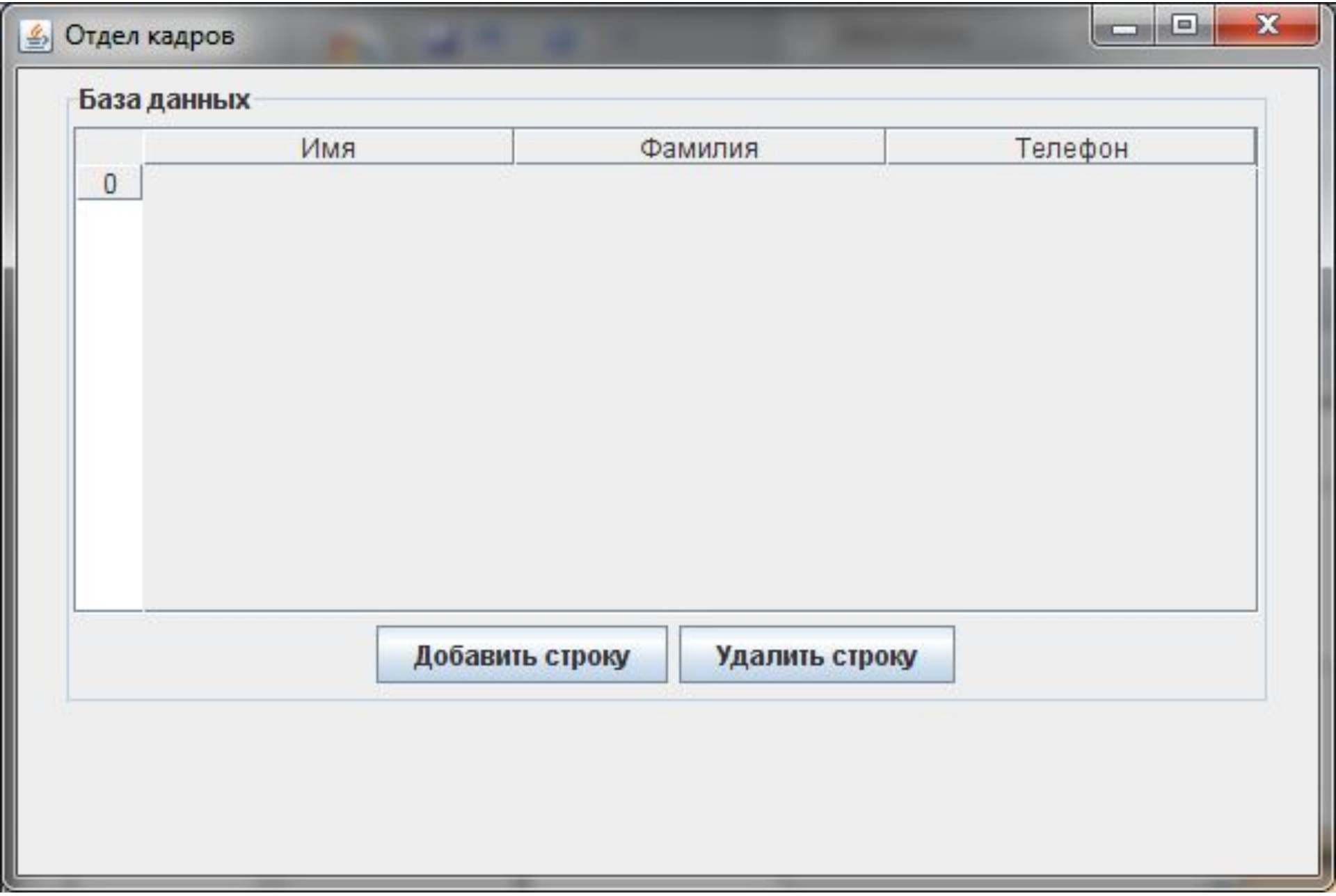
```
//1.Создадим список из объектов класса Human
human = new ArrayList<Human>();
//2.Создадим модель таблицы
tModel = new MyTableModel(human);
//3. На основе модели, создадим таблицу JTable
jTab = new JTable(tModel);
//4. Создадим модель списка названий строк
lm = new DefaultListModel();
lm.addElement("0"); // добавим в модель элемент с номером 0
                    // и значением "0"
//5.Создадим список названий строк
JList rowHeader = new JList(lm);
rowHeader.setFixedCellWidth(30); //ширина элемента списка
 //6.Установим визуализатор для элементов списка
rowHeader.setCellRenderer(new RowHeaderRenderer(jTab));
//7. Создадим панель прокрутки для списка
JScrollPane scroll = new JScrollPane(jTab);
// Обеспечим вывод заголовков строк (это функция панели прокрутки)
scroll.setRowHeaderView(rowHeader);
//8.Установим размеры прокручиваемой области
jTab.setPreferredScrollableViewportSize(new Dimension(500, 200));
//9.Добавим в контейнер панель прокрутки и таблицу вместе с ней
add(scroll);
```

```
//Создаем панель, на которой будут кнопки
JPanel butPanel = new JPanel();
//Создаем кнопку для добавления строки
JButton btnAddLine = new JButton("Добавить строку");
 // Добавляем к источнику (кнопке) анонимного
 // слушателя событий
btnAddLine.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent  ae) {
        //добавить объект в список:
        human.add(new Human("", "", ""));
        //изменить данные в таблице
        tModel.fireTableDataChanged();
        //добавить заголовок строки:
        lm.addElement(String.format("%d", ++rowNum));
}});
//Создаем кнопку для удаления строки (последней)
JButton btnDelLine = new JButton("Удалить строку");
 // Добавляем к источнику (кнопке) анонимного
 // слушателя событий
```

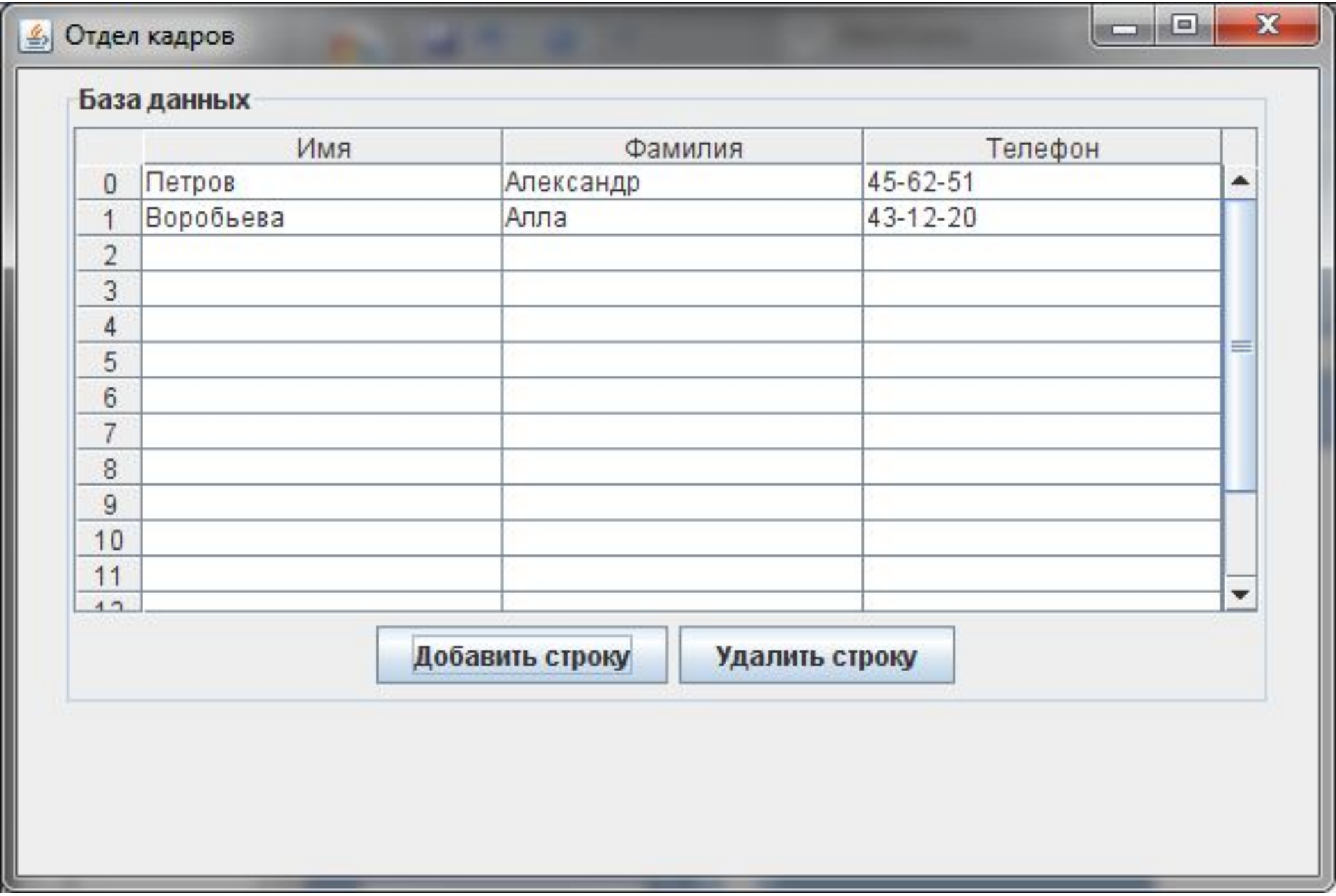
```
btnDelLine.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent  ae) {
          //удалить последний элемент из списка
          if (rowNum > 0){
             rowNum--;
             human.remove(rowNum);
             //изменить данные в таблице
             tModel.fireTableDataChanged();
             //удалить заголовок строки
             lm.removeElementAt(rowNum+1);
          }
      }});
      //добавить кнопки на панель кнопок:
      butPanel.add(btnAddLine);
      butPanel.add(btnDelLine);
      //добавить панель кнопок в контейнер:
      add(butPanel);
   }
}
```

```
import java.awt.*; import javax.swing.*; import java.util.*;
import java.awt.event.*; import javax.swing.event.*;
public class MainFrame{
    MainFrame() { //конструктор
        //Создаем новый контейнер JFrame
        JFrame jfrm = new JFrame("Отдел кадров");
        //Устанавливаем диспетчер компоновки
        jfrm.getContentPane().setLayout(new FlowLayout());
        //Устанавливаем размер окна
        jfrm.setSize(600, 400);
        //Устанавливаем завершение программы при закрытии окна
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JTablePane pane = new JTablePane();
        jfrm. getContentPane().add(pane);
        //отобразить фрейм:
        jfrm.setVisible(true);
    }
    //метод main, запускающийся при старте приложения
    public static void main(String[ ] args) {
        //Создаем фрейм в потоке обработки событий
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
              new MainFrame();
           }      });       } }
```

