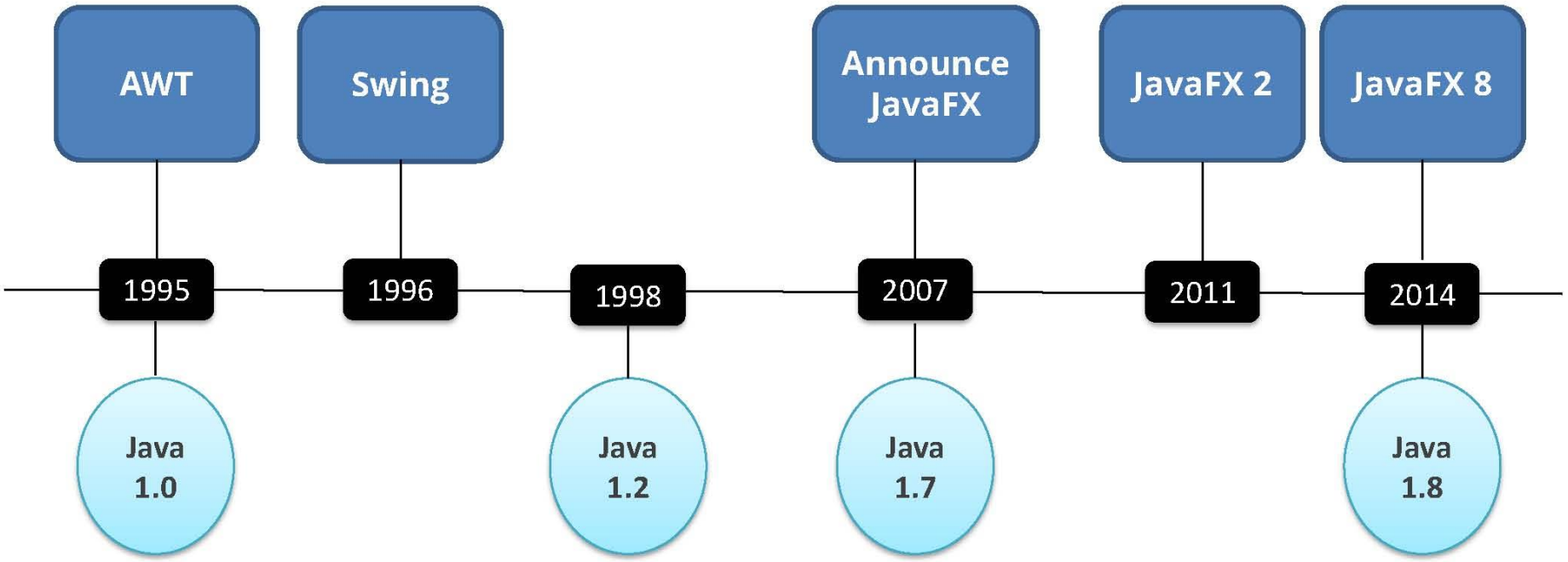
Module contents

1. Java GUI Programming
   - An Introduction to Swing
   - Swing - Controls
   - Event Handling
   - Layout Managers

Module contents

1. Java GUI Programming
   - An Introduction to Swing
   - Swing - Controls
   - Event Handling
   - Layout Managers

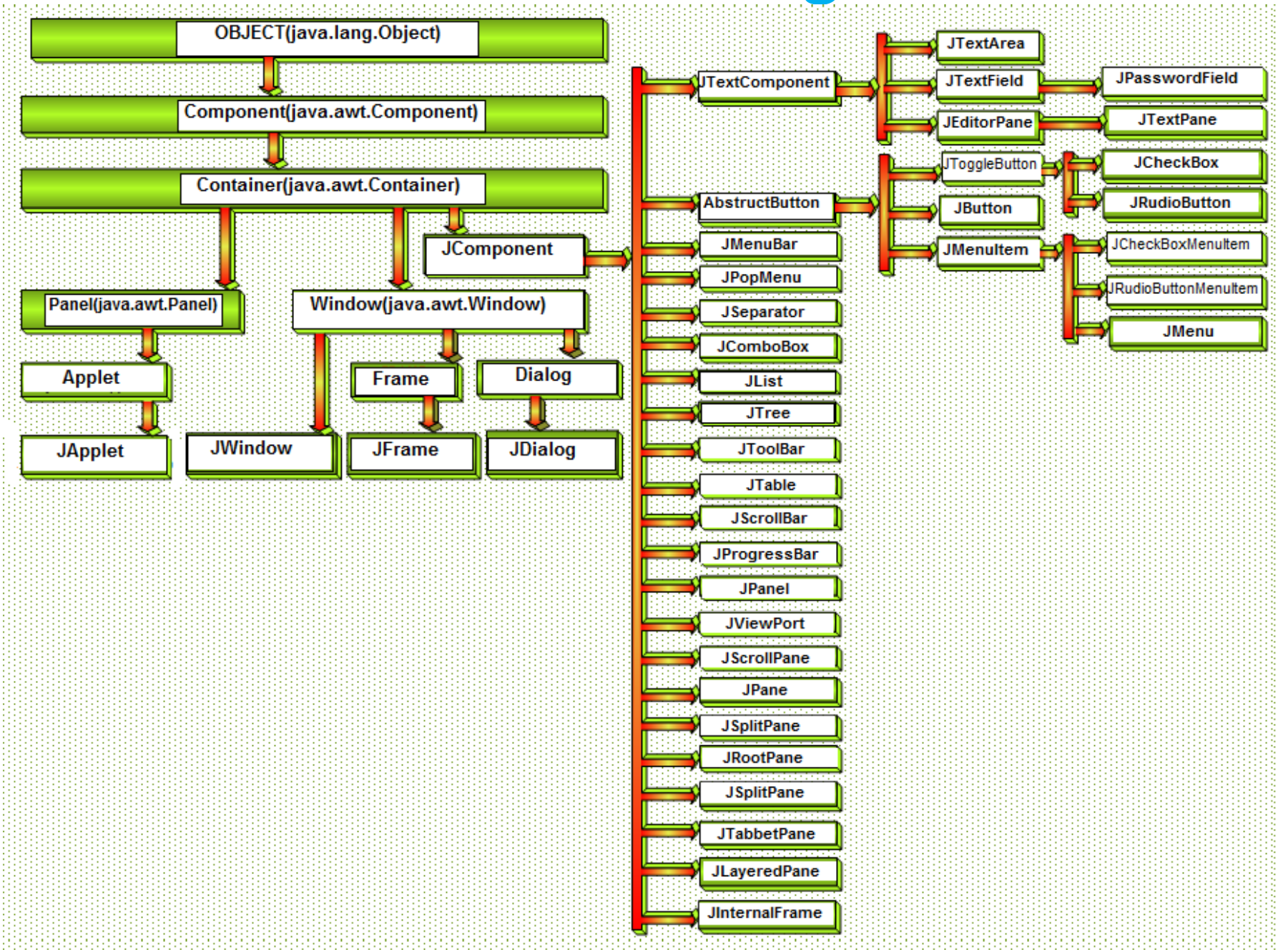
An Introduction to Swing 1/2



An Introduction to Swing 2/2

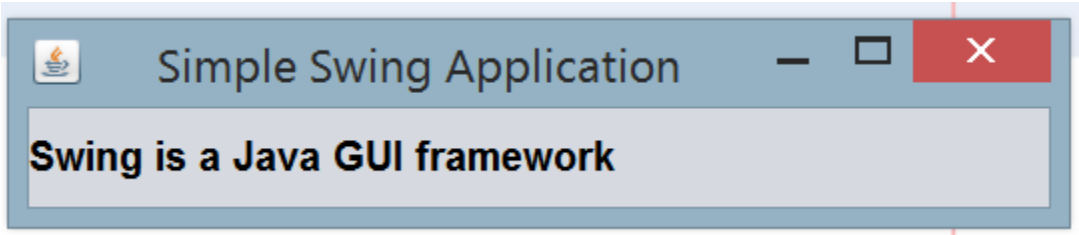
- Swing API is set of extensible GUI Components to ease developer's life to create JAVA GUI Applications.
- It is build upon top of AWT API and acts as replacement of AWT API as it has almost every control corresponding to AWT controls.

AWT - Swing


OBJECT(java.lang.Object)
Component(java.awt.Component)
Container(java.awt.Container)
JComponent
Panel(java.awt.Panel)
Window(java.awt.Window)
Applet
Frame
Dialog
JApplet
JWindow
JFrame
JDialog
JTextComponent
JTextArea
JTextField
JPasswordField
JEditorPane
JTextPane
AbstructButton
JToggleButton
JCheckBox
JRudioButton
JButton
JMenuItem
JCheckBoxMenuItem
JRudioButtonMenuItem
JMenu
JMenuBar
JPopMenu
JSeparator
JComboBox
JList
JTree
JToolBar
JTable
JScrollBar
JProgressBar
JPanel
JViewPort
JScrollPane
JPane
JSplitPane
JRootPane
JSplitPane
JTabbetPane
JLayeredPane
JInternalFrame

Simple Swing Application

```
import javax.swing.*;

public class SwingDemo {

    public SwingDemo() {
        JFrame jfrm = new JFrame("Simple Swing Application");
        jfrm.setSize(370, 100);
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel jlab = new JLabel("Swing is a Java GUI framework");
        jfrm.add(jlab);
        jfrm.setVisible(true);
    }

    public static void main(String args[]) {
        SwingDemo swingDemo = new SwingDemo();
    }
}
```

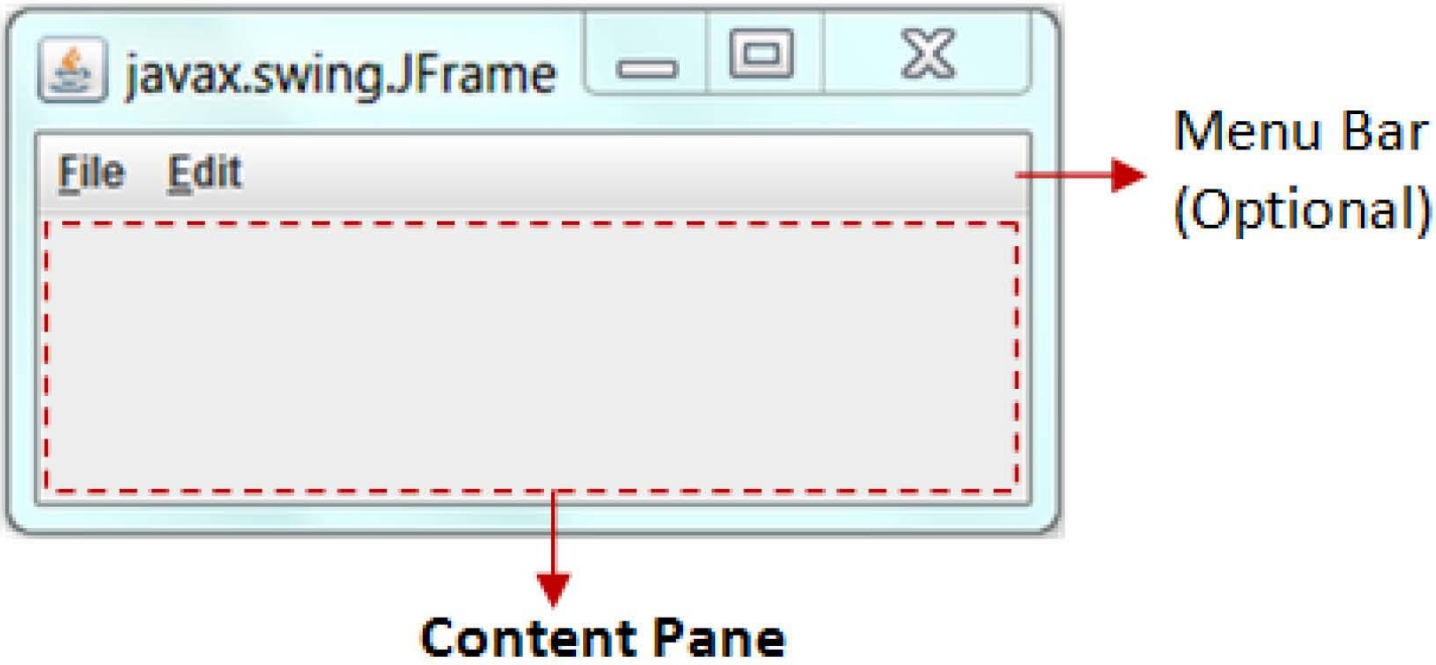


Module contents

1. Java GUI Programming
   - An Introduction to Swing
   - Swing - Controls
   - Event Handling
   - Layout Managers

Swing – Controls 2/6

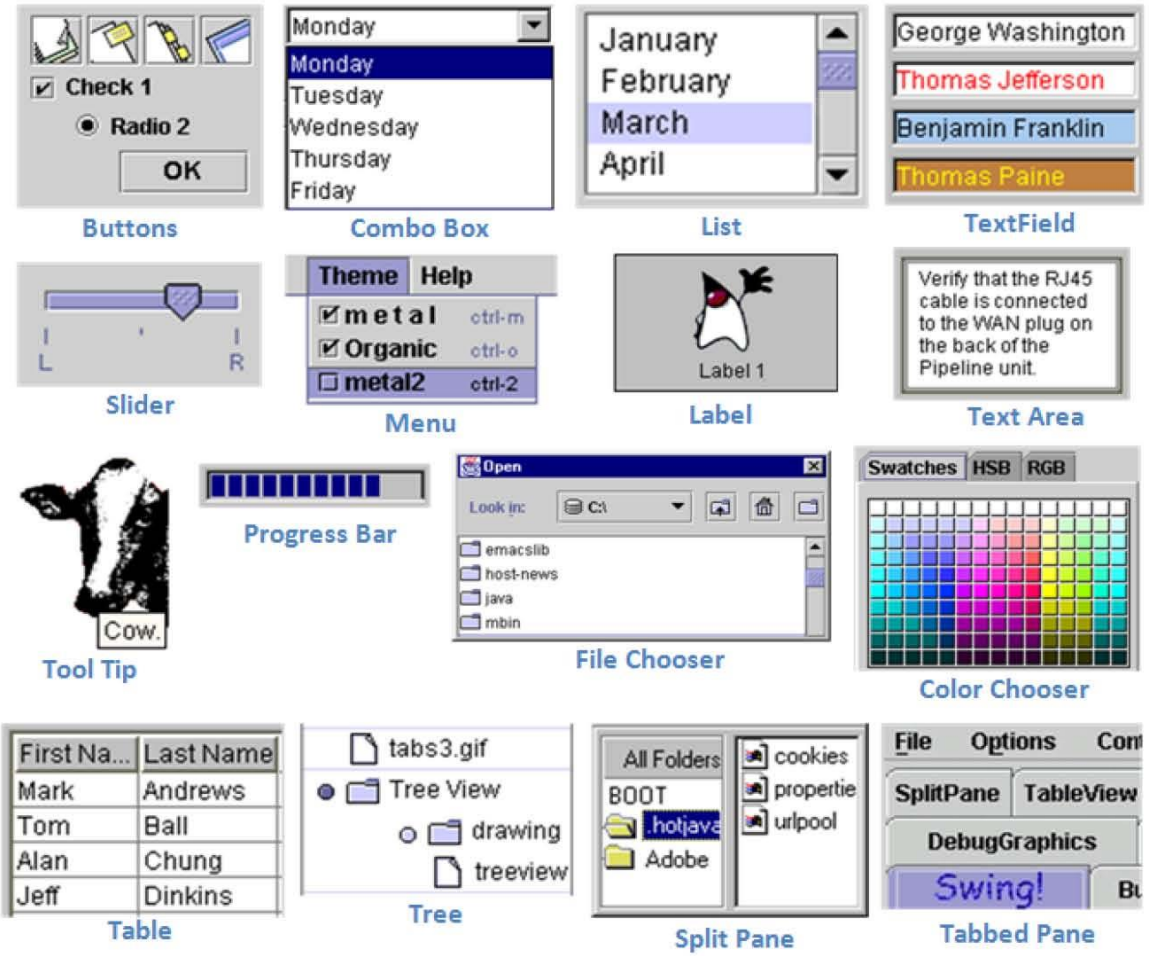
javax.swing.JFrame




```
Container cp = aJFrame.getContentPane();
aJFrame.setContentPane(aPanel);
```

