ПРАКТИЧЕСКАЯ РАБОТА №4. СОЗДАНИЕ GUI. СОБЫТИЙНОЕ ПРОГРАММИРОВАНИЕ В JAVA.

Цель работы: введение в событийное программирование на языке Java.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Данная практическая работа посвящена закреплению практических навыков по созданию приложений на Java с использованием следующих элементов GUI:

- Текстовые поля и области ввода текста;
- Менеджеры компоновки компонентов;
- Слушатель мыши;
- Создание меню.

Text Fields - текстовое поле или поля для ввода текста (можно ввести только одну строку). Примерами текстовых полей являются поля для ввода логина и пароля, например, используемые, при входе в электронную почту.

Пример создания объекта класса JTextField:

JTextField jta = new JTextField (10);

В параметрах конструктора задано число 10, это количество символов, которые могут быть видны в текстовом поле. Текст веденный в поле JText может быть возвращен с помощью метода getText(). Также в поле можно записать новое значение с помощью метода setText(String s).

Как и у других компонентов, мы можем изменять цвет и шрифт текста в текстовом поле.

Пример 1.

```
class LabExample extends JFrame

{

      JTextField jta = new JTextField(10);

      Font fnt = new Font("Times new roman",Font.BOLD,20);

      LabExample()

      {

            super("Example");

            setLayout(new FlowLayout());

            setSize(250,100);
```

```
            add(jta);
            jta.setForeground(Color.PINK);
            jta.setFont(fnt);

            setVisible(true);
      }
            public static void main(String[]args)
      {
            new LabExample();
      }
}
```

Рисунок 4.1

Важная замечание

Ответственность за выполнение проверки на наличие ошибок в коде лежит полностью на программисте, например, чтобы проверить произойдет ли ошибка, когда в качестве входных данных в JTextField ожидается ввод числа. Компилятор не будет ловить такого рода ошибку, поэтому ее необходимо обрабатывать пользовательским кодом.

Выполните следующий пример и наблюдайте за результатом, когда число вводится в неправильном формате:

Пример 2.

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

class LabExample extends JFrame
{
      JTextField jta1 = new JTextField(10);
      JTextField jta2 = new JTextField(10);
```

```
      JButton button = new JButton(" Add them up");


      Font fnt = new Font("Times new roman",Font.BOLD,20);


      LabExample()
      {
            super("Example");
            setLayout(new FlowLayout());
            setSize(250,150);
            add(new JLabel("1st Number"));
            add(jta1);
            add(new JLabel("2nd Number"));
            add(jta2);
            add(button);
            button.addActionListener(new ActionListener()
            {
                  public void actionPerformed(ActionEvent ae)
                  {
                        try
                        {
                              double x1 =
Double.parseDouble(jta1.getText().trim());
                              double x2 =
Double.parseDouble(jta2.getText().trim());


                              JOptionPane.showMessageDialog(null, "Result =
"+(x1+x2),"Alert",JOptionPane.INFORMATION_MESSAGE);


                        }
                        catch(Exception e)
                        {
                               JOptionPane.showMessageDialog(null, "Error in
Numbers !","alert" , JOptionPane.ERROR_MESSAGE);
                        }
                  }
            });
            setVisible(true);
      }
```

```
	public static void main(String[]args)
	{
		new LabExample();
	}
}
```

JTextArea

Компонент TextArea похож на TextField, но в него можно вводить более одной строки. В качестве примера TextArea можно рассмотреть текст, который мы набираем в теле сообщения электронной почты.

Пример 3.

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

class TextAreaExample extends JFrame
{
	JTextArea jta1 = new JTextArea(10,25);
	JButton button = new JButton("Add some Text");
	public TextAreaExample()
	{
		super("Example");
		setSize(300,300);
		setLayout(new FlowLayout());
		add(jta1);
		add(button);
		button.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent ae)
			{
				String txt = JOptionPane.showInputDialog(null,"Insert
some text");
				jta1.append(txt);
			}
		});
```

```
	}
	public static void main(String[]args)
	{
		new TextAreaExample().setVisible(true);
	}
}
```

Замечание.