После старта приложения



После добавления строк и значений



После удаления строк

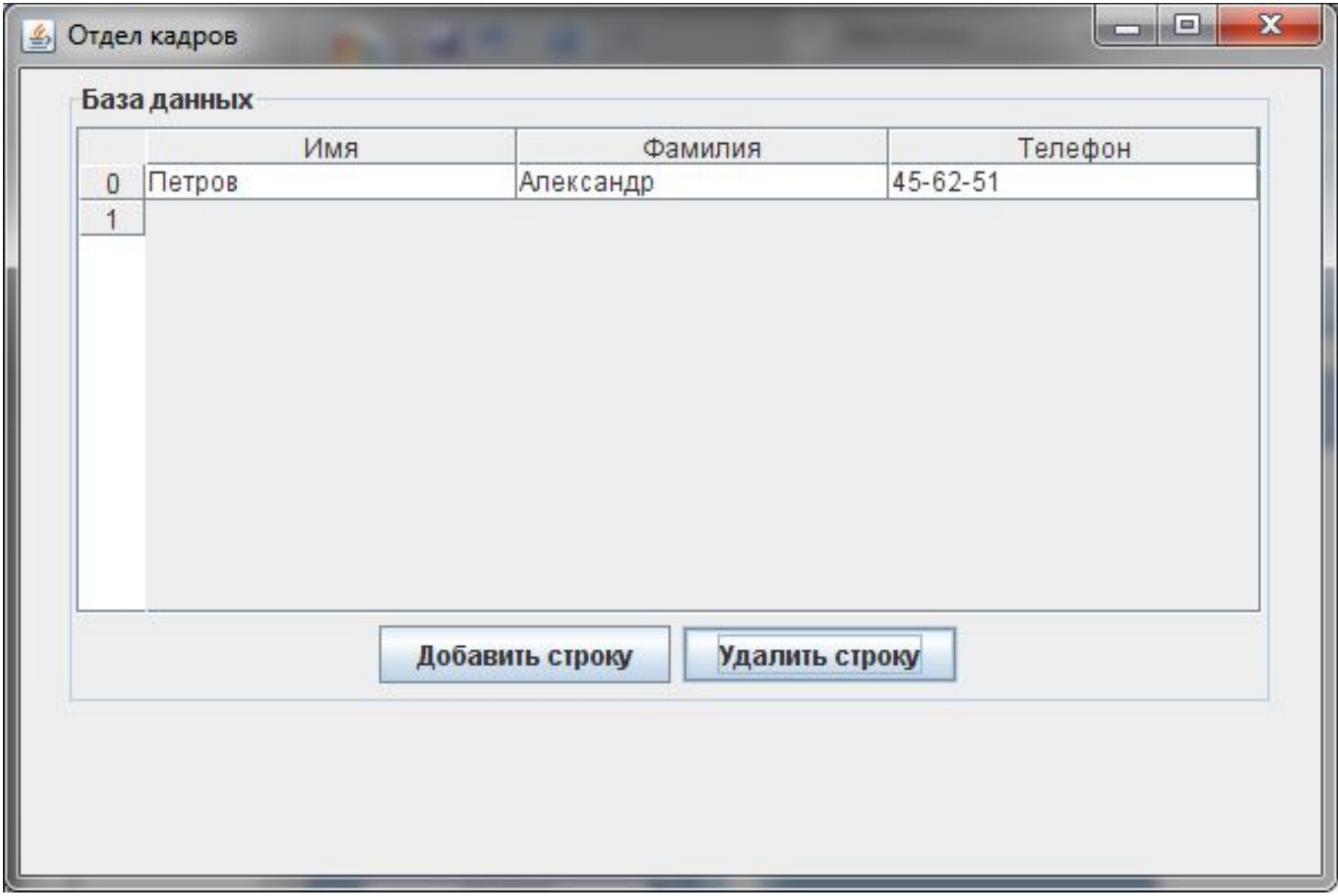


Таблица **JTable** позволяет отображать двухмерную информацию в виде строк и столбцов, настраивать и сортировать данные, выводить их в любом подходящем виде, управлять заголовками таблицы и ее выделенными элементами. **JTable** использует три модели, поставляющие ей данные и сохраняющие изменения при изменении этих данных. Первая и самая важная модель реализует интерфейс TableModel использует три модели, поставляющие ей данные и сохраняющие изменения при изменении этих данных. Первая и самая важная модель реализует интерфейс TableModel и хранит данные ячеек таблицы и дополнительную служебную информацию об этих ячейках. Вторая модель реализует интерфейс TableColumnModel использует три модели, поставляющие ей данные и сохраняющие изменения при изменении этих данных. Первая и самая важная модель реализует интерфейс TableModel и хранит данные ячеек таблицы и дополнительную служебную информацию об этих ячейках. Вторая модель реализует интерфейс TableColumnModel и управляет столбцами таблицы. TableColumnModel позволяет добавлять, перемещать и находить столбцы, узнавать об изменениях в их расположении, с ее помощью можно изменить расстояние между столбцами и находящимися в них ячейками. Третья модель таблицы SelectionModelиспользуется для выделения списка. Она управляет выделением строк таблицы и не затрагивает столбцы. Модель TableColumnModel имеет собственную модель выделения списка ListSelectionModel. Так что можно формально говорить о четырех моделях таблицы JTable. Против одна из этих моделей вызывается неявно, через другую модель. Помимо моделей таблица *JTable* предоставляет еще несколько удобных механизмов выделения строк, столбцов и ячеек.

http://java.sun.com/docs/books/tutorial/uiswing/components/table.html

Портянкин И. Swing: эффектные пользовательские интерфейсы. – М.:Издательство Улори, 2011. – 594 с.

http://ipsoftware.ru/books/swing_book_2/

Class JTextArea

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.text.JTextComponent
          javax.swing.JTextArea

Область многострочного текста с возможностью редактирования.