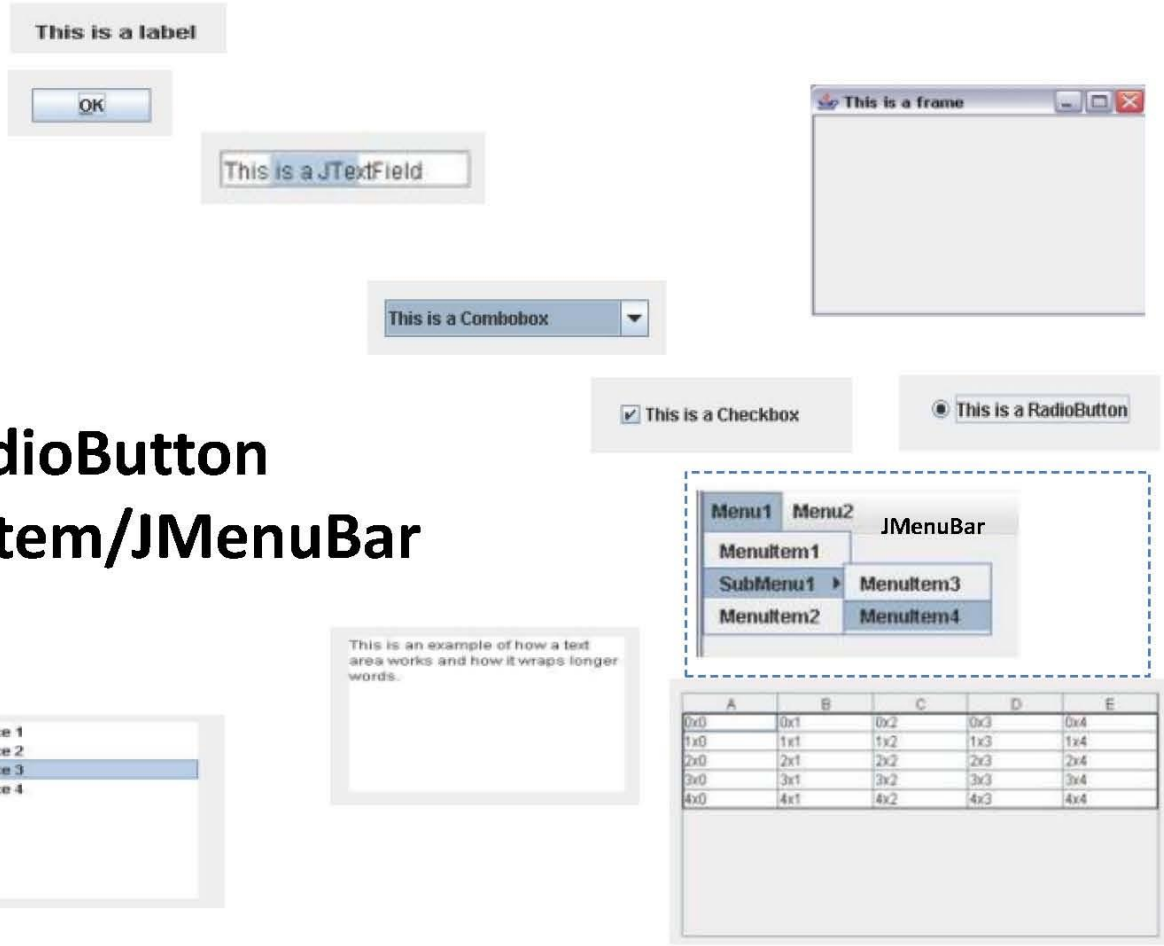
Swing – Controls 1/6

- Swing provides a rich set of advanced controls



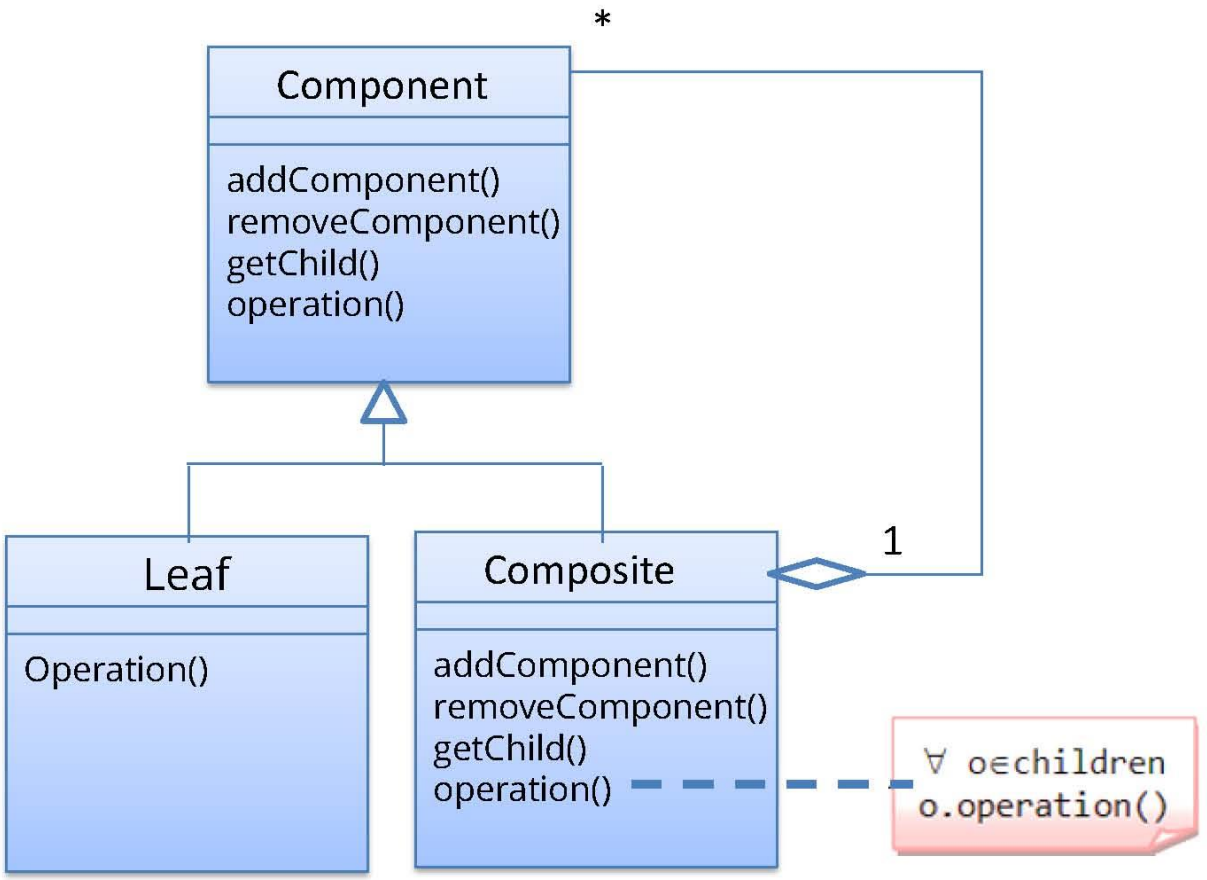
Swing – Controls 6/6

JLabel
JButton
JTextField
JFrame
JComboBox
JCheckBox/JRadioButton
JMenu/JMenuItem/JMenuBar
JTextArea
JList
JTable



Swing – Controls 3/6

- *Composite pattern*



Interface - Component

```
public interface SwComponent {

   /* A useful method - listing all the components of the
component container*/
   void list();
}
```

Class - leaf-1

```
public class SwTextField implements SwComponent {

    String name;

    public SwTextField(String name) {
        this.name = name;
    }

    @Override
    public void list() {
        System.out.println("This is the TextField: " + name);
    }
}
```

Class - leaf-2

```
public class SwButton implements SwComponent {

    String name;

    public SwButton(String name) {
        this.name = name;
    }

    @Override
    public void list() {
        System.out.println("This is the Button: " + name);
    }
}
```

Class - Composite

```
public class SwComposite implements SwComponent {

    private List<SwComponent> swComponents =
                                                new ArrayList<SwComponent>();
    @Override
    public void list() {
        int i = 0;
        System.out.println("Компоненты контейнера:");
        for (SwComponent component : swComponents) {
            System.out.print(i + ". ");
            component.list();
            i++;
        }
        System.out.println();
    }
...
```

Class - Composite

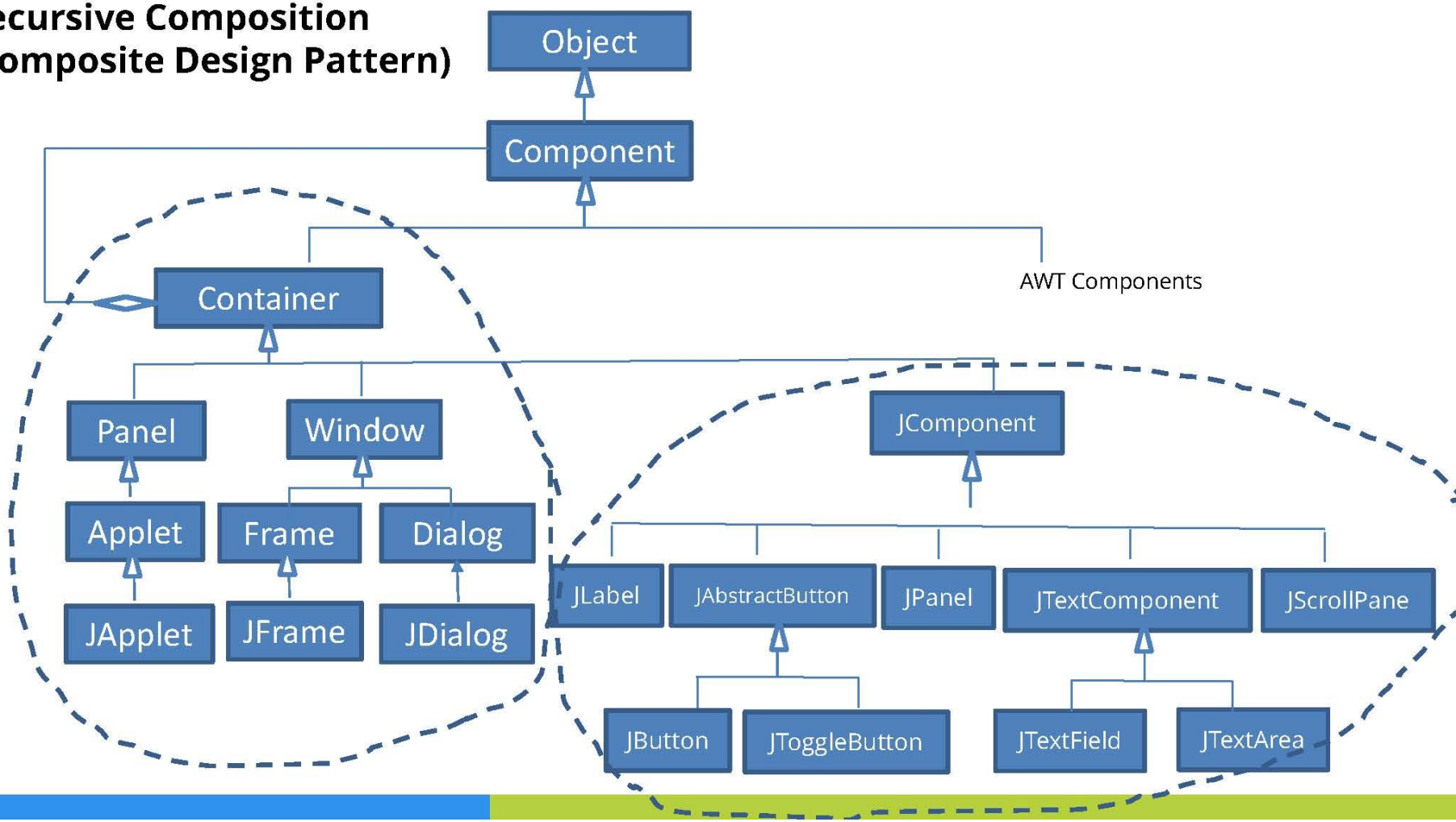
```
    …
    void add(SwComponent component) {
        swComponents.add(component);
    }

    void remove(SwComponent component) {
        swComponents.remove(component);
    }
}
```

