

ЛАБОРАТОРНАЯ РАБОТА №5

СОЗДАНИЕ ПРОГРАМ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ ПОЛЬЗОВАТЕЛЯ НА JAVA

ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ:

Цель данной лабораторной работы - научиться создавать графический интерфейс пользователя, освоить на практике работу с различными объектами для создания ГИП, менеджерами размещения компонентов.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ:

Для создания графического интерфейса пользователя можно использовать стандартную Java библиотеку Swing или AWT. В этих библиотеках имеются различные классы, позволяющие создавать окна, кнопки, текстовые поля, меню и другие объекты.

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 параметре конструктора задано число 10, это количество символов, которые могут быть видны в текстовом поле. Текст введенный 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.

```
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.PIN
        K); jta.setFont(fnt);
        setVisible(true);
    }

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

```



Важное 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:

```

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().tri
m());
                double x2 =
Double.parseDouble(jta2.getText().tri
m());

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

            }
            catch(Exception e)
            {
                JOptionPane.showMessageDialog(n
ull, "Error in Numbers !","alert" ,
JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}

```

```

        });

        setVisible(true);
    }

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

```

JTextArea

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

```

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!

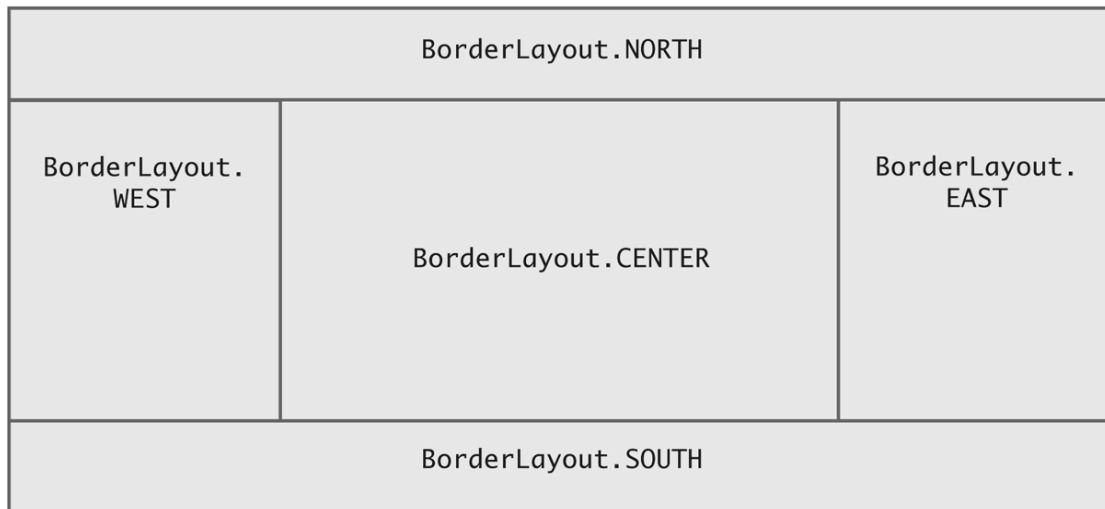


Layout Менеджеры:

BorderLayout:

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

Display 17.8 BorderLayout Regions



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

GridLayout менеджер

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

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 добавляться в порядке их появления.

Следующий пример иллюстрирует смесь компоновки различных компонентов