Конструкторы:

JTextArea()
JTextArea(int rows, int columns)
JTextArea(String text)
JTextArea(String text, int rows, int columns)
JTextArea(Document doc)
JTextArea(Document doc, String text, int rows, int columns)

В отличие от текстового поля область для ввода текста позволяет ввести не одну строку, а несколько. В связи с этим JTextArea предлагает несколько дополнительных функций.

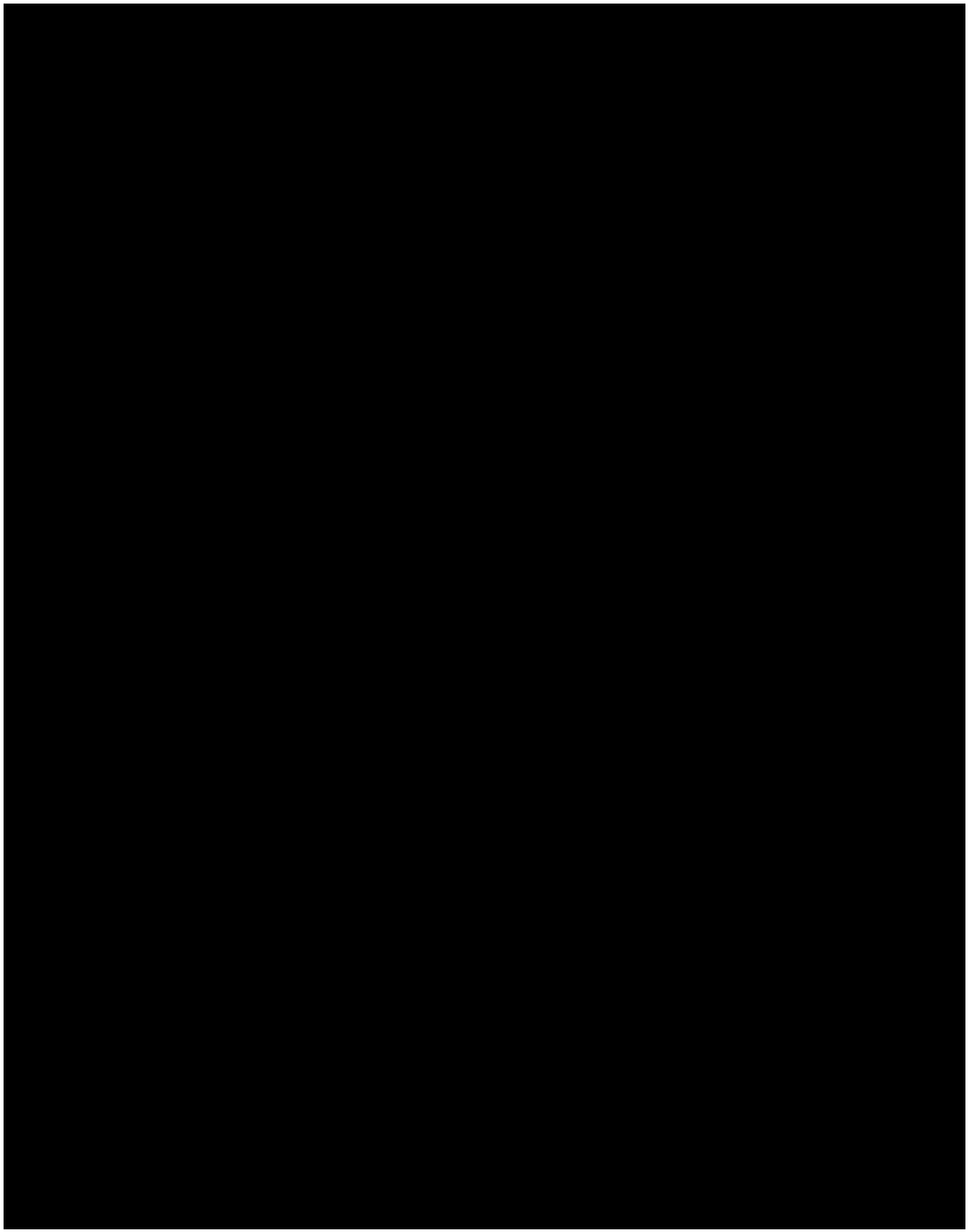
Во-первых, это способность переносить слова на соседнюю строку целиком, которой управляет метод setWrapStyleWord(boolean wrapStyle). Если вызвать этот метод с параметром true, то слова не будут разрываться в том месте, где они «натыкаются» на границу компонента, а будут целиком перенесены на новую строку.

Во-вторых, это способность переносить текст (то есть длинные строки будут укладываться в несколько строк вместо одной, уходящей за границы компонента. Этой способностью управляет метод setLineWrap(boolean lineWrap). Методы isWrapStyleWord() и isLineWrap() возвращают текущее состояние данных способностей (true — активирована и false — деактивирована).

При создании JTextArea чаще всего используют конструктор JTextArea(int rows, int columns), устанавливающий высоту (количество строк) и ширину (количество символов) компонента.

Для работы со своим содержимым JTextArea дополнительно предлагает два удобных метода. Метод append(String text) добавляет строку text в конец уже имеющегося текста, а метод insert(String text, int position) вставляет ее в позицию position.

Кроме указанных методов можно использовать методы, наследуемые от классов-предков, например, setText и setBorder.



```
//Проект - 5 GUI_текст_область
import java.awt.*; import javax.swing.*;
import java.awt.event.*;

public class TextProcess extends JPanel
                implements ActionListener{
  JLabel name; //название панели
  JTextArea  textVis; //визуализированный текст
  String textIn; //соответствующее внутреннее
                 //представление текста
  JButton but1; //кнопка "Получить текст"
  JButton but2; //кнопка "Стереть текст"
```

```
public TextProcess (){ //конструктор
    textVis = new JTextArea("");
    textVis.setBorder(new
            javax.swing.border.EtchedBorder());
    textIn = "";
    //установки для главной панели класса
    //(потомок класса "Панель")
    Dimension d1 = new Dimension(400,400);
    setPreferredSize(d1);
    setBorder(new
             javax.swing.border.LineBorder(Color.ORANGE, 4));
    setLayout(new BorderLayout(10,10));
```