Class - Composite

```
public class CompsitePatternDemo {

   public static void main(String[] args) {
      SwButton btn1 = new SwButton("Кнопка 1");
      SwButton btn2 = new SwButton("Кнопка 2");

      SwComposite panel = new SwComposite();
      panel.add(btn1);
      panel.add(btn2);

       SwButton btn3 = new SwButton("Кнопка 3");
      SwTextField txt1 = new SwTextField("Текстовое поле 1");

      SwComposite subPanel = new SwComposite();
      subPanel.add(btn3);
      subPanel.add(txt1);
      panel.add(subPanel);

      panel.list();
   }
```

Swing – Controls 4/6

Recursive Composition
(Composite Design Pattern)

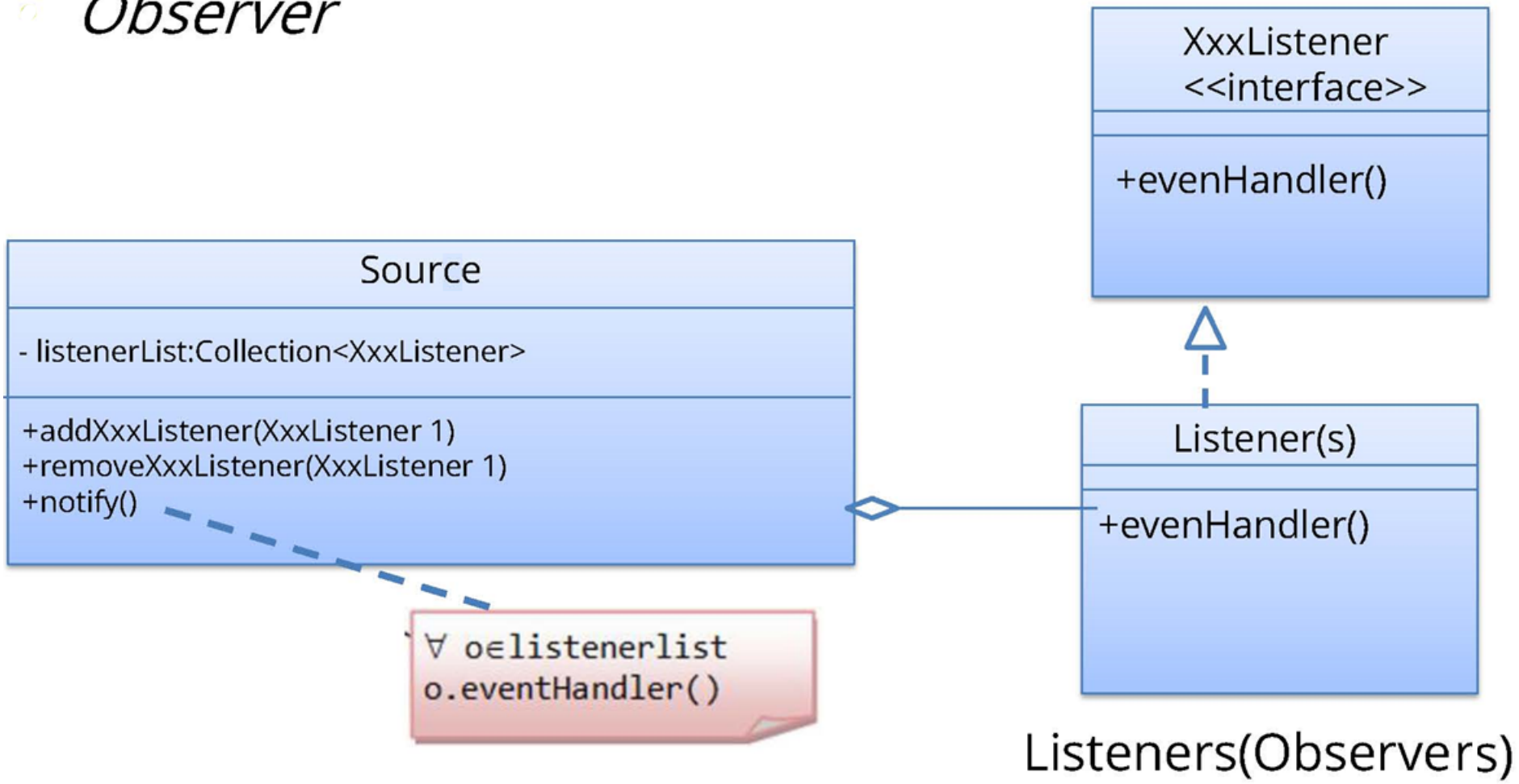


Module contents

1. Java GUI Programming
   - An Introduction to Swing
   - Swing - Controls
   - Event Handling
   - Layout Managers

Listener (observer) pattern

- *Observer*



Listener (observer) pattern

```
public interface SwListener {

   /*Event handler method in an observer */
   void actionPerformed(SwEvent event);
}

public interface SWSubject {

   /*Methods of registration and unregistration of observers*/
   void addListener(SwListener listener);
   void removeListener(SwListener listener);
   /*Observers notice method to change the state of an object that
implements the Subject*/
   void notifyListeners();
}
```

Listener (observer) pattern

```
public class SwComponent implements SWSubject {
    List<SwListener> listeners = new ArrayList<>();
    SwEvent event;
    public void fireEvent() {
        event = new SwEvent("нажата кнопка");
        notifyListeners();
    }
    @Override
    public void addListener(SwListener listener) {
        listeners.add(listener);
    }
    @Override
    public void removeListener(SwListener listener) {
        listeners.remove(listener);
    }
…
```

Listener (observer) pattern

```
…
    @Override
    public void notifyListeners() {
        for (SwListener listener : listeners) {
            listener.actionPerformed(event);
        }
    }
}
public class SwEvent {
    String name;
    public SwEvent(String name) {
        this.name = name;
    }
    @Override
    public String toString() {
        return name;
    }
}
```

Listener (observer) pattern

```
public class SwingListenerDemo {
    public static void main(String[] args) {
        SwComponent button = new SwComponent();
        MyListener listener = new MyListener();
        button.addListener(listener);
        button.fireEvent();
    }
}

public class MyListener implements SwListener {
    @Override
    public void actionPerformed(SwEvent event) {
        System.out.println("Событие " + event + " произошло.");
    }
}
```

Event Handling 2/2

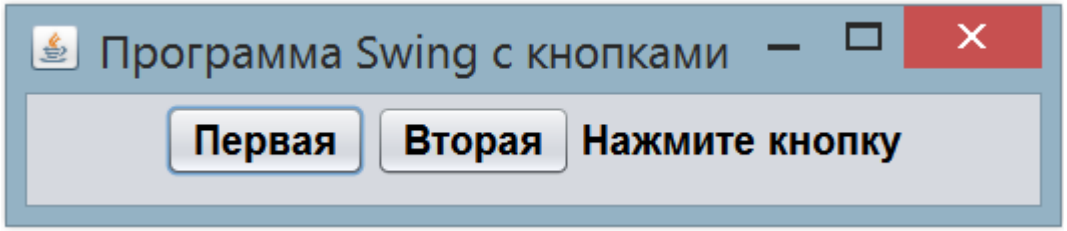
```
1. JButton btn = new JButton(“Click me!");
2. btn.addActionListener(new ActionListener() {
3.     @Override
4.     public void actionPerformed(ActionEvent e) {
5.         System.out.println("Button Click!");
6.     }
7. });
```

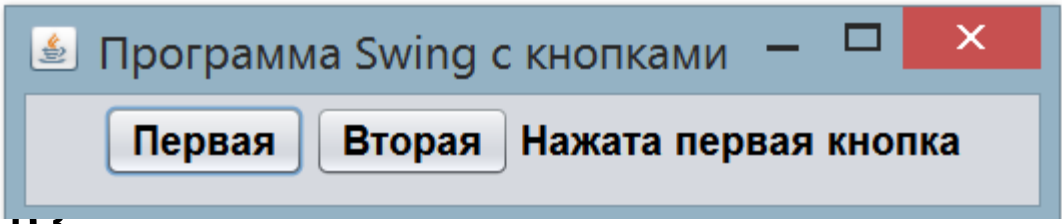
JButton Action Event - 1

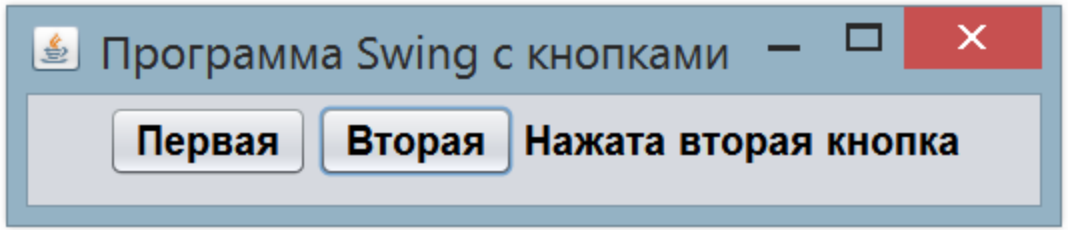
```
import java.awt.FlowLayout;
import java.awt.event.*;
import javax.swing.*;

public class ButtonDemo implements ActionListener {

   JLabel jlab;

   ButtonDemo() {
      JFrame jfrm = new JFrame("Программа Swing с кнопками");
      jfrm.setLayout(new FlowLayout());
      jfrm.setSize(380, 100);
      jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JButton jbtnFirst = new JButton("Первая");
      JButton jbtnSecond = new JButton("Вторая");
      jbtnFirst.addActionListener(this);
      jbtnSecond.addActionListener(this);
      jfrm.add(jbtnFirst);
      jfrm.add(jbtnSecond);
...
```

JButton Action Event - 2

```
...
    jlab = new JLabel("Нажмите кнопку");
        jfrm.add(jlab);
        jfrm.setVisible(true);
    }

     public void actionPerformed(ActionEvent ae) {
        if (ae.getActionCommand().equals("Первая")) {
            jlab.setText("Нажата первая кнопка");
        } else {
            jlab.setText("Нажата вторая кнопка");
        }
    }

    public static void main(String args[]) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                ButtonDemo buttonDemo = new ButtonDemo();
            }
        });
    }
}
```







Some Event Classes - 1

Класс события	Описание	Слушатель
	java.awt.event	
ActionEvent	Генерируется при выполнении действий с интерфейсным элементом, например по щелчку на кнопке	ActionListener
AdjustmentEvent	Генерируется при выполнении действий с полосой прокрутки	AdjustmentListener
FocusEvent	Генерируется тогда, когда компонент получает или теряет фокус ввода	FocusListener
ItemEvent	Генерируется при выборе или отмене выбора элемента, например, по щелчку на флажке опций	ItemListener
KeyEvent	Генерируется при вводе данных с клавиатуры	KeyListener
MouseEvent	Генерируется при перемещении или перетаскивании мыши, нажатии или отпускании клавиши, а также при помещении курсора мыши на компонент или выводе курсора за пределы компонента	MouseListener и MouseMotionListener

Some Event Classes - 2

Класс события	Описание	Слушатель
	java.awt.event	
MouseWheelEvent	Генерируется при движении колёсика мыши	MouseWheelListener
WindowEvent	Генерируется при активизации, деактивизации, закрытии, сворачивании и разворачивании окна	WindowListener
	javax.swing.event	
AncestorEvent	Генерируется при добавлении, перемещении или удалении предка компонента	AncestorListener
CaretEvent	Генерируется при изменении позиции курсора в текстовом компоненте	CaretListener
ChangeEvent	Генерируется при изменении состояния компонента	ChangeListener
HyperlinkEvent	Генерируется при действиях с гипертекстовой ссылкой	HyperlinkListener
ListDataEvent	Генерируется при изменении содержимого списка	ListDataListener

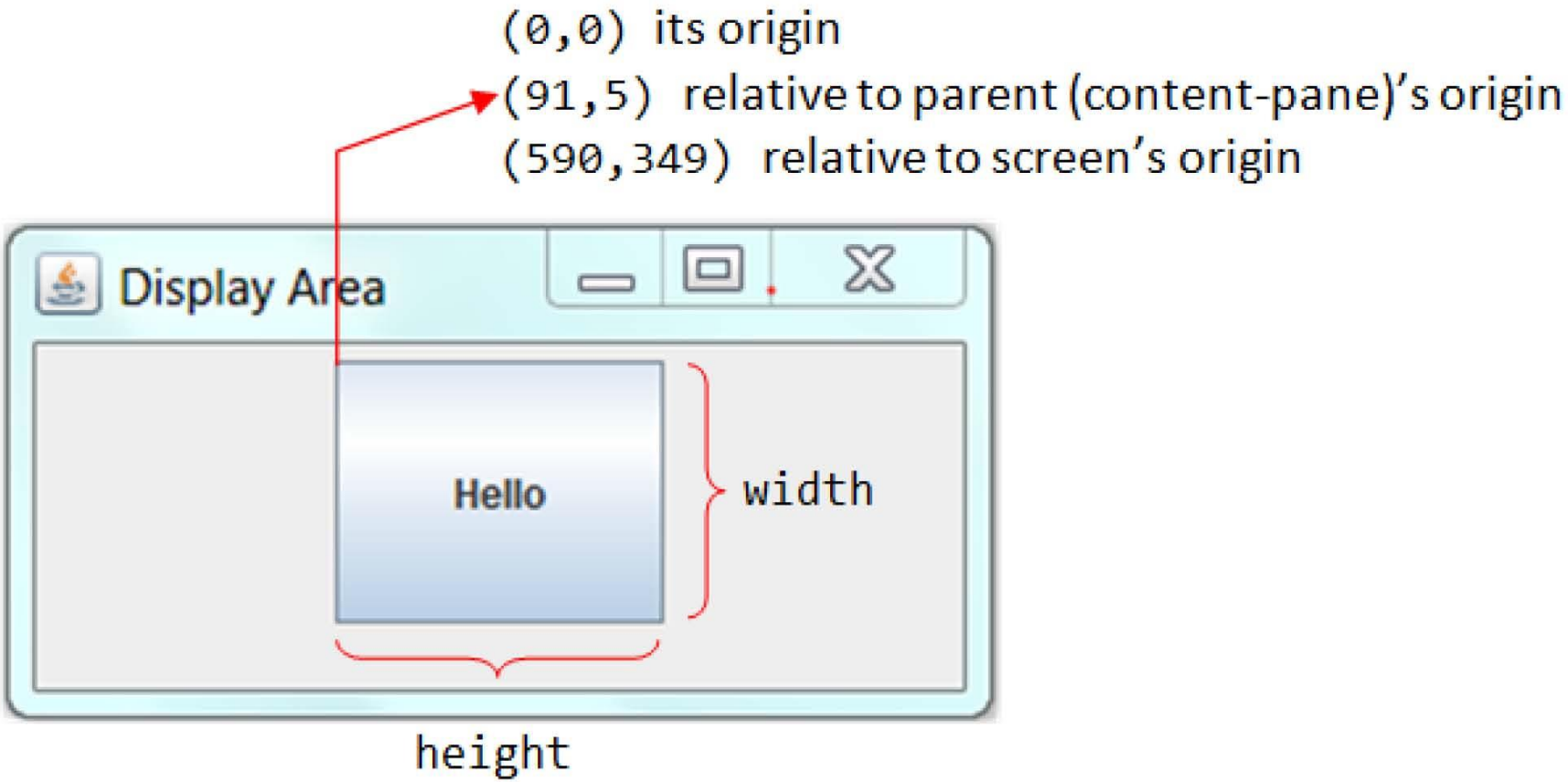
Some Event Classes - 3

Класс события	Описание	Слушатель
javax.swing.event		
ListSelectionEvent	Генерируется при выборе или отмене выбора пунктов списка	ListSelectionListener
MenuEvent	Генерируется при выборе или отмене выбора пунктов меню	MenuListener
TableModelEvent	Генерируется при изменении модели таблицы	TableModelListener
TreeExpansionEvent	Генерируется при разворачивании или сворачивании дерева	TreeExpansionListener
TreeModelEvent	Генерируется при изменении модели дерева	TreeModelListener
TreeSelectionEvent	Генерируется при выборе узла дерева	TreeSelectionListener