Мы можем легко добавить возможность прокрутки к текстовому полю, добавив его в контейнер с именем `JScrollPane` следующим образом:

```
JTextArea txtArea = new JTextArea(20,20)
JScrollPane jScroll = new JScrollPane(txtArea);
// …
add(Scroll); // we add the scrollPane and not the text area.
```

Попробуйте выполнить сами!

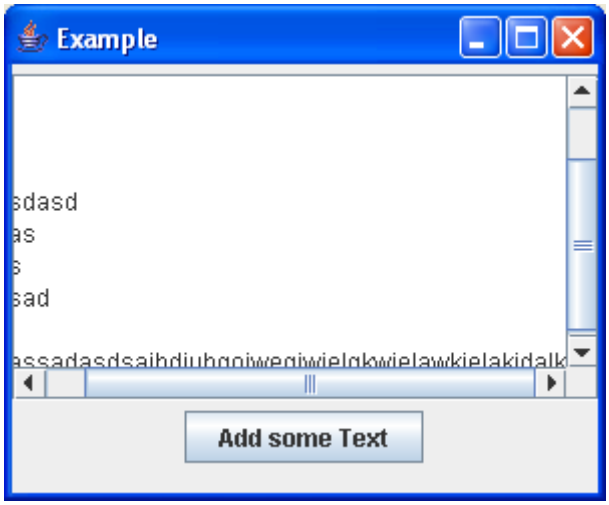

Рисунок 4.2

Менеджеры компоновки компонентов или Layout Менеджеры.

Менеджер BorderLayout:

Разделяет компонент на пять областей (WEST, EAST, NOTH, SOUTH and Center). Другие компоненты могут быть добавлены в любой из этих компонентов пятерками.