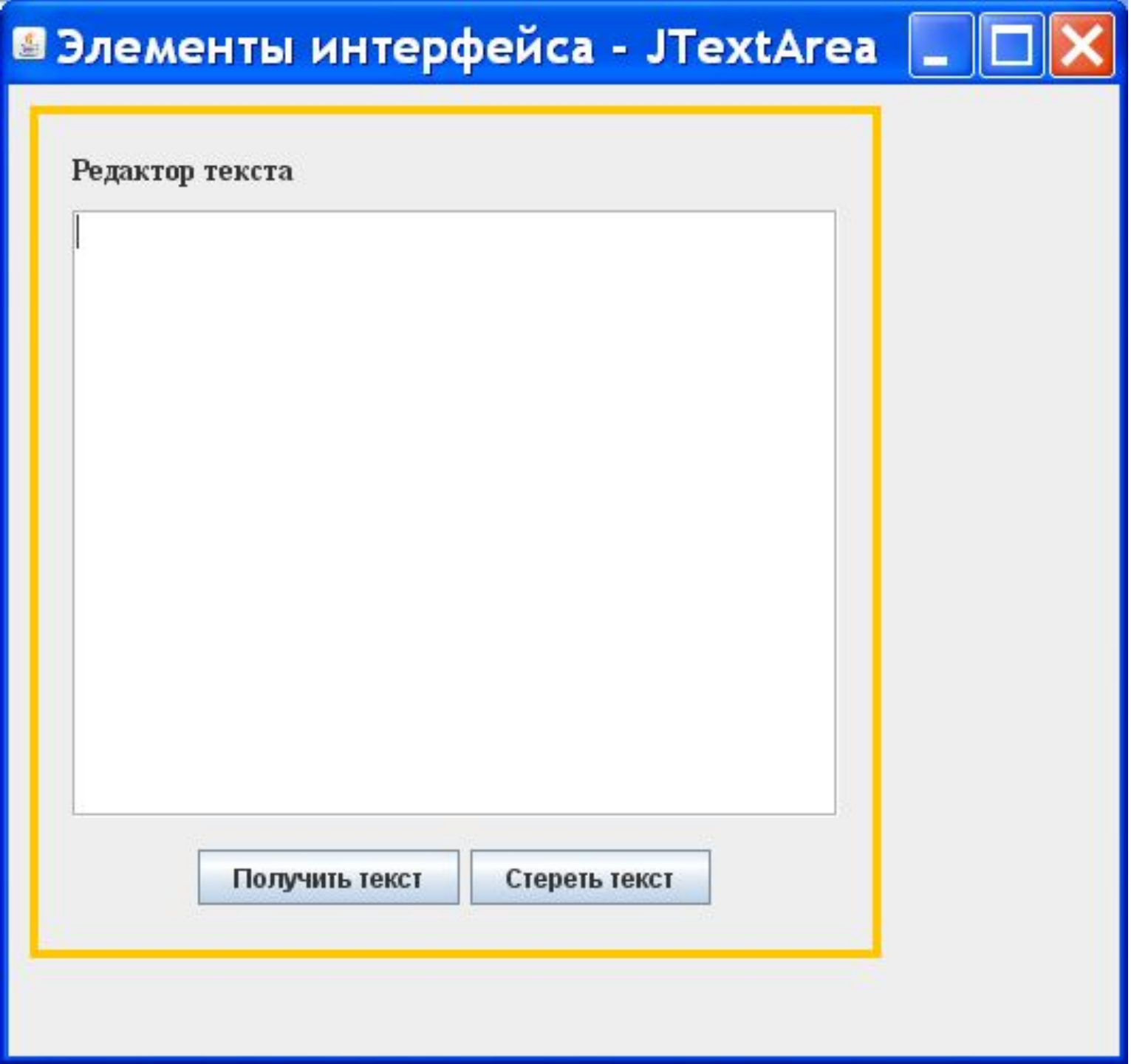
```
name = new JLabel("Редактор текста");
name.setFont(new Font("Serif", Font.BOLD,14));
add(name, BorderLayout.NORTH);

JPanel butPan = new JPanel(); //панель для кнопок

JButton but1 = new JButton ("Получить текст");
but1.setActionCommand("Получить");

JButton but2 = new JButton ("Стереть текст");
but2.setActionCommand("Стереть");

but1.addActionListener (this);
but2.addActionListener (this);

butPan.add(but1);  butPan.add(but2);
add(butPan,BorderLayout.SOUTH);
add(textVis,BorderLayout.CENTER);  }
```

```
public boolean isChange(){
   //сравнение  визуального
   //и внутреннего представления текста
    boolean f;
    f = textIn.equals(textVis.getText());
    return !f ;
 }
 public void toWork(){
    //переписать текст из  визуального
    //компонента во внутренний
    textIn = textVis.getText();
}
  public void reset(){ //стереть текст
        textIn="";
        textVis.setText("");
 }
```

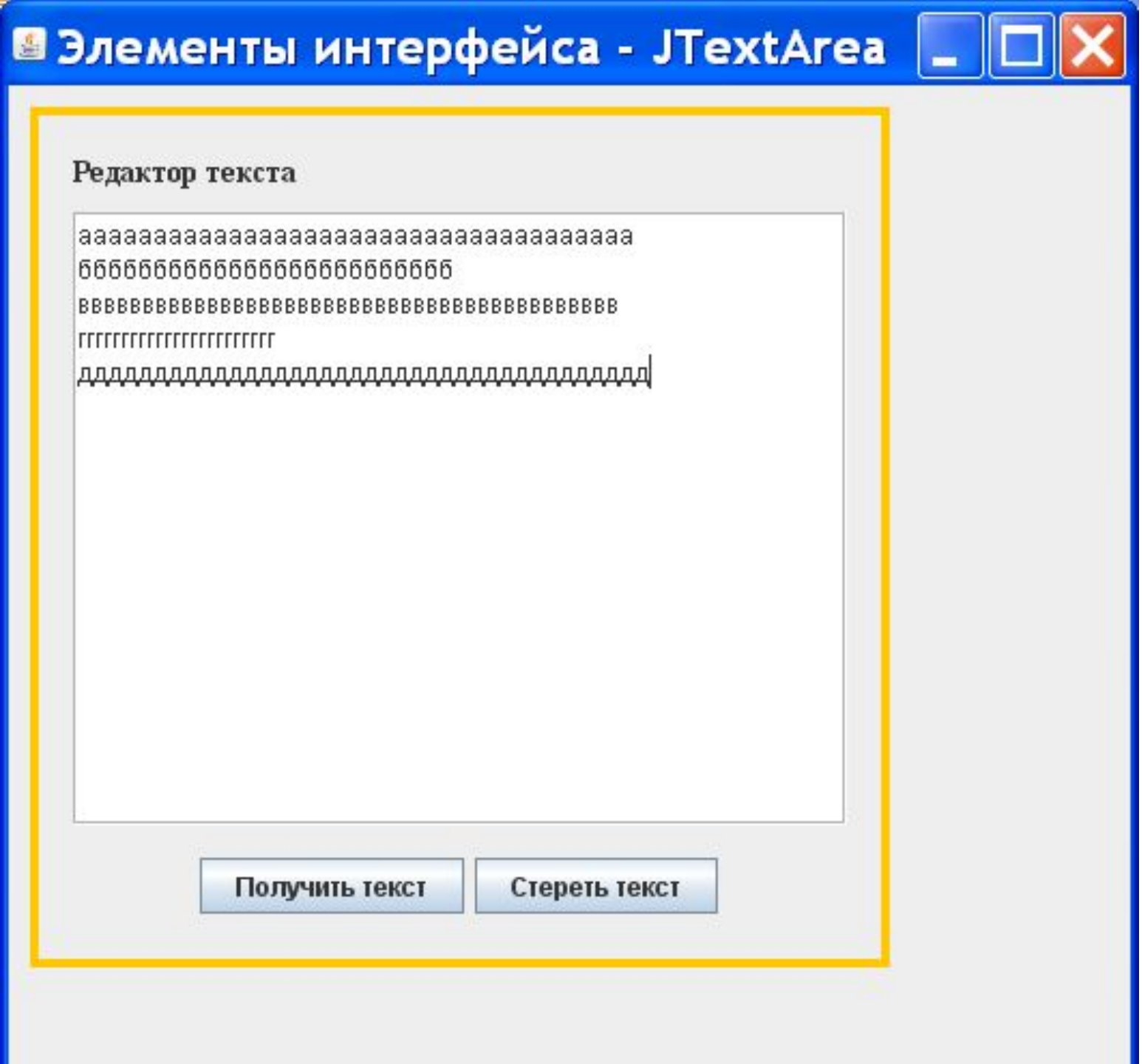
Метод определяет, изменился ли текст в визуальном компоненте.

```
public void actionPerformed (ActionEvent e){
     System.out.println(e);
   if("Получить".equals(e.getActionCommand()))
      if (isChange()){ //только если текст изменился
        toWork();  //переписываем  во  внутр. представление
        System.out.println(textIn); //выводим на терминал
      }
   if("Стереть".equals(e.getActionCommand())) {
       reset(); //стираем во внутр. и внеш. представлении
       textVis.requestFocus(); //даем фокус текстовой области
   }
 }
 public Insets getInsets(){ //задает отступы рамки
                                  //от панели
    //переопределение метода класса-родителя
    return new Insets(20,20,20,20);
 }
}
```