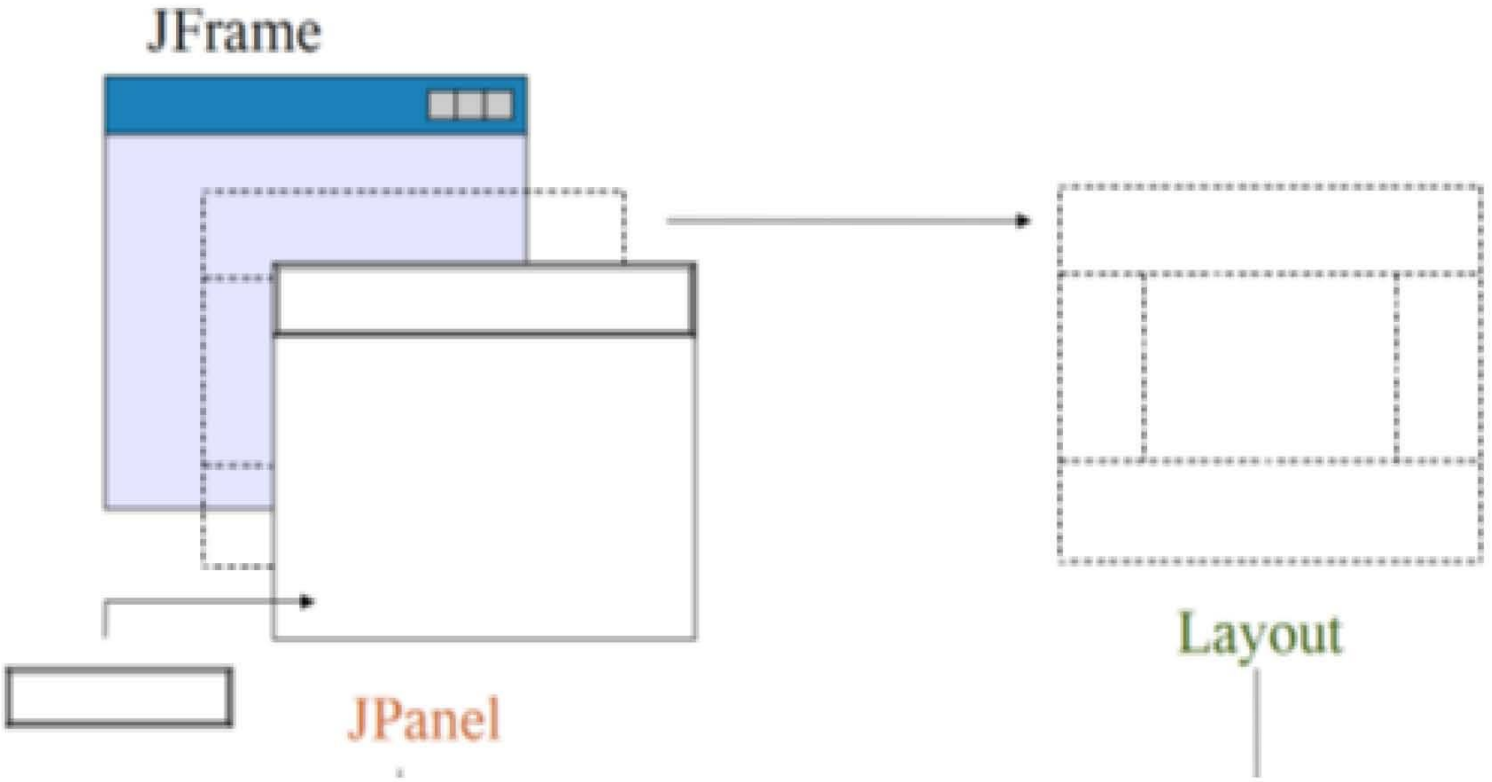
Module contents

1. Java GUI Programming
   - An Introduction to Swing
   - Swing - Controls
   - Event Handling
   - Layout Managers

Swing Layout Managers 1/9

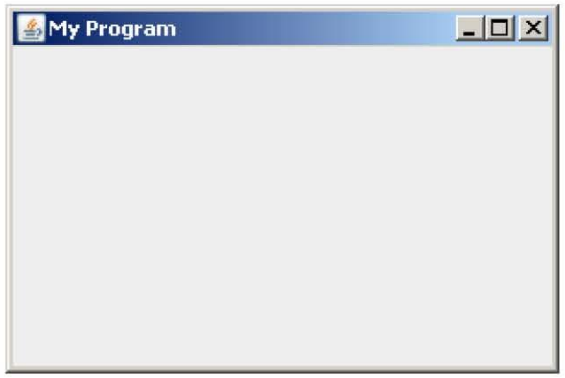


Swing Layout Managers 2/9



Swing Layout Managers 3/9

```
1. import  javax.swing.*;
2. public class SwingMy {
3.     public static void main(String[] args)
4.     {
5.         JFrame jfrm = new  JFrame("My Program");
6.         jfrm.setSize (300, 200);
7.         jfrm.setLocation (100, 200);
8.         jfrm.setVisible (true);
9.     }
10.}
```

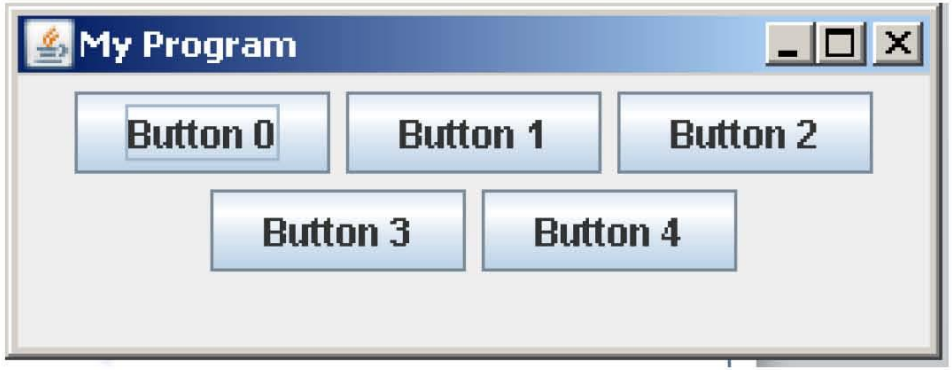


Swing Layout Managers 4/9

```
1. public static void main(String[] args){
2.     JFrame jfrm = new  JFrame("My Program");
3.     jfrm.setSize (300, 200);
4.     jfrm.setLocation (100, 200);
5.     jfrm.setVisible (true);
6.     JButton jbtn = new JButton("КНОПКА");
7.     jbtn.setSize(120,50);
8.     jfrm.add(jbtn);
9. }
```

Swing Layout Managers 5/9

```
JFrame jfrm = new  JFrame("My Program");
JPanel jpan = new JPanel();
jpan.setLayout ( new FlowLayout() );
for (int i=0; i<5; i++){
    jpan.add(new JButton("Button "+ i));
}
jfrm.add(jpan);
jfrm.setVisible(true);
```

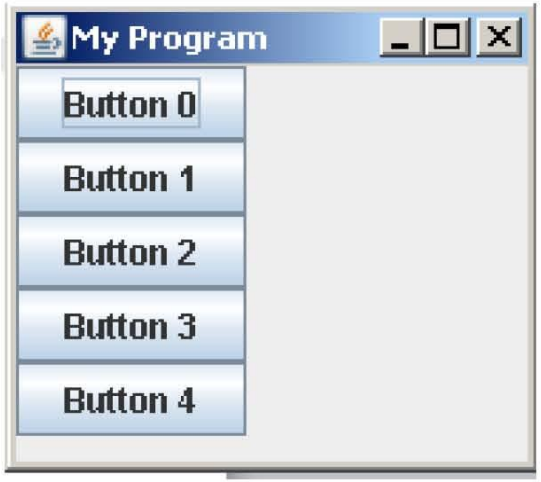


Swing Layout Managers 6/9

- JFrame jfrm = **new** JFrame(**"My Program"**);
- JPanel jpan = **new** JPanel();
- jpan.setLayout (**new** BorderLayout());
- jpan.add(**new** JButton(**"North"**), BorderLayout.***NORTH***);
- jpan.add(**new** JButton(**"South"**), BorderLayout.***SOUTH***);
- jpan.add(**new** JButton(**"West"**), BorderLayout.***WEST***);
- jpan.add(**new** JButton(**"East"**), BorderLayout.***EAST***);
- jpan.add(**new** JButton(**"Center"**), BorderLayout.***CENTER***);
- jfrm.add(jpan);
- jfrm.setVisible(**true**);

Swing Layout Managers 7/9

```
JFrame jfrm = new  JFrame("My Program");
JPanel jpan = new JPanel();
jpan.setLayout (
        new BoxLayout(jpan, BoxLayout.Y_AXIS));
for (int i=0; i<5; i++){
    jpan.add(new JButton("Button "+ i));
}
jfrm.add(jpan);
jfrm.setVisible(true);
```



Swing Layout Managers 8/9

```
JFrame jfrm = new  JFrame("My Program");
JPanel jpan = new JPanel();
jpan.setLayout ( new GridLayout(4, 3, 10, 10));
for (int i=0; i<12; i++) {
    jpan.add(new JButton("Button "+ i));
}
jfrm.add(jpan);
jfrm.setVisible(true);
```

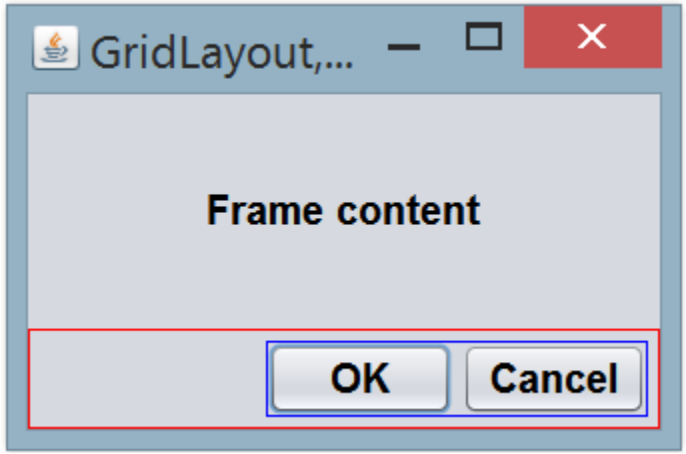
Swing Layout Managers 9/9

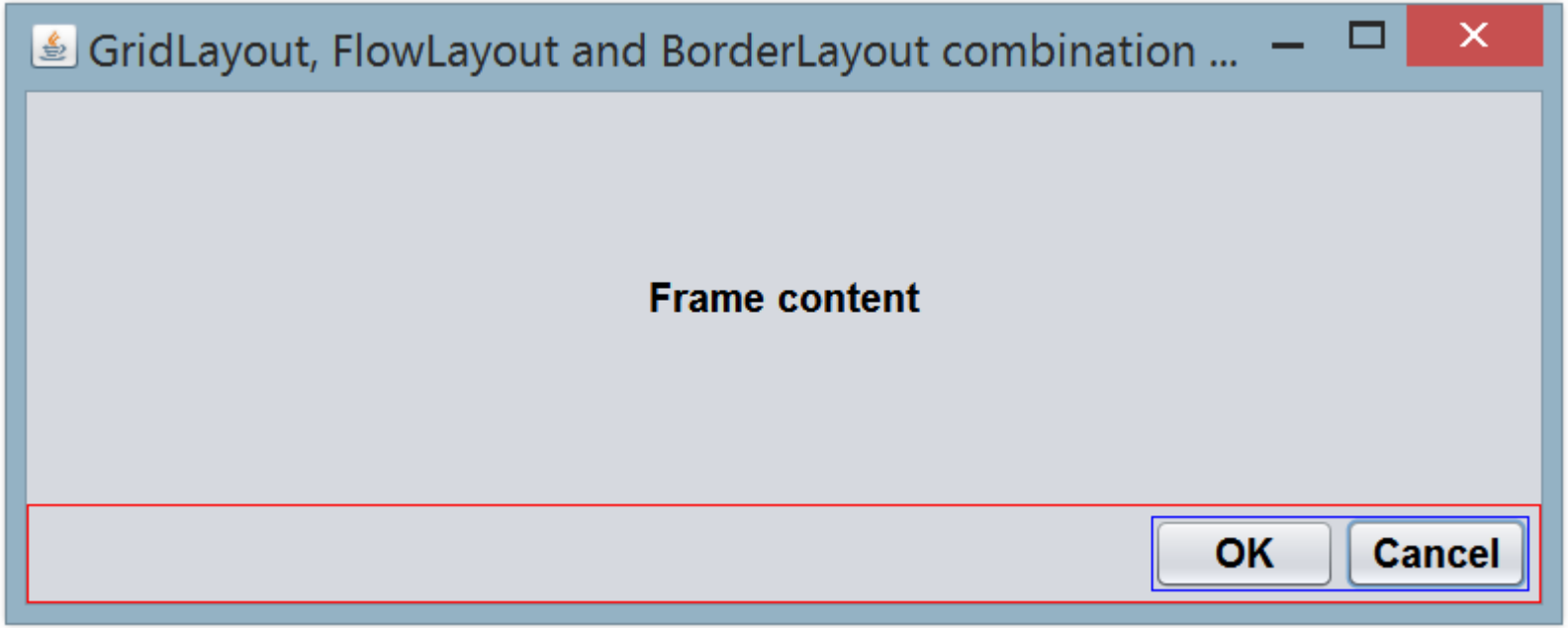
- JFrame jfrm = **new** JFrame(**"My Program"**);
- JPanel jpan = **new** JPanel();
- JButton btn = **new** JButton(**"Size 180x50"**);
- btn.setPreferredSize(**new** Dimension(180,50));
- jpan.add(btn);
- jfrm.add(jpan);
- jfrm.setVisible(**true**);