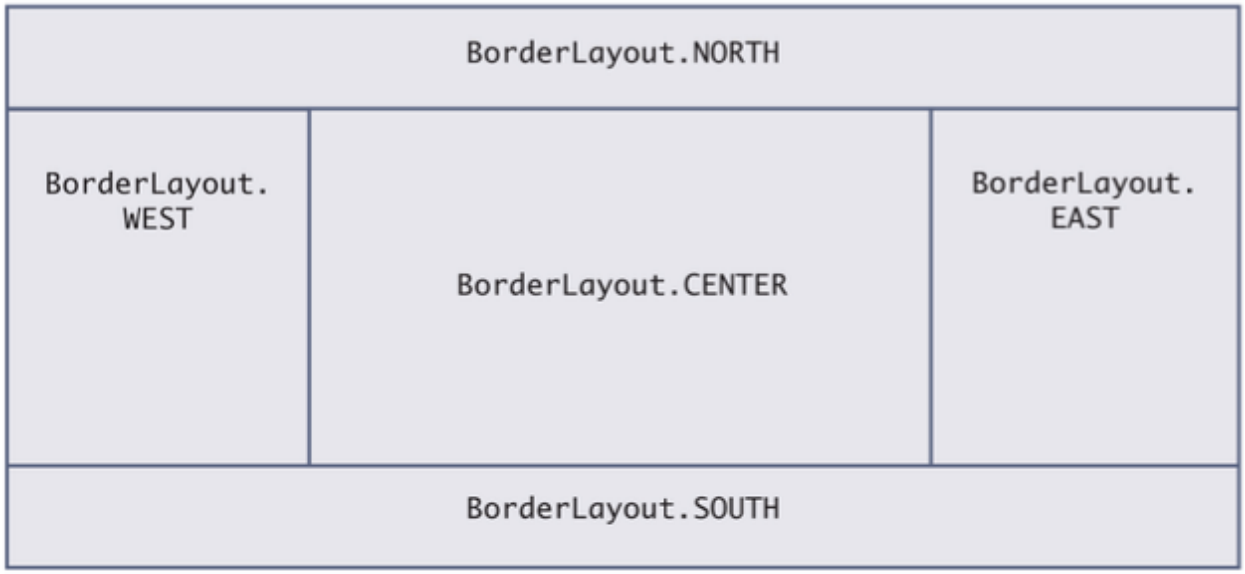

Рисунок 4.3

Метод для добавления в контейнер, который есть у менеджера BorderLayout отличается и выглядит следующим образом:

```
add(comp, BorderLayout.EAST);
```

Обратите внимание, что мы можем, например, добавить панели JPanel в эти области и затем добавлять компоненты этих панелей. Мы можем установить расположение этих JPanel используя другие менеджеры.

Менеджер GridLayout.

С помощью менеджера GridLayout компонент может принимать форму таблицы, где можно задать число строк и столбцов.

Таблица 1.

1	2	3	4
5	6	7	8
9	10	11	12

Если компоненту GridLayout задать 3 строки и 4 столбца, то компоненты будут принимать форму таблицы, показанной выше, и будут всегда будут добавляться в порядке их появления.

Следующий пример иллюстрирует смесь компоновки различных компонентов.

Пример 4.

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

class BorderExample extends JFrame
{
      JPanel[] pnl = new JPanel[12];

      public BorderExample()
      {
            setLayout(new GridLayout(3,4));
            for(int i = 0 ; i < pnl.length ; i++)
            {
                  int r = (int) (Math.random() * 255);
                  int b = (int) (Math.random() * 255);
                  int g = (int) (Math.random() * 255);
                  pnl[i] = new JPanel();
                  pnl[i].setBackground(new Color(r,g,b));
                  add(pnl[i]);
            }

            pnl[4].setLayout(new BorderLayout());
            pnl[4].add(new JButton("one"),BorderLayout.WEST);
            pnl[4].add(new JButton("two"),BorderLayout.EAST);
            pnl[4].add(new JButton("three"),BorderLayout.SOUTH);
            pnl[4].add(new JButton("four"),BorderLayout.NORTH);
            pnl[4].add(new JButton("five"),BorderLayout.CENTER);

            pnl[10].setLayout(new FlowLayout());
            pnl[10].add(new JButton("one"));
```

```
            pnl[10].add(new JButton("two"));
            pnl[10].add(new JButton("three"));
            pnl[10].add(new JButton("four"));
            pnl[10].add(new JButton("fve"));