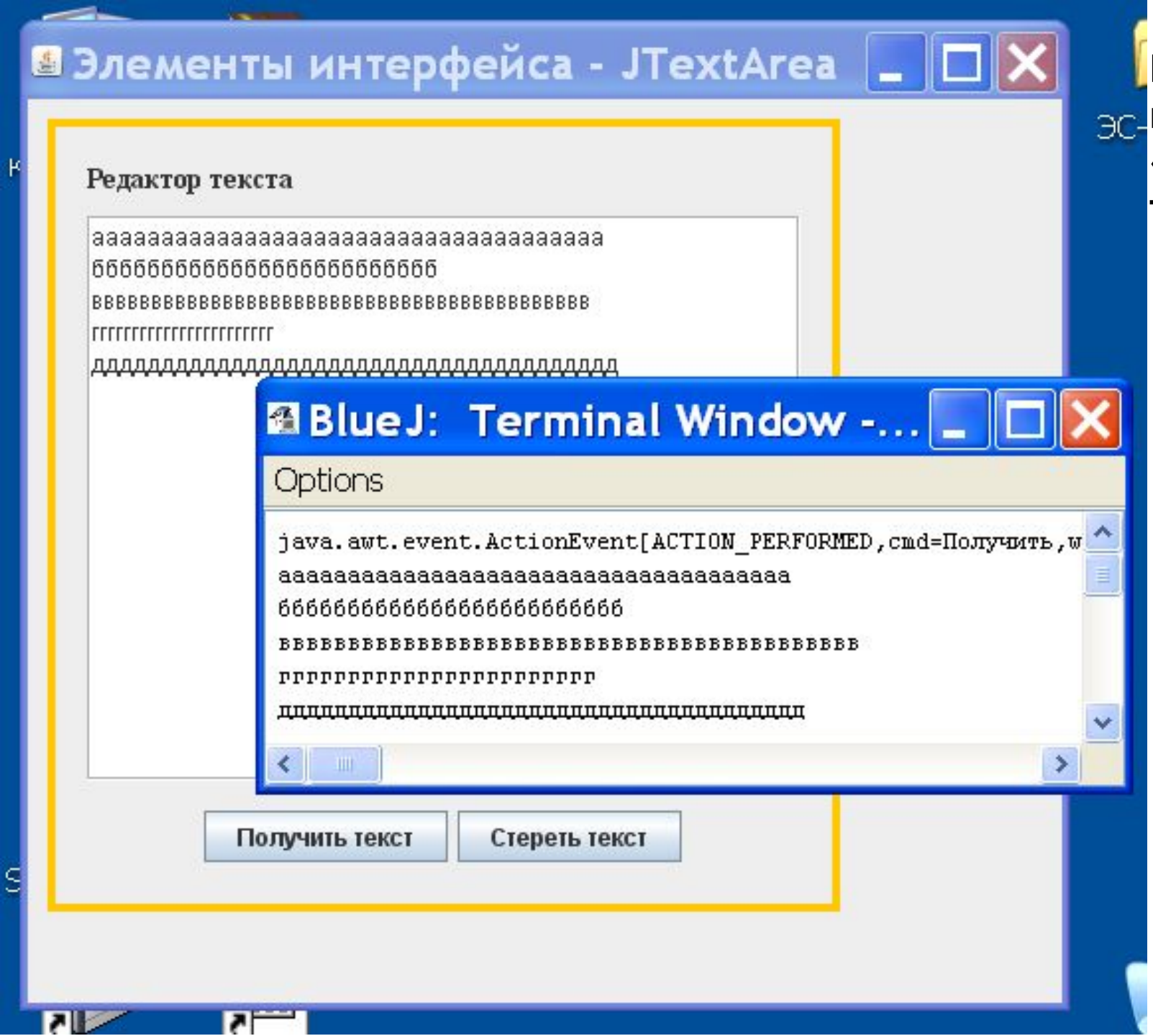
```
import java.awt.*;
import javax.swing.*;
public class MyFrame{
  private static void createAndShowGUI(){
      JFrame frame = new JFrame("Элементы интерфейса - JTextArea");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      Container myC=frame.getContentPane();
      myC.setLayout(new FlowLayout(FlowLayout.LEFT,10,10));
      TextProcess tp = new TextProcess();
      myC.add(tp);
      frame.setSize(530,500);
      frame.setLocation(10,10);
      frame.setVisible(true);
  }
  public static void main (String[ ] args){
      javax.swing.SwingUtilities.invokeLater(new Runnable(){
          public void run(){createAndShowGUI();}});
  }
}
```



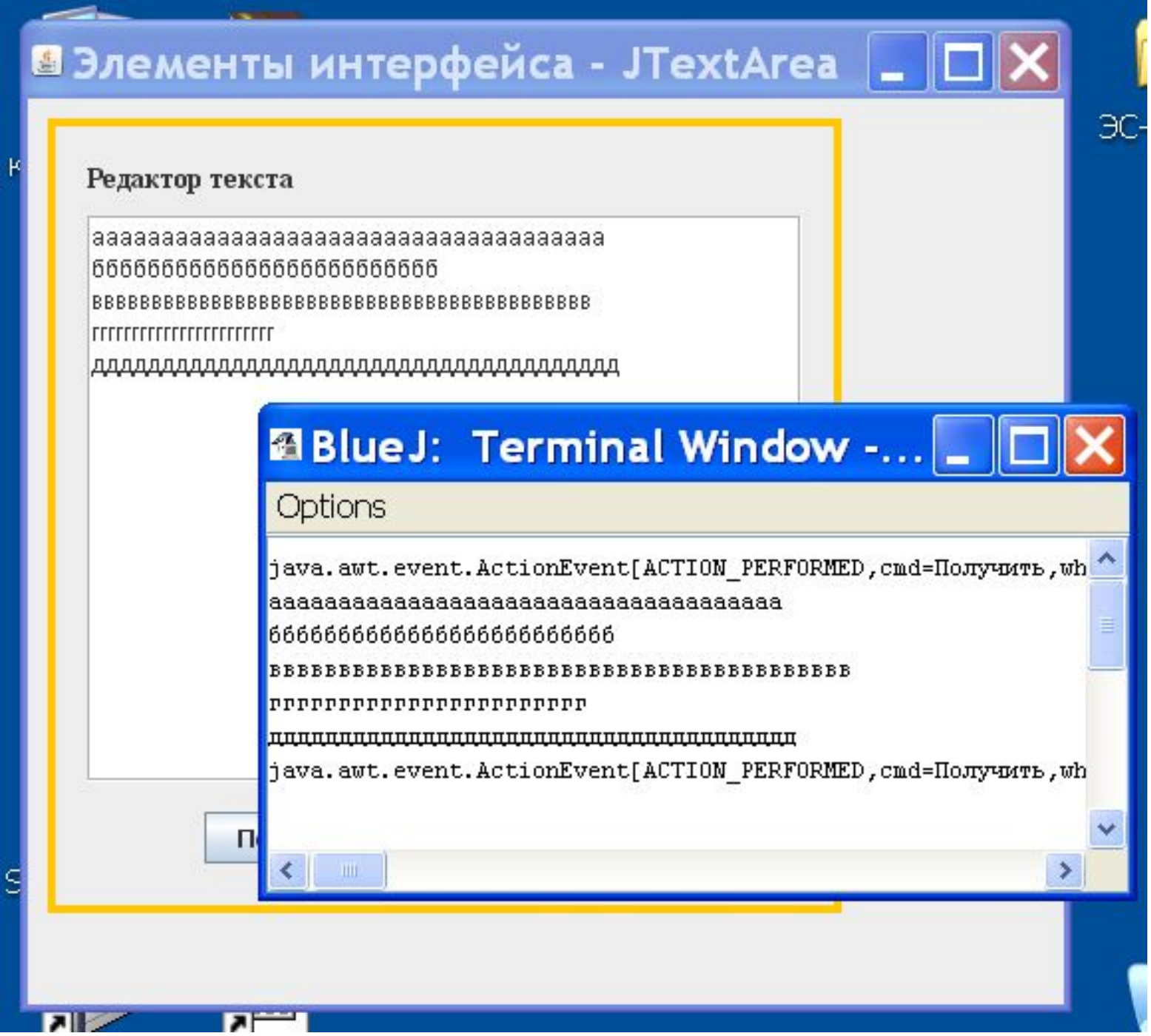
После запуска


Элементы интерфейса - JTextArea
Редактор текста
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
бббббббббббббббббббббббббб
вввввввввввввввввввввввввввввввввввввв
гггггггггггггггггггг
дд
Получить текст
Стереть текст