Composite Layout - 1

```
public class CompositeLayoutSample extends JFrame {

   public CompositeLayoutSample() {
      super("GridLayout, FlowLayout and BorderLayout combination sample");
      setSize(350, 250);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      JPanel grid = new JPanel(new GridLayout(1, 2, 5, 0));
      grid.setBorder(BorderFactory.createLineBorder(Color.blue));
      grid.add(new JButton("OK"));
      grid.add(new JButton("Cancel"));
      JPanel flow = new JPanel(new FlowLayout(FlowLayout.RIGHT));
      flow.setBorder(BorderFactory.createLineBorder(Color.red));
      flow.add(grid);
      Container contents = getContentPane();
      contents.add(flow, BorderLayout.PAGE_END);
      contents.add(new Label("Frame content", Label.CENTER));
      setVisible(true);
   }
   public static void main(String[] args) {
      CompositeLayoutSample app = new CompositeLayoutSample();
   } }
```

Composite Layout - 2





JTextField - 1

```
public class TextFieldDemo extends JFrame {

   JLabel jlabAll;
   JLabel jlabSelected;
   JLabel jlabCurPos;
   JTextField jtf;
   JButton jbtnCut;
   JButton jbtnCopy;
   JButton jbtnPaste;

   public TextFieldDemo() {
      setTitle("Работа с компонентом JTextField");
      setLayout(new FlowLayout());
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setSize(400, 200);
      jlabAll = new JLabel("Выделите часть текста в поле ");
      jlabSelected = new JLabel("и нажмите Enter");
      add(jlabAll);
      add(jlabSelected);
...
```

JTextField - 2

```
...
    jtf = new JTextField("Это тестовая фраза", 35);
     add(jtf);
     jtf.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
           jlabAll.setText("Полный текст в поле: " + jtf.getText());
           jlabSelected.setText("Выделенный текст: " + jtf.getSelectedText());
        }
     });
     jbtnCut = new JButton("Вырезать");
     jbtnCopy = new JButton("Копировать");
     jbtnPaste = new JButton("Вставить");
     add(jbtnCut);
     add(jbtnCopy);
     add(jbtnPaste);
     jbtnCut.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
           jtf.cut();
           jlabAll.setText("Полный текст в поле: " + jtf.getText());   ...
```

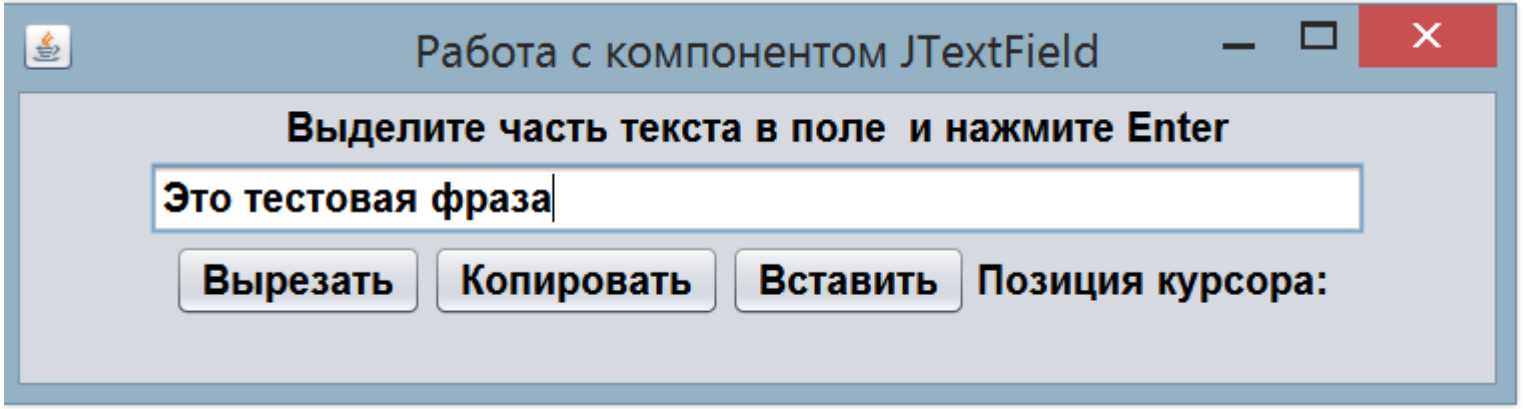
JTextField - 3

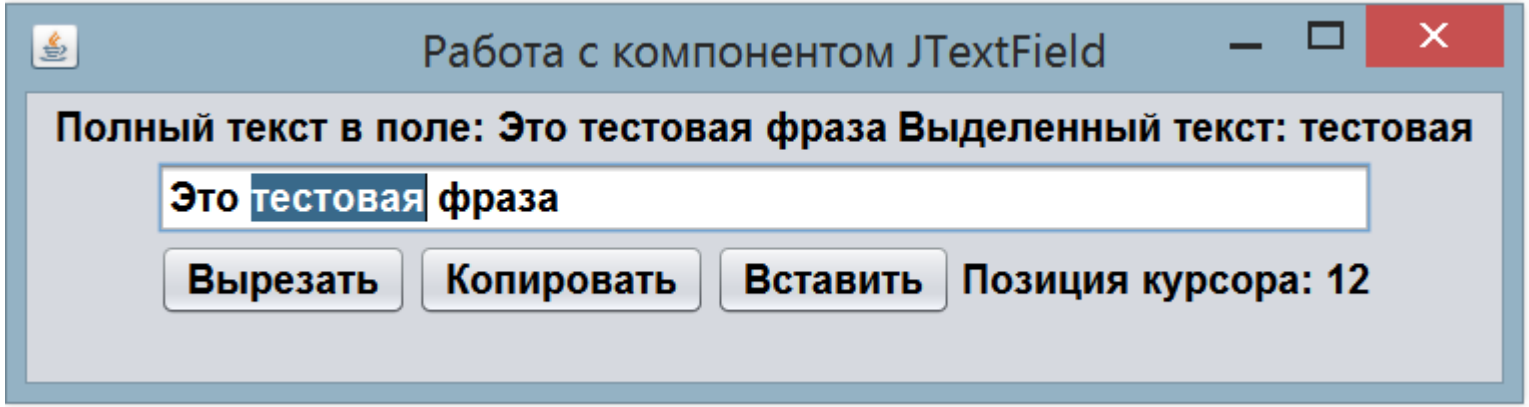
```
...
        JTextField buftf = new JTextField();
        buftf.paste();
        jlabSelected.setText("Текст в буфере обмена: " + buftf.getText());
    }
});
jbtnCopy.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        jtf.copy();
        JTextField buftf = new JTextField();
        buftf.paste();
        jlabSelected.setText("Текст в буфере обмена: " + buftf.getText());
    }
});
jbtnPaste.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        jtf.paste();
        jlabAll.setText("Полный текст в поле: " + jtf.getText());
...
```

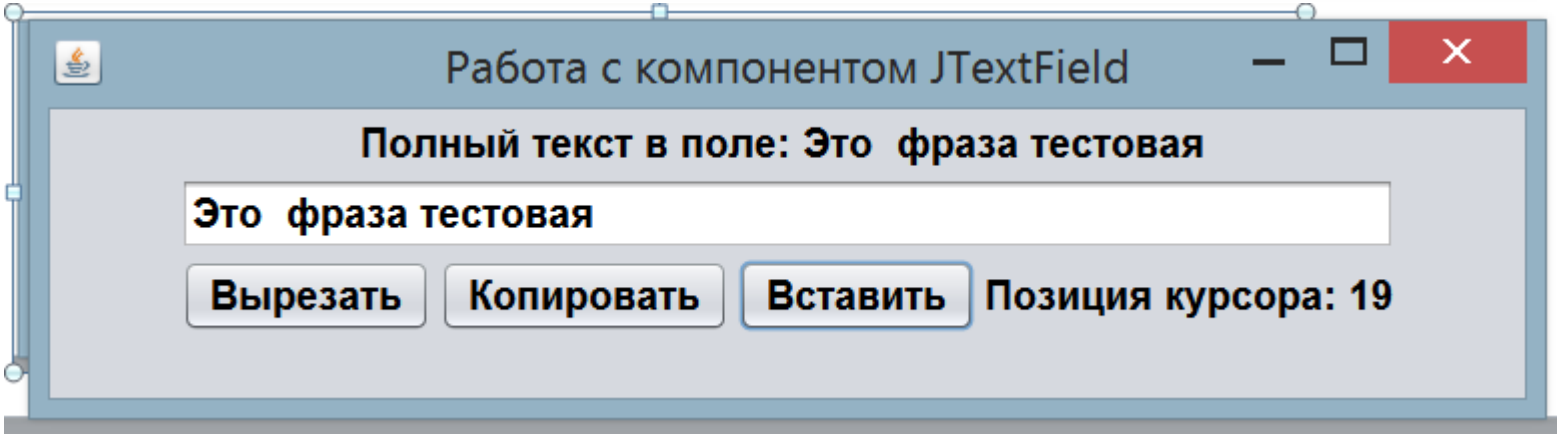
JTextField - 4

```
...
            jlabSelected.setText(""); //Выделенный текст показывать нет смысла
        }
    });
    jlabCurPos = new JLabel("Позиция курсора: ");
    jtf.addCaretListener(new CaretListener() {
        @Override
        public void caretUpdate(CaretEvent e) {
            jlabCurPos.setText("Позиция курсора: " + jtf.getCaretPosition());
        }
    });
    add(jlabCurPos);
    setVisible(true);
  }
  public static void main(String[] args){
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            TextFieldDemo tfd = new TextFieldDemo();
        }
    });
  }
}
```

JTextField - 5







JCheckBox and JRadioButton - 1

```
import java.awt.BorderLayout;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import javax.swing.*;

public class ComponentRadioButtonCheckBox extends JFrame {

    private final JPanel pnlChoose;
    private final JPanel pnlContent;
    private final JPanel pnlFruitContent;
    private final JPanel pnlVegetableContent;
    private final JCheckBox chkApple;
    private final JCheckBox chkGrape;
    private final JCheckBox chkPear;
    private final JCheckBox chkTomato;
    private final JCheckBox chkCucumber;
    private final JCheckBox chkPotato;
    private final JLabel lblInfo;
 …
```

JCheckBox and JRadioButton - 2

```
 …
public ComponentRadioButtonCheckBox() {
        super("JRadioButton and JCheckBox Using Example");
        setSize(600, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        /*Панель выбора фрукты или овощи - группа радиокнопок*/
        pnlChoose = new JPanel();
        JRadioButton rbtnFruits = new JRadioButton("Fruits");
        JRadioButton rbtnVegetables = new JRadioButton("Vegetables");
        ButtonGroup btngrChoose = new ButtonGroup();
        btngrChoose.add(rbtnFruits);
        btngrChoose.add(rbtnVegetables);
        pnlChoose.add(rbtnFruits);
        pnlChoose.add(rbtnVegetables);
        add(pnlChoose, BorderLayout.NORTH);
        pnlContent = new JPanel();
        add(pnlContent, BorderLayout.CENTER);
…
```

JCheckBox and JRadioButton - 3

```
...
    /*Панель выбора фруктов - чекбоксы*/
    pnlFruitContent = new JPanel();
    chkApple = new JCheckBox("Apple");
    chkGrape = new JCheckBox("Grape");
    chkPear = new JCheckBox("Pear");
    pnlFruitContent.add(chkApple);
    pnlFruitContent.add(chkGrape);
    pnlFruitContent.add(chkPear);
    /*Панель выбора овощей - чекбоксы*/
    pnlVegetableContent = new JPanel();
    chkTomato = new JCheckBox("Tomato");
    chkCucumber = new JCheckBox("Cucumber");
    chkPotato = new JCheckBox("Potato");
    pnlVegetableContent.add(chkTomato);
    pnlVegetableContent.add(chkCucumber);
    pnlVegetableContent.add(chkPotato);
...
```

JCheckBox and JRadioButton - 4

```
…
    /*Информационная строка - отображает список выбранных фруктов
      или овощей*/
    lblInfo = new JLabel("", JLabel.CENTER);
    add(lblInfo, BorderLayout.SOUTH);

    MyRadioButtonsListener radioButtonsListener =
                              new MyRadioButtonsListener();
    rbtnFruits.addItemListener(radioButtonsListener);
    rbtnVegetables.addItemListener(radioButtonsListener);

    MyCheckBoxListener fruitsCheckBoxListener =
           new MyCheckBoxListener(chkApple, chkGrape, chkPear);
    chkApple.addItemListener(fruitsCheckBoxListener);
    chkGrape.addItemListener(fruitsCheckBoxListener);
    chkPear.addItemListener(fruitsCheckBoxListener);
…
```

JCheckBox and JRadioButton - 5

```
…
    MyCheckBoxListener vegetablesCheckBoxListener =
      new MyCheckBoxListener(chkTomato, chkCucumber, chkPotato);
    chkTomato.addItemListener(vegetablesCheckBoxListener);
    chkCucumber.addItemListener(vegetablesCheckBoxListener);
    chkPotato.addItemListener(vegetablesCheckBoxListener);

    setVisible(true);
  }
  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
      @Override
      public void run() {
        new ComponentRadioButtonCheckBox();
      }
    });
  }
…
```