            setSize(800,500);
      }
      public static void main(String[]args)
      {
            new BorderExample().setVisible(true);
      }
}
```

Код представленный выше, будет иметь вид как на рисунке ниже.

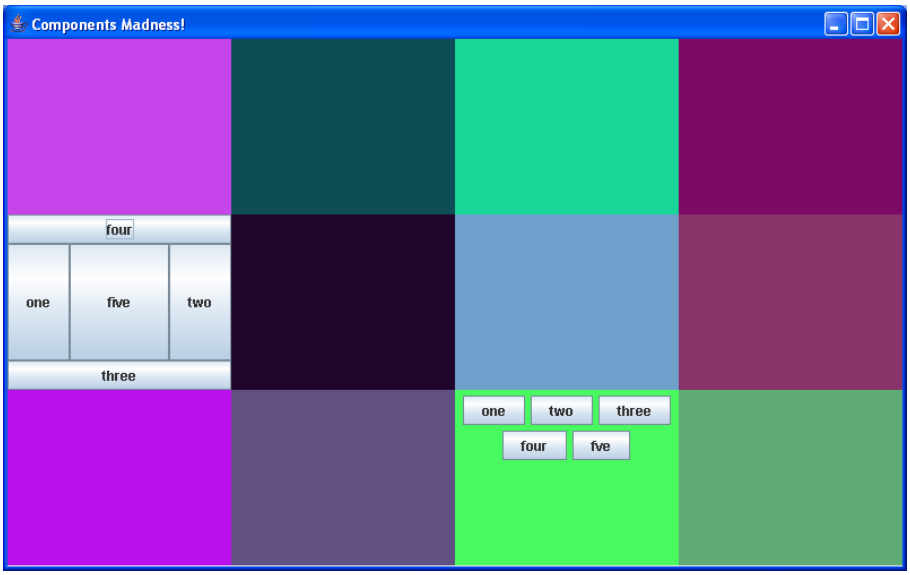

Рисунок 4.4

Заметьте, что JFrame имеет GridLayout размера 3 на 4 (таблица), в то время как JPanel размером (2, 1) имеет менеджер BorderLayout. А JPanel (3, 3) имеет FLowLayout.

Менеджер Null Layout Manager.

Иногда бывает нужно изменить размер и расположение компонента в контейнере. Таким образом, мы должны указать программе не использовать

никакой менеджер компоновки, то есть (setLayout (нуль)). Так что мы получим что-то вроде этого:

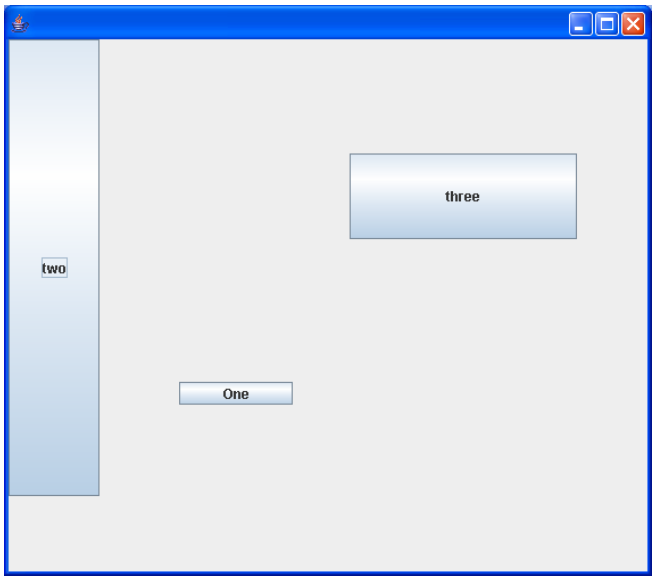

Рисунок 4.5

Пример 5.

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

class NullLayout extends JFrame
{
      JButton but1 = new JButton("One");;
      JButton but2 = new JButton("two");;
      JButton but3 = new JButton("three");;

      public NullLayout()
      {
            setLayout(null);
            but1.setBounds(150,300,100,20); // added at 150,300 width = 100,
height=20
            but2.setSize(80,400); // added at 0,0 width = 80, height=400
            but3.setLocation(300,100);
```

```
            but3.setSize(200,75);
            // those two steps can be combined in one setBounds method call
            add(but1);
            add(but2);
            add(but3);
            setSize(500,500);
      }
      public static void main(String[]args)
      {
            new NullLayout().setVisible(true);
      }
}
```

Слушатель событий мыши MouseListener.

Мы можем реализовывать слушателей мыши и также слушателей клавиатуры на компонентах GUI. Интерфейс MouseListener имеет следующие методы:

Таблица 1. Методы класса MouseListener.

Методы класса		
Возвращаемое значение	Прототип метода	Описание
`void`	**`mouseClicked`**`(MouseEvent e)`	Вызывается, когда кнопка мыши была нажата (нажата и отпущена) на области компонента.
`void`	**`mouseEntered`**`(MouseEvent e)`	Вызывается, когда мышь входит в область компонент.
`void`	**`mouseExited`**`(MouseEvent e)`	Вызывается, когда мышь выходит из области компонента.

void	**mousePressed**(MouseEvent e)	Вызывается при нажатии кнопки мыши на область компонента.
void	**mouseReleased**(MouseEvent e)	Вызывается, когда над областью компонента отпущена кнопка мыши.