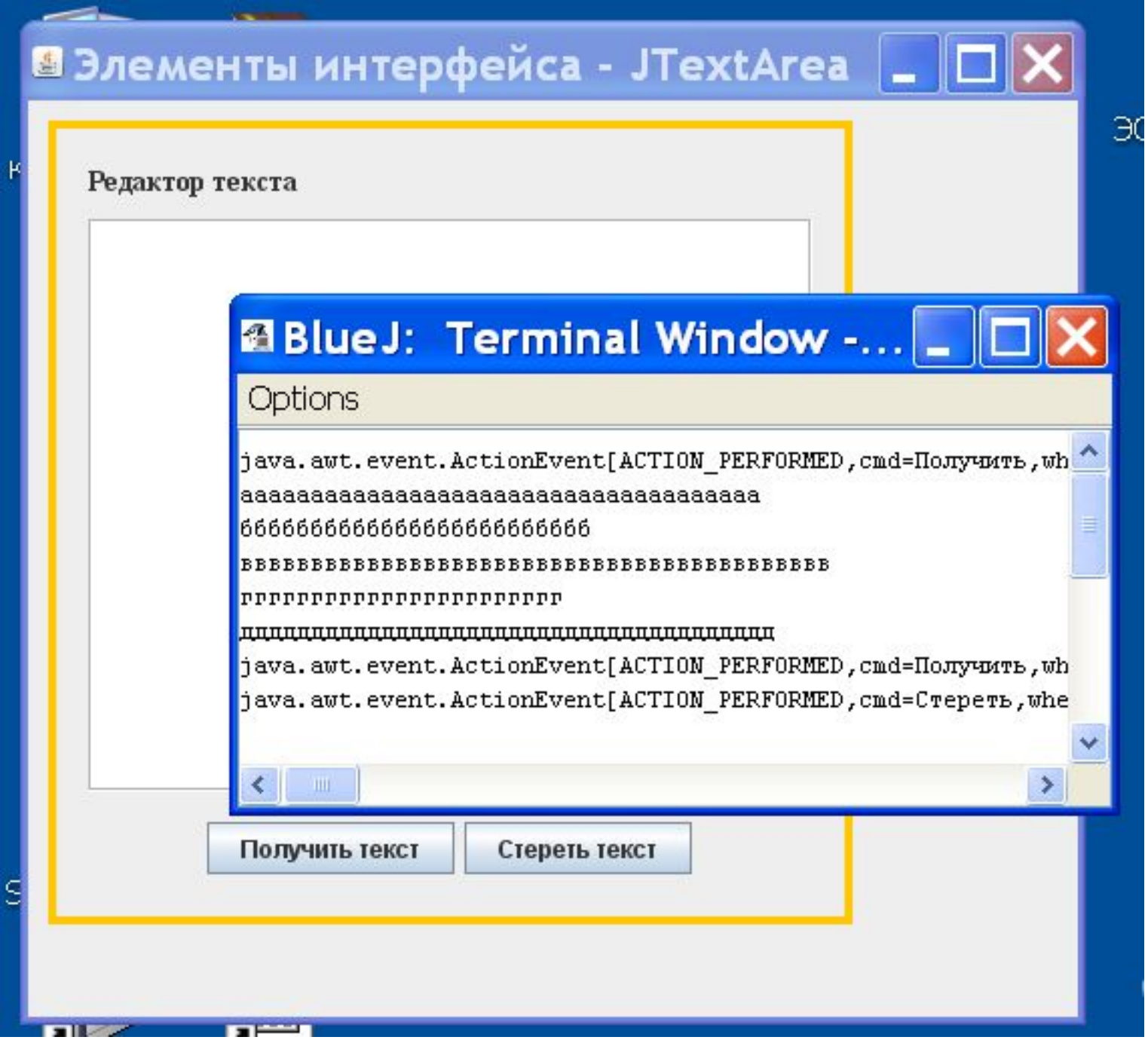
Ввод текста



Нажата кнопка «Получить текст»

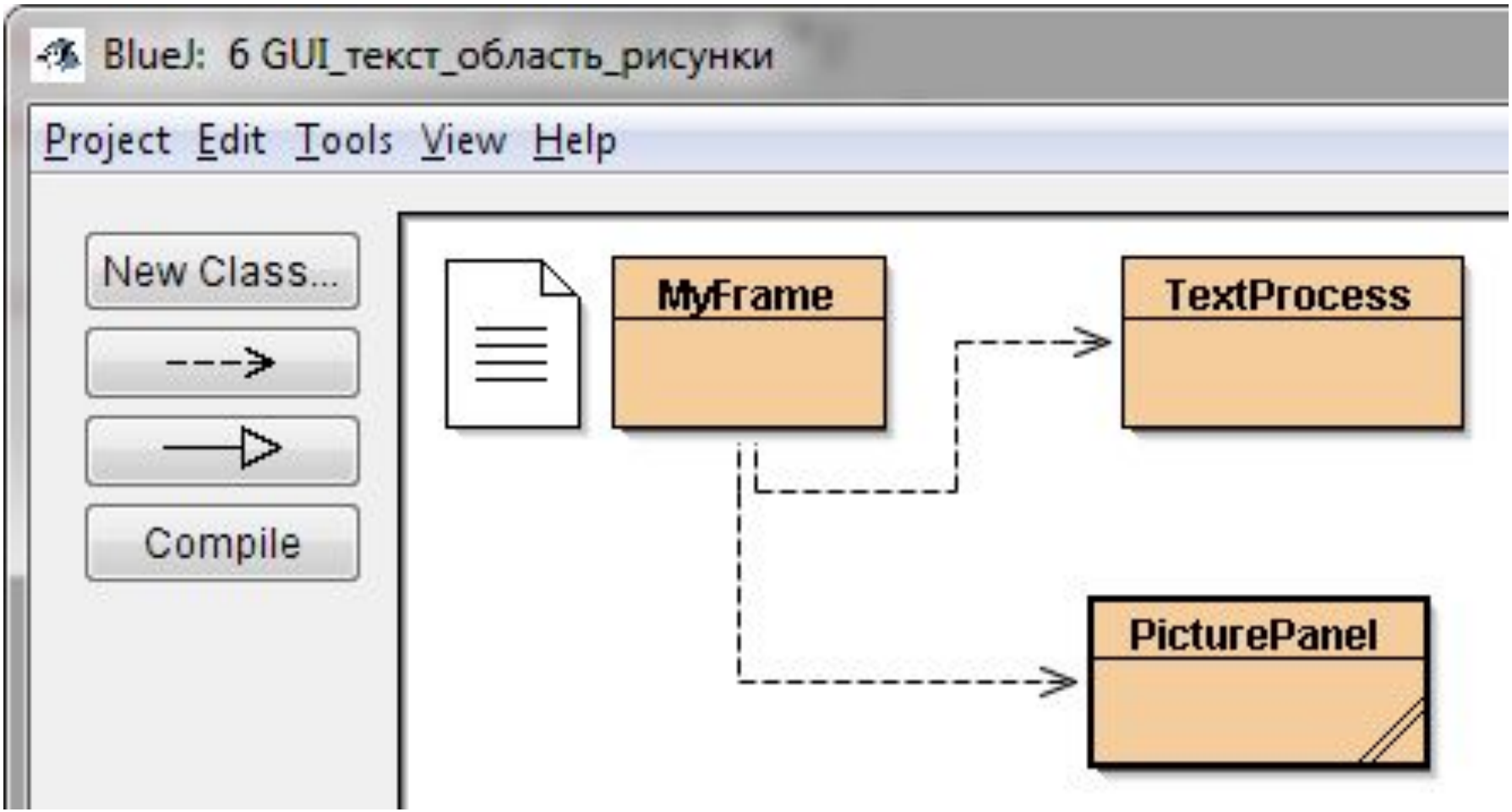


Еще раз нажата кнопка «Получить текст» (текст не менялся) – нет вывода текста в окно терминала.