JCheckBox and JRadioButton - 6

```
…
  /**
   * Слушатель выбора радиокнопок фрукты или овощи.
   */
  private class MyRadioButtonsListener implements ItemListener {

      @Override
      public void itemStateChanged(ItemEvent e) {
          /*Очистка*/
          pnlContent.removeAll();
          lblInfo.setText("");
          chkApple.setSelected(false);
          chkGrape.setSelected(false);
          chkPear.setSelected(false);
          chkTomato.setSelected(false);
          chkCucumber.setSelected(false);
          chkPotato.setSelected(false);
…
```

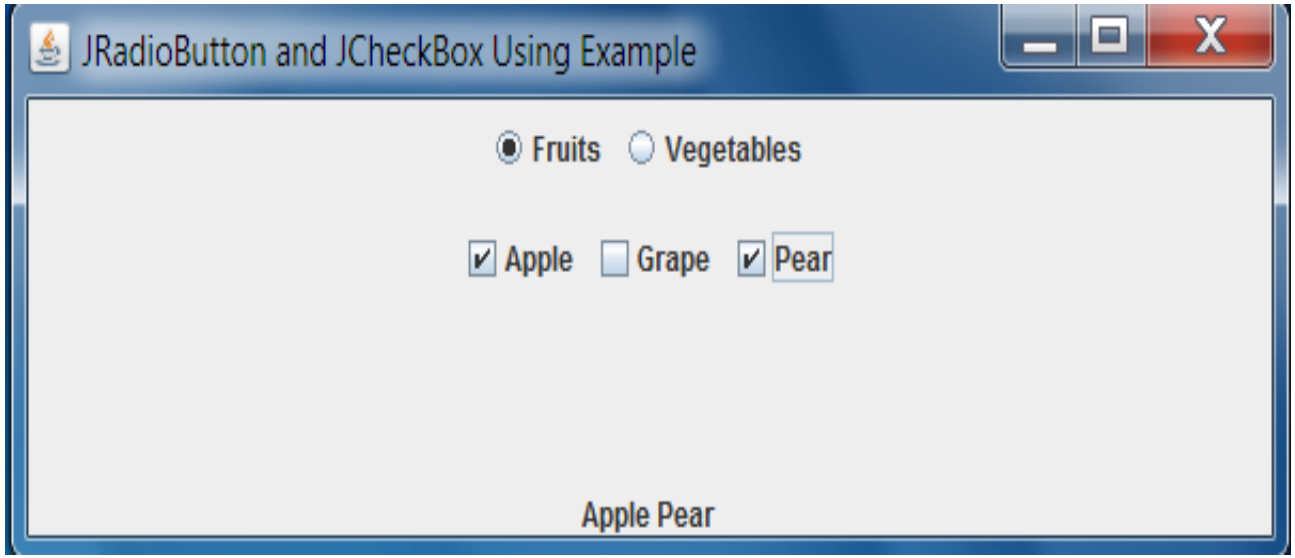
JCheckBox and JRadioButton - 7

```
...
    /*Перерисовка компонента - указывает Swing, что нужно
      удалить компоненты, удалённые методом remove() или
      removeAll()*/
    pnlContent.repaint();
    /*Указывает менеджеру компоновки на необходимость
      пересчёта компоновки, что необходимо при добавлении
      новых компонентов*/
    pnlContent.revalidate();
    /*Установка панели, соответствующей выбранной
      радиокнопке*/
    if ("Fruits".equals(((JRadioButton) e.getSource())
                                .getText())) {
       pnlContent.add(pnlFruitContent, BorderLayout.CENTER);
    } else {
       pnlContent.add(pnlVegetableContent,
                   BorderLayout.CENTER);
    }
    pnlContent.repaint();
    pnlContent.revalidate();
}       }    ...
```

JCheckBox and JRadioButton - 8

```
…
/**
    * Слушатель выбора овощей или фруктов.
    */
   private class MyCheckBoxListener implements ItemListener {
      private final JCheckBox[] items;

      public MyCheckBoxListener(JCheckBox… items) {
         this.items = items;
      }
      @Override
      public void itemStateChanged(ItemEvent e) {
         StringBuilder selected = new StringBuilder();
         for (JCheckBox item : items) {
            if (item.isSelected()) {
               selected.append(item.getText()).append(" ");
            }
         }
         lblInfo.setText(selected.toString());
      }
   }  }
```

JCheckBox and JRadioButton - 9



JList - 1

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

public class ComponentList {

    public ComponentList() {
        final JFrame jfrm = new JFrame("Компонент JList");
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jfrm.setSize(500, 300);
        final JLabel jlab = new JLabel();
        jfrm.add(jlab);
        /*Массив с данными списка (вместо массива также могут
          использоваться вектор и модель списка)*/
        final String[] data = {"one", "two", "three", "four", "five",
                               "six", "seven", "eight", "nine", "ten"};
        /*При создании списка передаём в конструктор массив строк*/
        final JList datalist = new JList(data);
…
```

JList - 2

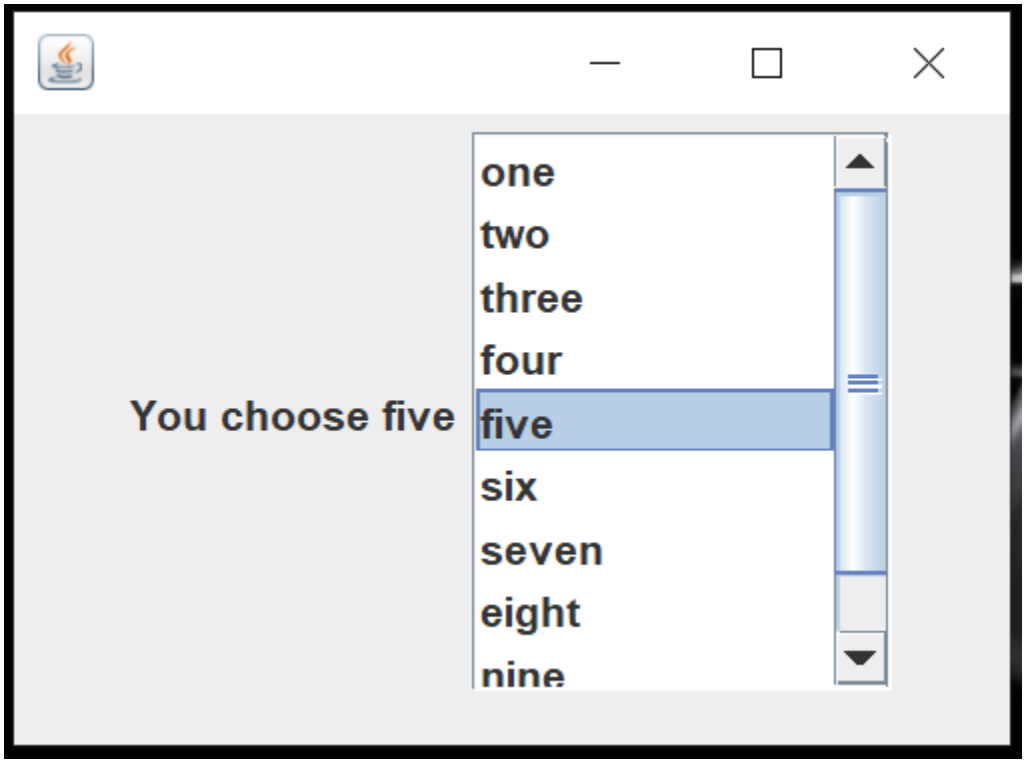
```
…
    /*Установка режима, позволяющего выбирать только один элемент
      списка, ещё варианты SINGLE_INTERVAL_SELECTION
      и MULTIPLE_INTERVAL_SELECTION*/
    datalist.setSelectionMode(ListSelectionModel
                              .SINGLE_SELECTION);
    /*Для отображения списка в прокручиваемом окне создаём объект
      JScrollPane, в конструктор которого передаём созданный
      список*/
    JScrollPane jscpane = new JScrollPane(datalist);
    /*Установка размера прокручиваемой панели*/
    jscpane.setPreferredSize(new Dimension(120, 160));
    jfrm.add(jscpane);
…
```

JList - 3

```
...
/*Создаём слушатель события выбора элемента в списке
  и реализуем абстрактный метод valueChanged класса
  ListSelectionListener, получающий событие
  ListSelectionEvent*/
ListSelectionListener lsldatalist =
                        new ListSelectionListener() {
   @Override
   public void valueChanged(ListSelectionEvent e) {
      /*Получаем индекс выбранного элемента, если ни один
        элемент не выбран возвращается -1*/
      int i = datalist.getSelectedIndex();
      if (i != -1) {
         jlab.setText("You choose " + data[i]);
      } else {
         jlab.setText("You don't choose list item");
      }
   }
};
datalist.addListSelectionListener(lsldatalist);
jfrm.setVisible(true);     }   …
```

JList - 4

```
...
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {

            @Override
            public void run() {
                ComponentList comlst = new ComponentList();
            }
        });
    }
}
```



JComboBox - 1

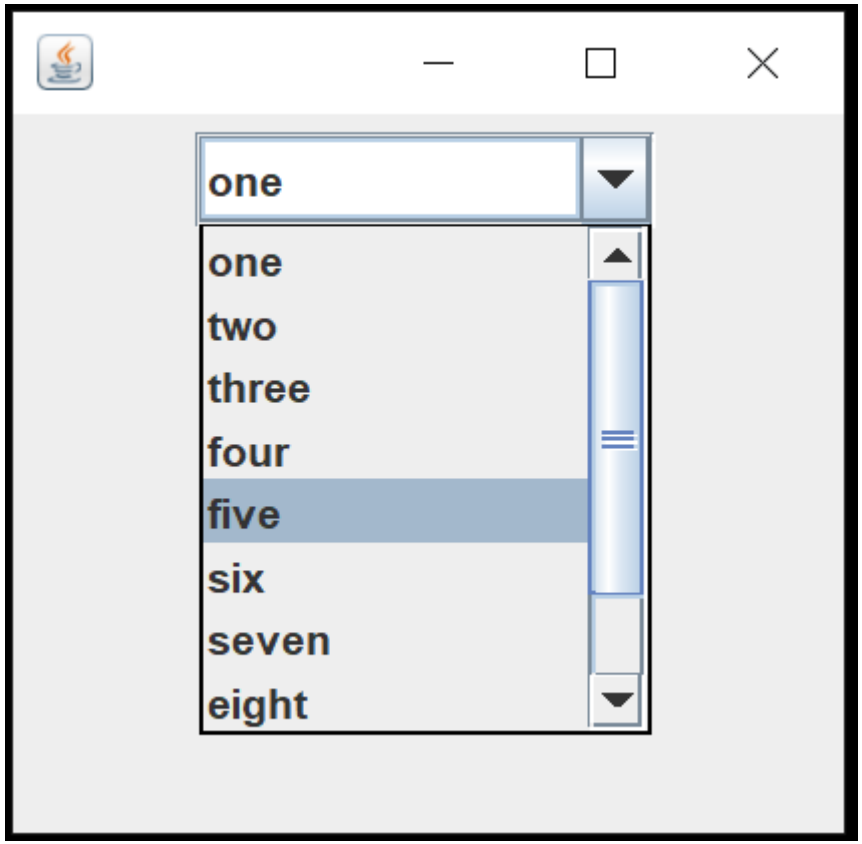
```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class ComponentComboBox extends JFrame
                              implements ActionListener {

   private final JLabel jlab;

   public ComponentComboBox() {
      super("Компонент JComboBox");
      setSize(600, 400);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JPanel pnlRoot = new JPanel();
      /*Массив с данными списка (вместо массива также могут
        использоваться вектор и модель списка)*/
      final String[] data = {"one", "two", "three", "four", "five",
                        "six", "seven", "eight", "nine", "ten"};
      JComboBox comboBox = new JComboBox(data);
…
```

JComboBox - 2

```
…
    /*Делаем компонент редактируемым*/
    comboBox.setEditable(true);
    comboBox.addActionListener(this);
    JScrollPane scrollPane = new JScrollPane(comboBox);
    pnlRoot.add(scrollPane);
    add(pnlRoot, BorderLayout.NORTH);
    jlab = new JLabel("", JLabel.CENTER);
    add(jlab, BorderLayout.SOUTH);
    setVisible(true);
  }

  @Override
  public void actionPerformed(ActionEvent e) {
    JComboBox jcb = (JComboBox) e.getSource();
    String item = (String) jcb.getSelectedItem();
    jlab.setText("Ваш выбор: " + item);
  }
…
```

JComboBox - 3

```
…
   public static void main(String[] args) {
      SwingUtilities.invokeLater(new Runnable() {
         @Override
         public void run() {
            new ComponentComboBox();
         }
      });
   }
}
```



JMenuBar, JMenu and JMenuItem - 1

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.io.*;
import javax.swing.*;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;

public class SimpleMenu extends JFrame {
   static JTextArea info = new JTextArea(500, 300);
   public SimpleMenu () {
      setTitle("Работа с меню и его элементами");
      setSize(600, 400);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      add(info, BorderLayout.CENTER);
      JMenuBar menuBar = new JMenuBar();
      JMenu file = new JMenu("File");
      file.setMnemonic(KeyEvent.VK_F);
      JMenuItem exit = new JMenuItem("Exit");   …
```

JMenuBar, JMenu and JMenuItem - 2

…

```
exit.setMnemonic(KeyEvent.VK_X);
exit.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});
JMenuItem open = new JMenuItem("Open");
open.setMnemonic(KeyEvent.VK_O);
open.addActionListener(new OpenActionListener());

JMenuItem saveAs = new JMenuItem("Save As");
saveAs.setMnemonic(KeyEvent.VK_S);
saveAs.addActionListener(new MenuActionListener(info));

file.add(open);
file.add(saveAs);
file.addSeparator();
file.add(exit);
menuBar.add(file);    …
```

JMenuBar, JMenu and JMenuItem - 3

```
…
/*Устанавливаем панель меню в форму*/
      setJMenuBar(menuBar);
      setVisible(true);
   }

   private static class OpenActionListener implements ActionListener  {
      @Override
      public void actionPerformed(ActionEvent e) {
         File currentDirectory =
                     new File(System.getProperty("user.dir"));
         JFileChooser fileChooser =
                           new JFileChooser(currentDirectory);
         FileFilter filter =
               new FileNameExtensionFilter("Text files", "txt");
         fileChooser.setFileFilter(filter);
         int result = fileChooser
                  .showOpenDialog((JMenuItem) e.getSource());
…
```