Слушатель мыMouseListener можно добавить к компоненту следующим образом:

Component.addMouseListener(listener);

Здесь слушатель является экземпляром класса, который реализует интерфейс MouseListener. Обратите внимание, что он должен обеспечивать выполнение всех методов, перечисленных в таблице 1 в данной практической работе.

Пример 6.

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

class MyMouse extends JFrame
{
	JLabel lbl = new JLabel("");
	public MyMouse()
	{
		super("Dude! Where's my mouse ?");
		setSize(400,400);
		setLayout(new BorderLayout());
		add(lbl,BorderLayout.SOUTH);
		addMouseListener(new MouseListener()
		{
			public void mouseExited(MouseEvent a){}
```

```
                  public void mouseClicked(MouseEvent a)
{lbl.setText("X="+a.getX()+" Y="+a.getY());}
                  public void mouseEntered(MouseEvent a) {}
                  public void mouseReleased(MouseEvent a) {}
                  public void mousePressed(MouseEvent a) {}


            });
      }
      public static void main(String[]args)
      {
            new MyMouse().setVisible(true);
      }
}
```

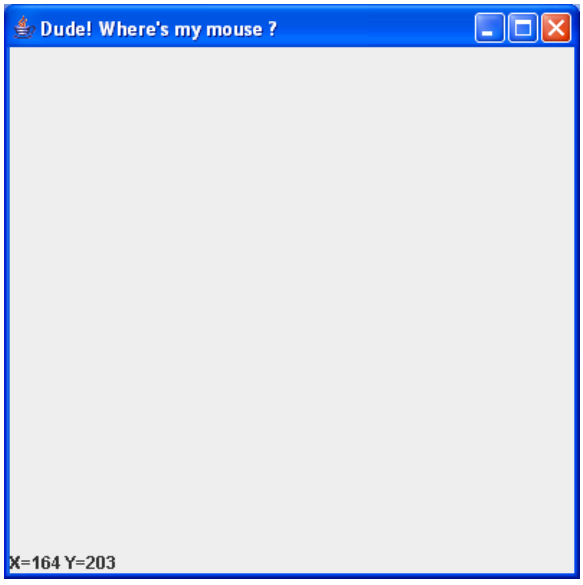

Рисунок 4.6

Создание меню.

Добавление меню в программе Java проста. Java определяет три компонента для обработки:

- JMenuBar: который представляет собой компонент, который содержит меню.
- JMenu: который представляет меню элементов для выбора.
- JMenuItem: представляет собой элемент, который можно кликнуть из меню.

Рисунок 4.7

Подобно компоненту Button (на самом деле MenuItems являются подклассами класса AbstractButton). Мы можем добавить ActionListener к ним так же, как мы делали с кнопками

ЗАДАНИЯ.

Упражнение 1.

Напишите интерактивную программу с использованием GUI имитирует таблицу результатов матчей между командами Милан и Мадрид. Создайте JFrame приложение у которого есть следующие компоненты GUI:

- одна кнопка JButton labeled “AC Milan”
- другая JButton подписана “Real Madrid”
- надпись JLabel содержит текст “Result: 0 X 0”
- надпись JLabel содержит текст “Last Scorer: N/A”
- надпись Label содержит текст “Winner: DRAW”;

Всякий раз, когда пользователь нажимает на кнопку AC Milan, результат будет увеличиваться для Милана, сначала 1 X 0, затем 2 X 0 и так далее. Last Scorer означает последнюю забившую команду. В этом случае: AC Milan. Если пользователь нажимает кнопку для команды Мадрид, то счет приписывается ей. Победителем становится команда, которая имеет больше кликов кнопку на соответствующую, чем другая.