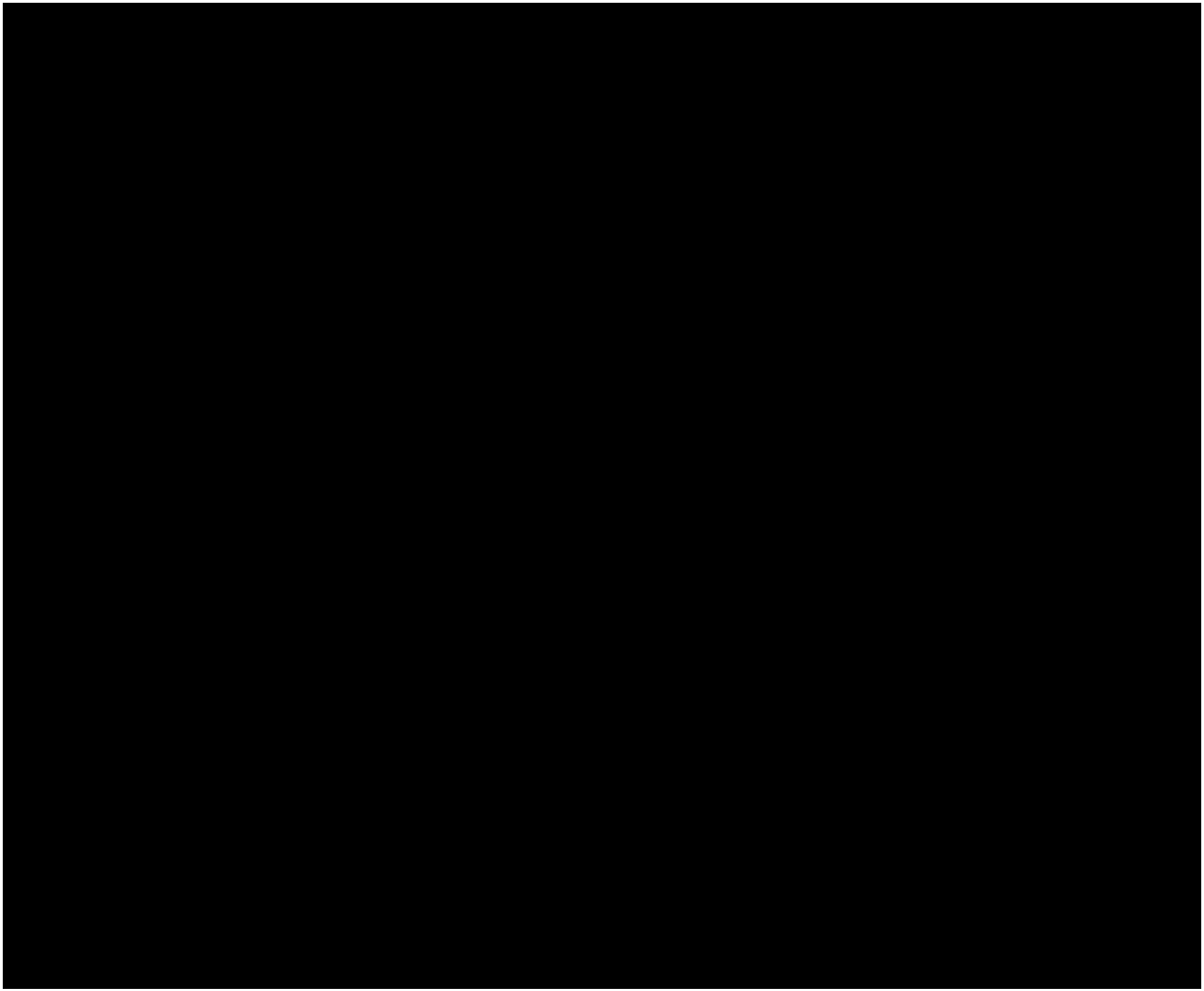


Нажата кнопка «Стереть текст»

Фрейм с двумя панелями и скроллингом.



Класс TextProcess взят из прошлого проекта



```
import java.awt.*; //Проект - 6 GUI_текст_область_рисунки
import javax.swing.*;
import java.awt.event.*;
public class PicturePanel extends JPanel implements ActionListener{

 JLabel pic1, pic2, pic3, pic4, name;
 JButton but1; //кнопка "Перемешать"
 String [ ] PicFileName = {"аист.gif","динозавр.gif",
                           "петух.gif","лошадь.gif"};
 public PicturePanel (){
    Dimension d1 = new Dimension(400,400);
    setPreferredSize(d1);
    setBorder(new javax.swing.border.LineBorder(Color.ORANGE, 4));
    setLayout(new BorderLayout(10,10));
    name = new JLabel("Панель изображений");
    pic1 = new JLabel(new ImageIcon(PicFileName[0]));
    pic2 = new JLabel(new ImageIcon(PicFileName[1]));
    pic3 = new JLabel(new ImageIcon(PicFileName[2]));
    pic4 = new JLabel(new ImageIcon(PicFileName[3]));
    JPanel picPan = new JPanel();
    picPan.setLayout(new GridLayout(2,2));
    picPan.setBorder(new javax.swing.border.EtchedBorder());
```

```
    picPan.add(pic1); picPan.add(pic2);
    picPan.add(pic3); picPan.add(pic4);
    add(picPan,BorderLayout.CENTER);
    JButton but1 = new JButton ("Перемешать");
    but1.addActionListener(this);
    add(but1,BorderLayout.SOUTH);
    add(name,BorderLayout.NORTH);
 }
public void actionPerformed (ActionEvent e){
      //перемешивание элементов в массиве имен
      // файлов иконок
      for (int i = 3; i >= 0; i--){
          int j = (int) (Math.random() * (i+1)); //j  не будет больше 3
          String a = PicFileName[ i ];
          PicFileName[ i ] = PicFileName[ j ];
          PicFileName[ j ] = a;
      }
      //установка иконок
      pic1.setIcon(new ImageIcon(PicFileName[0]));
      pic2.setIcon(new ImageIcon(PicFileName[1]));
      pic3.setIcon(new ImageIcon(PicFileName[2]));
      pic4.setIcon(new ImageIcon(PicFileName[3]));
}
 public Insets getInsets(){ //задает отступы рамки от панели
    return new Insets(20,20,20,20);
   }}
```

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
public class MyFrame {
    private static void createAndShowGUI(){
      JScrollPane SP_main; //панель скроллинга для главной панели проекта
      JPanel P_main; //главная панель проекта
      Container  myC; //ссылка на панель контента
      int WinSizeG = 850; //начальный размер окна по горизонтали
      int WinSizeV = 500; //начальный размер окна по вертикали
      JFrame frame = new JFrame("Фрейм с двумя панелями");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      //******************Создание элементов главной панели**********************
      TextProcess tp = new TextProcess();
      PicturePanel pp=new PicturePanel();
      //*******************сборка панелей*************************************
      myC = frame.getContentPane();
      myC.setLayout(new BorderLayout()); //должна быть такая компоновка,
                                         //чтобы отображались полосы прокрутки

      P_main = new JPanel();  //главная панель проекта
      P_main.setBorder(new javax.swing.border.LineBorder(Color.GREEN, 4));
```

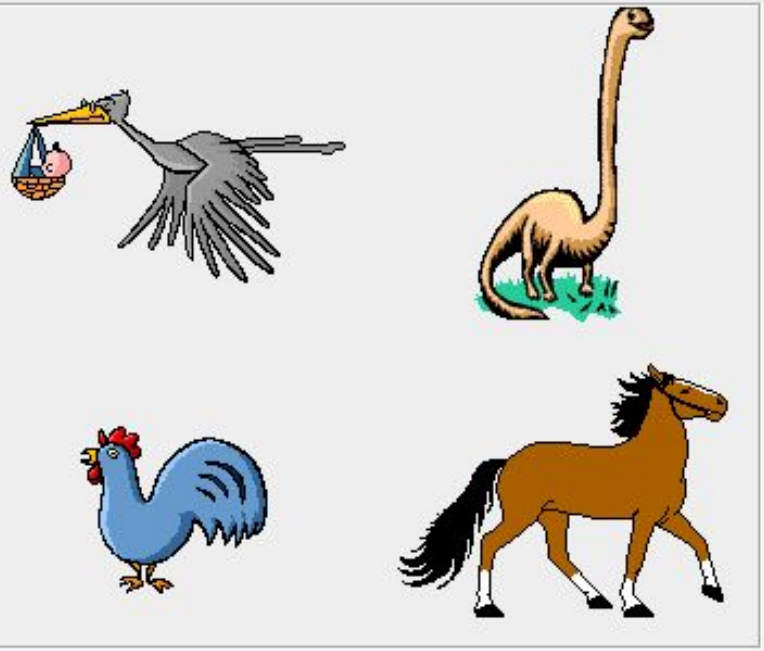
```
    P_main.add(tp);
    P_main.add(pp);