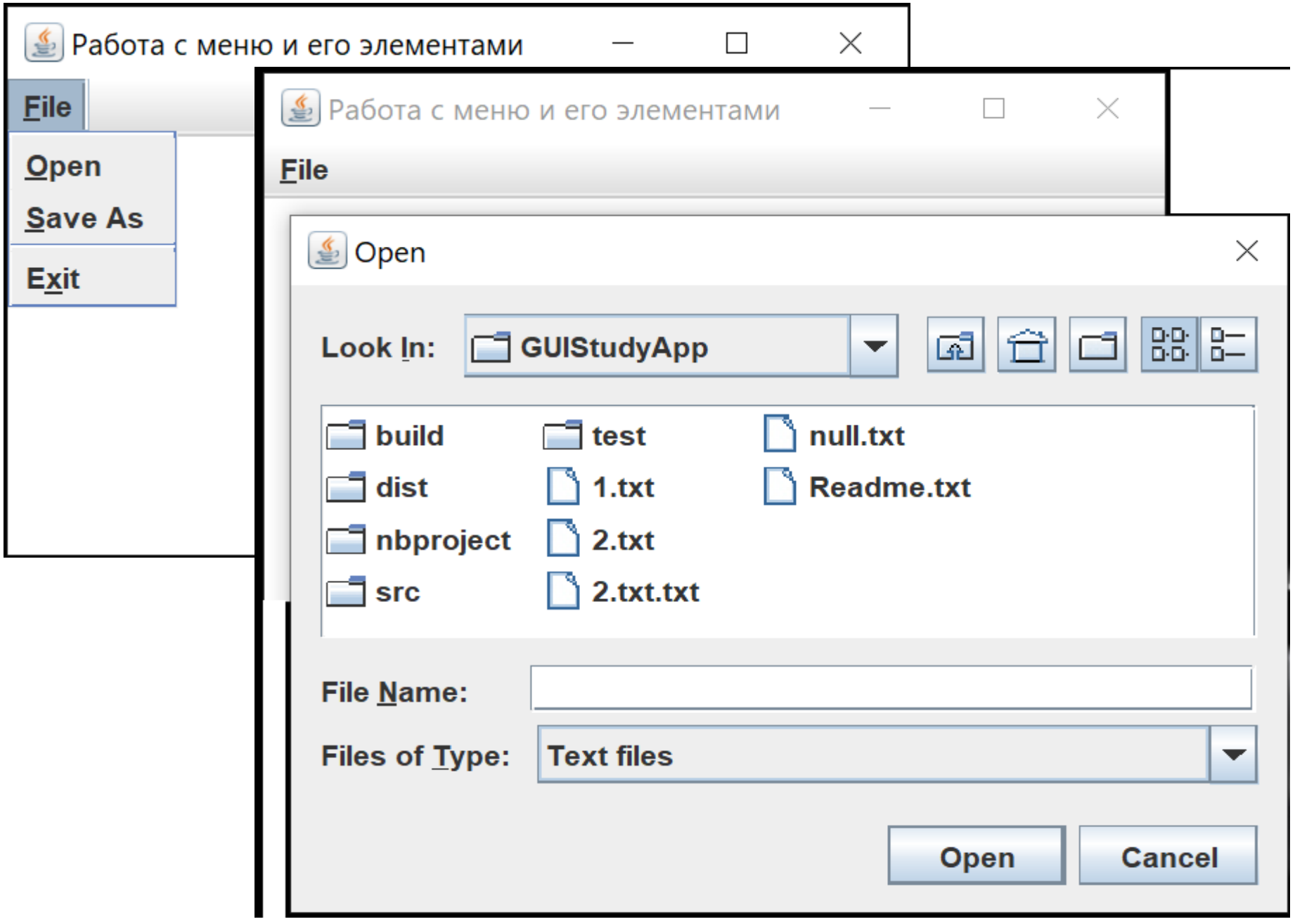
JMenuBar, JMenu and JMenuItem - 4

```
...
        if (result == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            try (BufferedReader br = new BufferedReader(new
  InputStreamReader(new FileInputStream(selectedFile), "CP1251"))) {
                StringBuilder sb = new StringBuilder();
                String s;
                while ((s = br.readLine()) != null) {
                    sb.append(s+"\n");
                }
                info.setText(sb.toString());
            } catch (IOException ex) {
                System.out.println(ex.getMessage());
            }
        }
    }
}
...
```

JMenuBar, JMenu and JMenuItem - 5

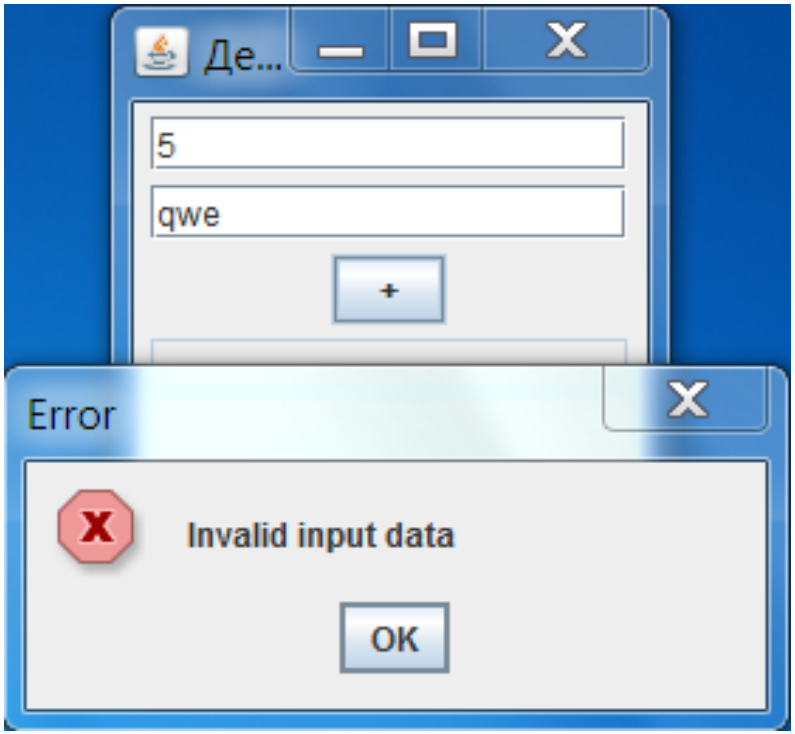
```
…
private class MenuActionListener implements ActionListener {
   JTextArea info;
   public MenuActionListener(JTextArea info) {
         this.info = info;
      }

      @Override
      public void actionPerformed(ActionEvent e) {
         String item = e.getActionCommand();
         info.setText(item);
      }
   }
   public static void main(String[] args) {
      SwingUtilities.invokeLater(new Runnable() {
         @Override
         public void run() {
            new SimpleMenu();
         }
      });
   }  }
```

JMenuBar, JMenu and JMenuItem - 6



JOptionPane - 1

```
public void actionPerformed(ActionEvent e) {
      try {
         double num1 = Double.parseDouble(tfNum1.getText());
         double num2 = Double.parseDouble(tfNum2.getText());
         tfRes.setText(String.valueOf(num1 + num2));
      } catch (NumberFormatException ex) {
         JOptionPane.showMessageDialog(this, "Invalid input data",
                  "Error", JOptionPane.ERROR_MESSAGE);
      }
   }
```



JOptionPane - 2

messageType:

 - JOptionPane.QUESTION_MESSAGE

 - JOptionPane.INFORMATION_MESSAGE

 - JOptionPane.WARNING_MESSAGE

 - JOptionPane.ERROR_MESSAGE

без иконки - JOptionPane.PLAIN_MESSAGE

```
int showConfirmDialog(Component parentComponent, Object
message, String title, int optionType, int messageType,
Icon icon)
int showOptionDialog(Component parentComponent, Object
message, String title, int optionType, int messageType,
Icon icon, Object[] options, Object initialValue)
int showInternalConfirmDialog(Component
parentComponent, Object message, String title, int
optionType, int messageType, Icon icon)
Object showInputDialog(Component parentComponent,
Object message, String title, int messageType, Icon icon,
Object[] selectionValues, Object initialSelectionValue)
…
```

JOptionPane - 3

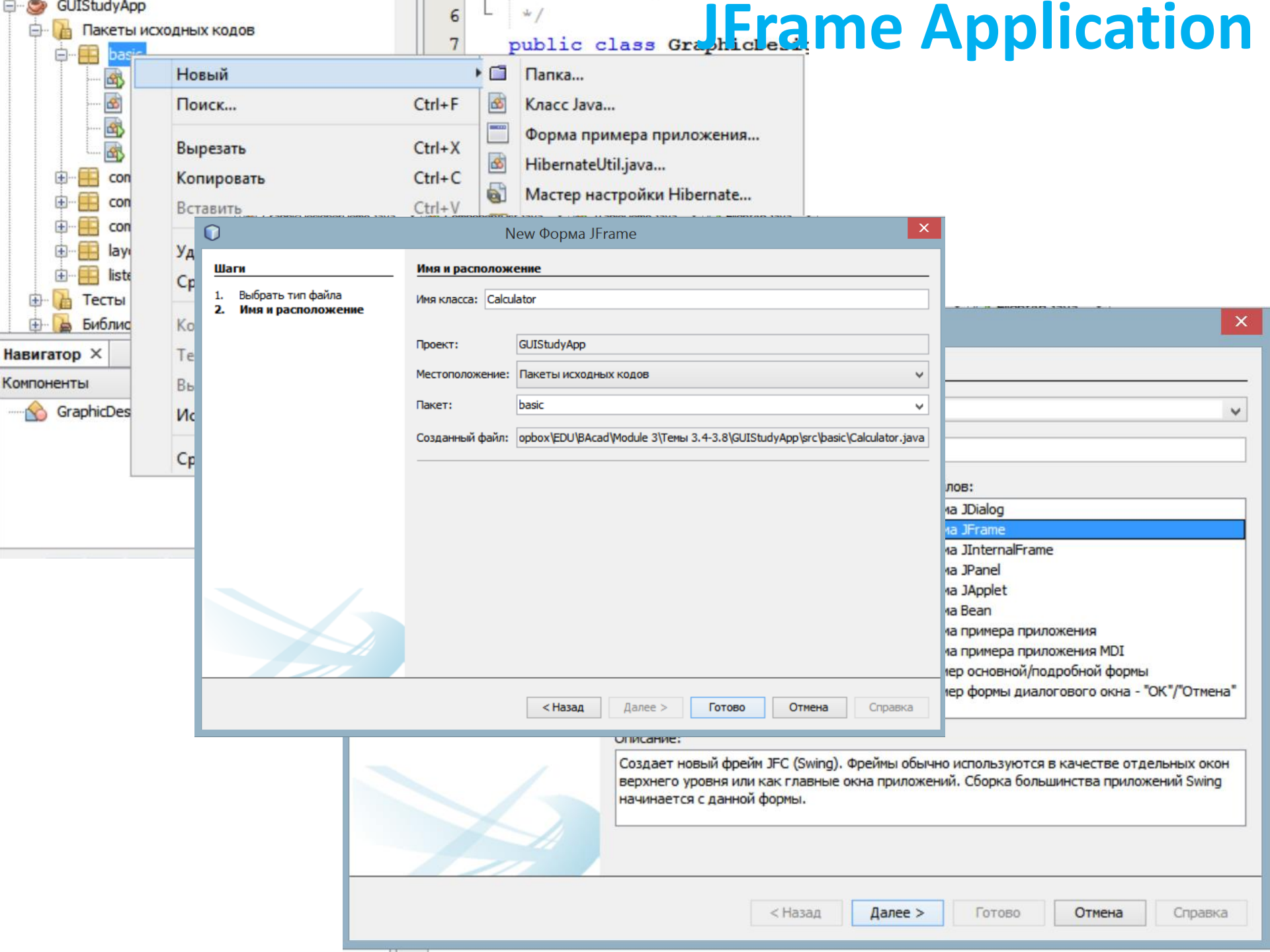
…

```
Object showInternalInputDialog(Component
parentComponent, Object message, String title, int
messageType, Icon icon, Object[] selectionValues, Object
initialSelectionValue)
```

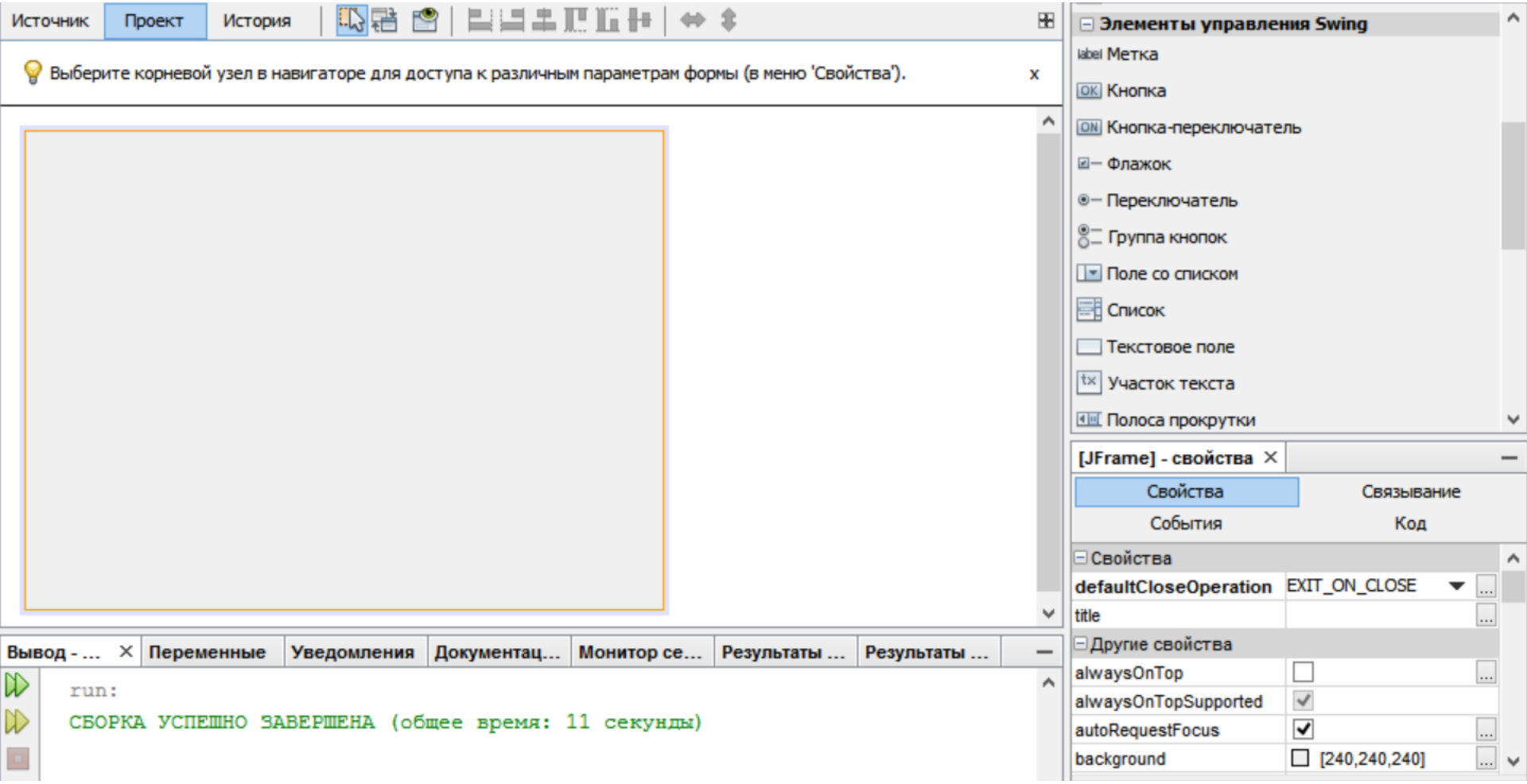
optionType:

```
JOptionPane.DEFAULT_OPTION
JOptionPane.YES_NO_OPTION
JOptionPane.YES_NO_CANCEL_OPTION
JOptionPane.OK_CANCEL_OPTION
```