```
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);
    }
}
```

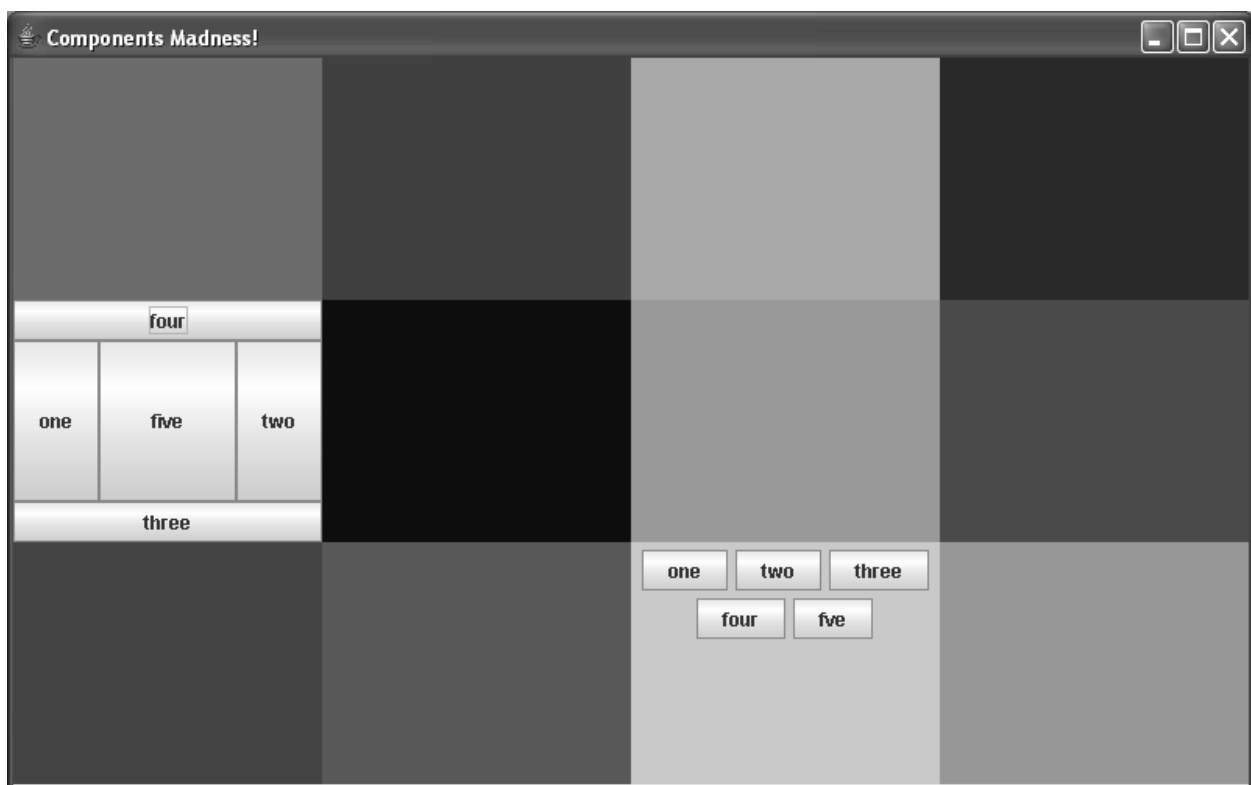
```

        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);
    }
}

```

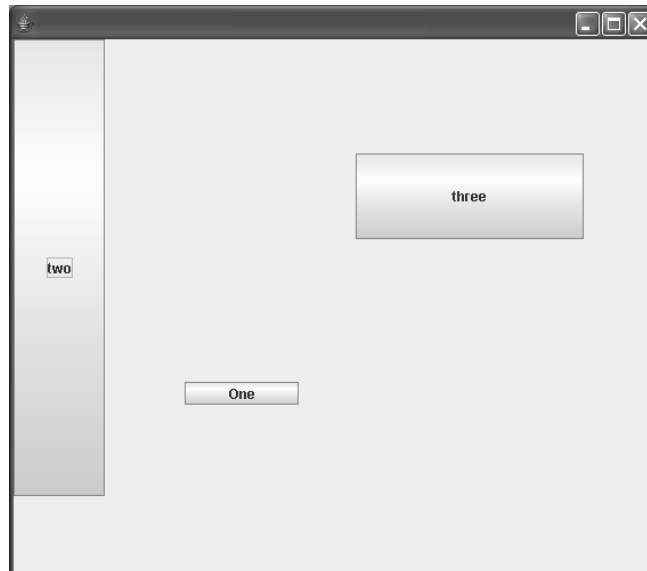
Вот такой будет иметь вид, представленный выше код



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.

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:

```
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(300,10
0);
        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);
    }
}

```



Меню

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

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



кликнуть из меню.

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

ВАРИАНТЫ ЗАДАНИЙ

1. Создать окно, нарисовать в нем 20 случайных фигур, случайного цвета. Классы фигур должны наследоваться от абстрактного класса Shape, в котором описаны свойства фигуры: цвет, позиция.
2. Создать окно, отобразить в нем картинку, путь к которой указан в аргументах командной строки.
3. Создать окно, реализовать анимацию, с помощью картинки, состоящей из нескольких кадров.