    int v = ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
    int h = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
    SP_main = new JScrollPane(P_main, v, h); //панель прокрутки для P_main
    myC.add(SP_main, BorderLayout.CENTER);
    // полосы прокрутки будут отображаться только тогда,
    //когда размеры контейнера, содержащего панель прокрутки
    //будут меньше, чем размеры панели P_main,
    // которые складываются из размеров панелей tp и pp
    // и отступов
    frame.setSize(WinSizeG,WinSizeV);
    frame.setLocation(10,10);
    frame.setVisible(true);
  }
public static void main (String[] args){
    javax.swing.SwingUtilities.invokeLater(new Runnable(){
        public void run(){createAndShowGUI();}});
  }
}
```

Фрейм с двумя панелями

Редактор текста

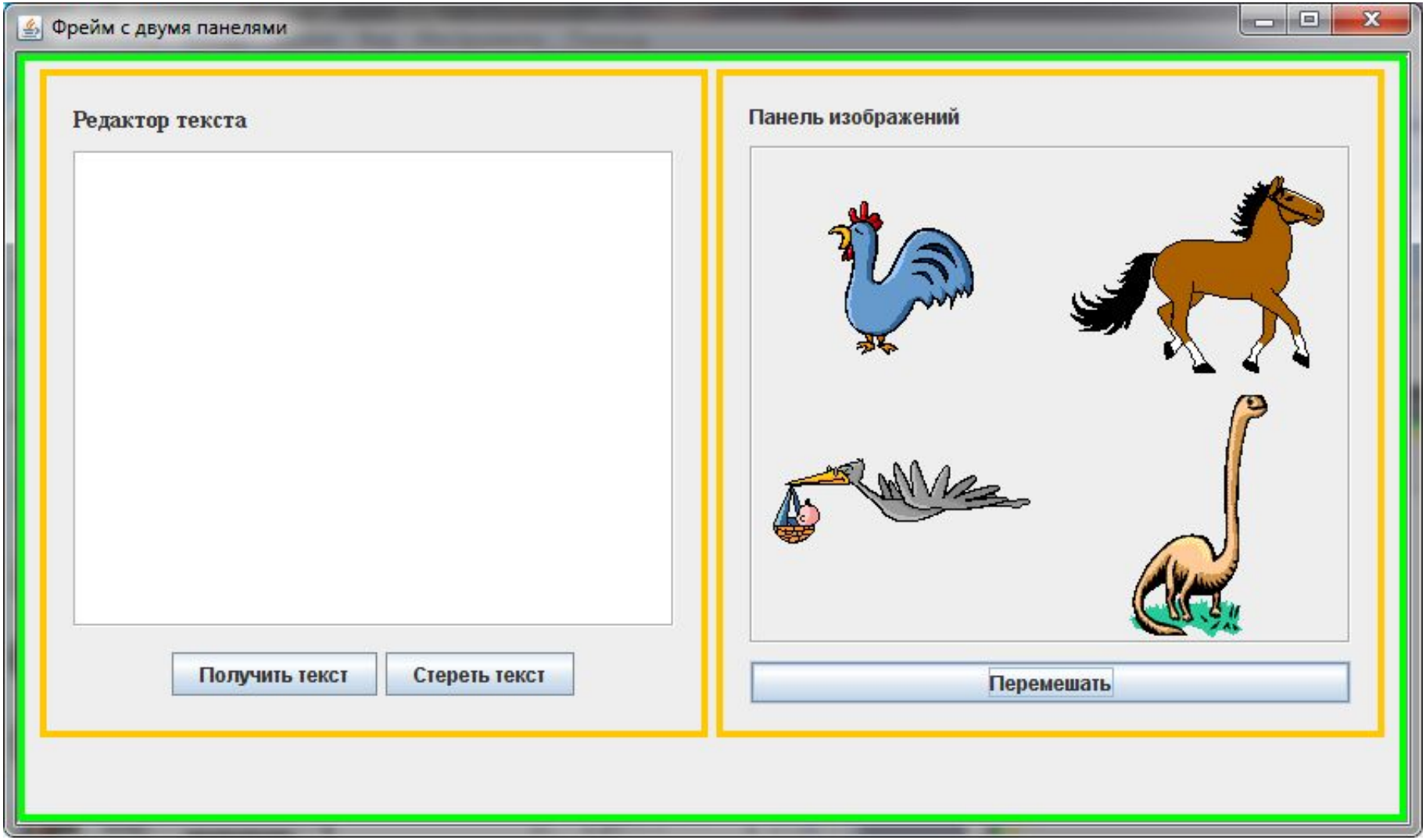
Получить текст | Стереть текст

Панель изображений

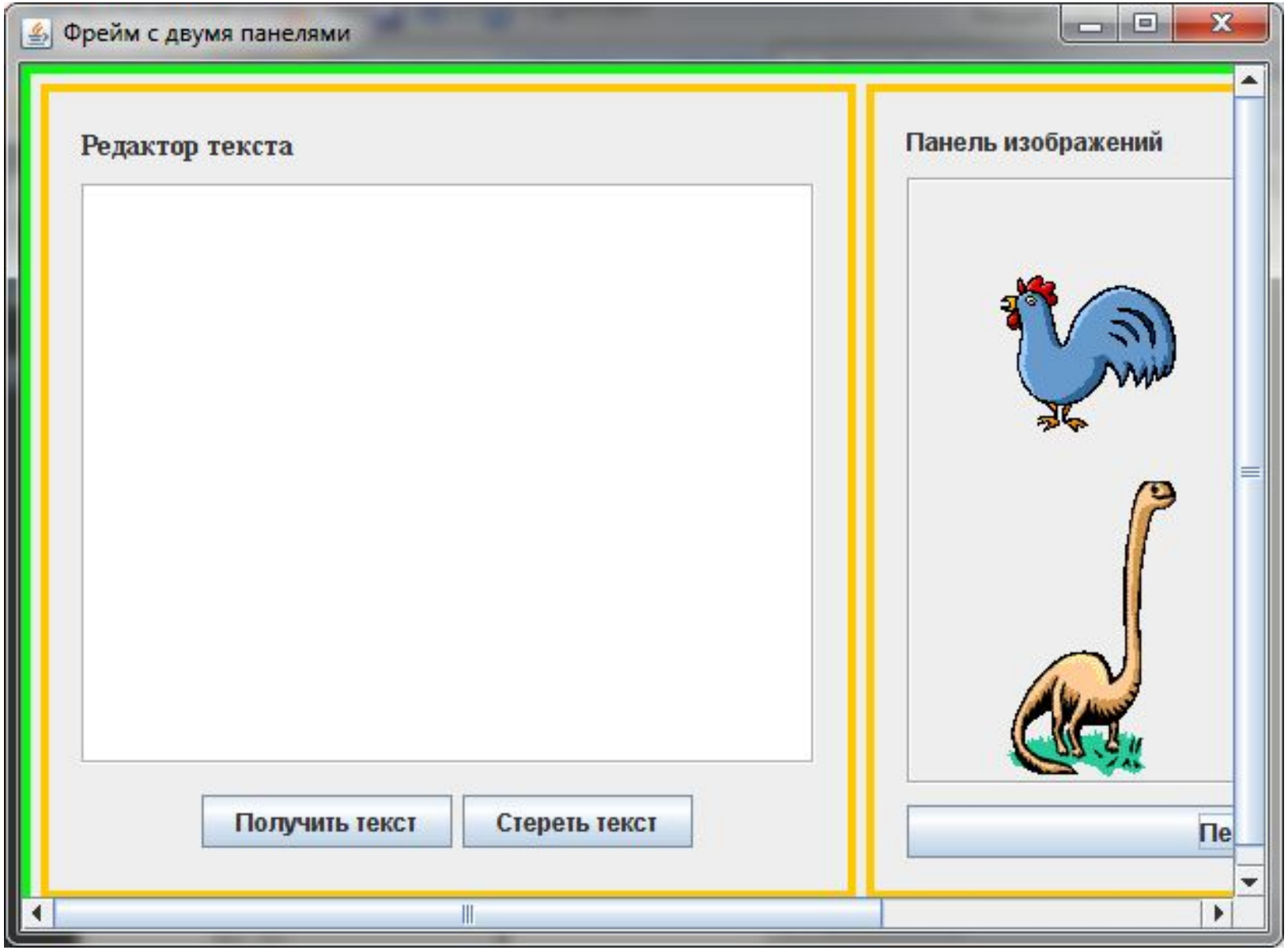


Перемешать

После нажатия на кнопку «Перемешать»:



При уменьшении размера окна появляются полосы прокрутки



Создание собственных событий разобрать самостоятельно (книга И. Портянкина и гугл вам в помощь ;-)