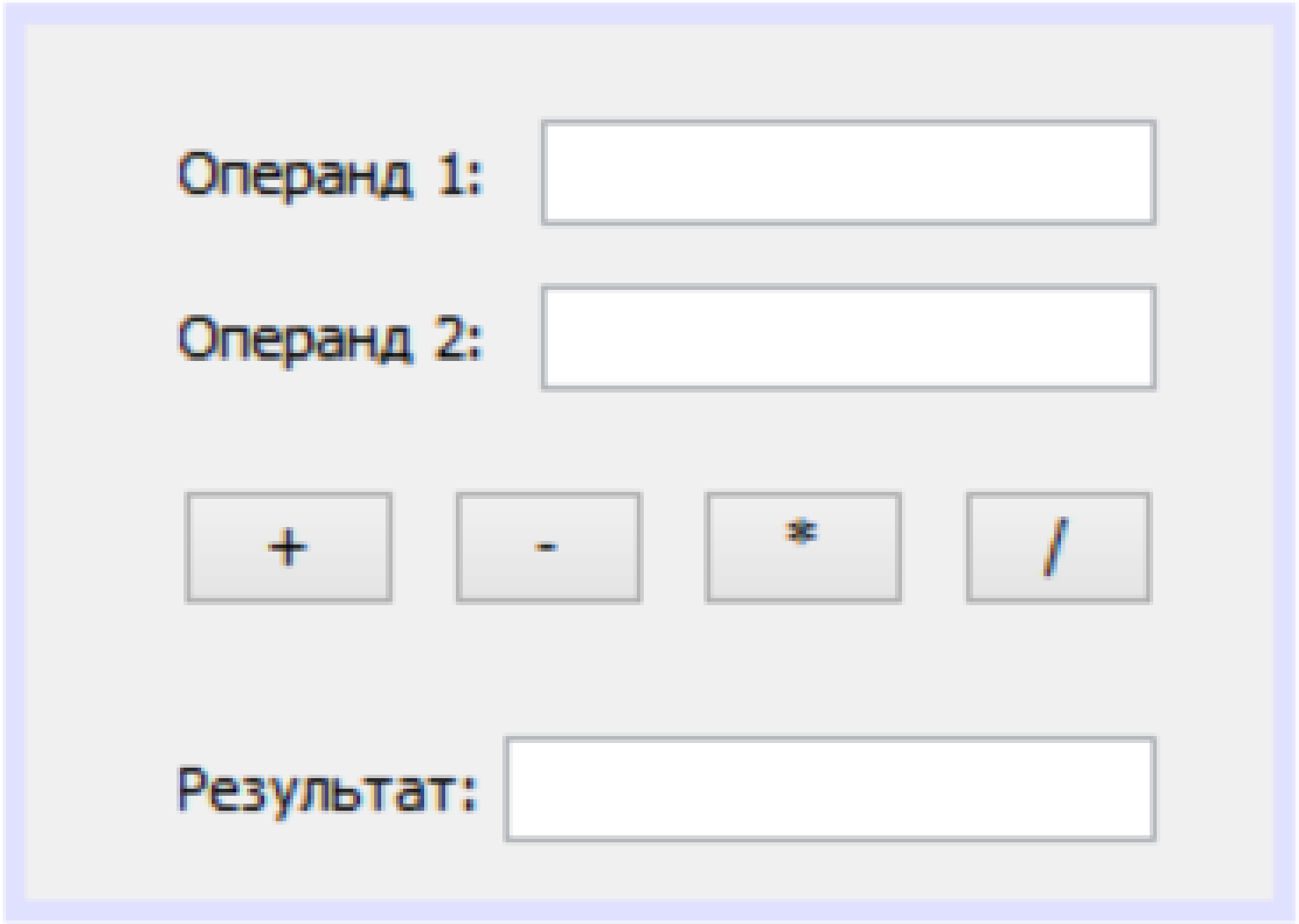
JFrame Application



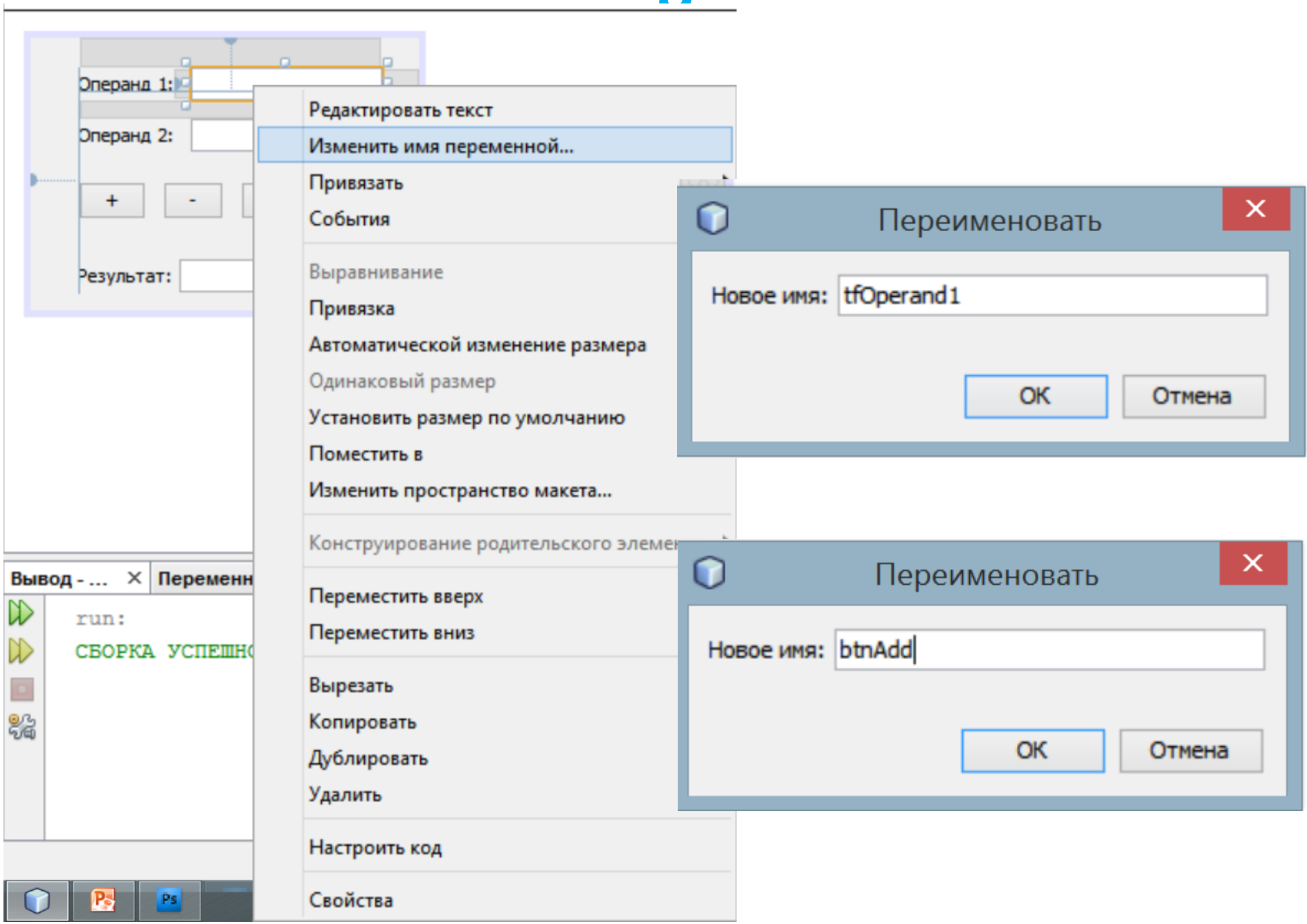
GUI Designer - 1



GUI Designer - 2



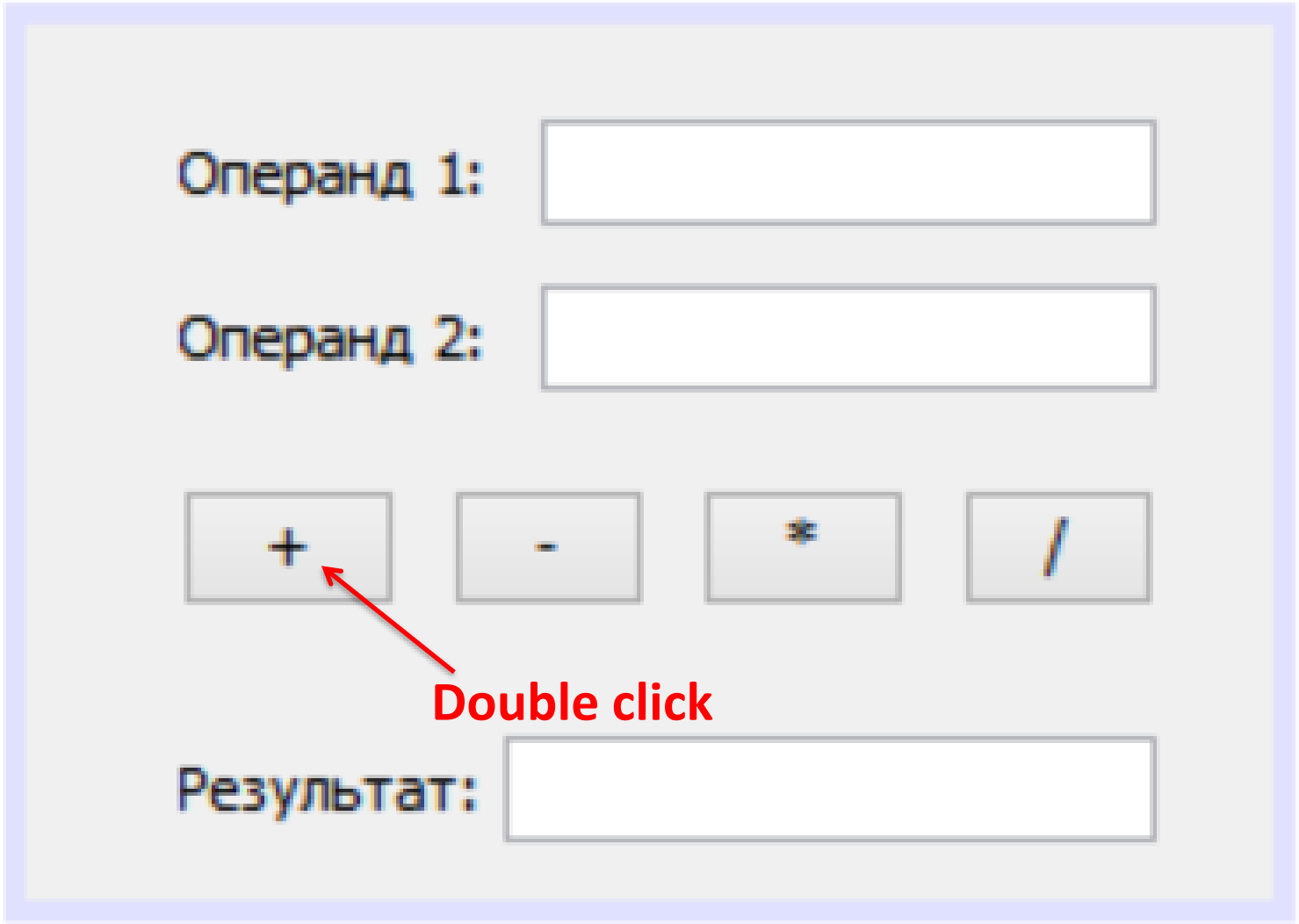
GUI Designer - 3



GUI Designer - 4

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

    public Calculator() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    private void initComponents() {
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        tfOperand1 = new javax.swing.JTextField();
        tfOperand2 = new javax.swing.JTextField();
        btnAdd = new javax.swing.JButton();
        btnSub = new javax.swing.JButton();
        btnMul = new javax.swing.JButton();
        btnDiv = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
        tfResult = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("Операнд 1:");      ...
```

GUI Designer - 5



GUI Designer - 6

```
 private void btnAddActionPerformed(java.awt.event.ActionEvent
evt) {
        try {
            double oper1 = Double.parseDouble(tfOperand1.getText());
            double oper2 = Double.parseDouble(tfOperand2.getText());
            tfResult.setText(String.valueOf(oper1 + oper2));
        } catch (NumberFormatException ex) {
            tfResult.setText("Wrong input!!!");
        }
    }
```

GUI Designer - 7