Тема 12 - Swing - Визуални приложения

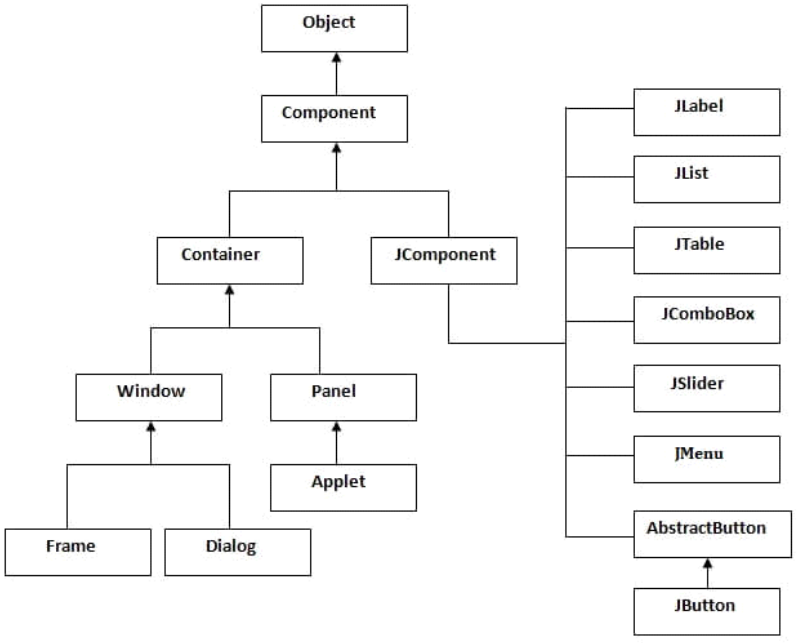
24 юни 2018 г. 22:40

Повечето програма, които използваме, не работят само в конзолата ,както до сега работиха нашите. Те имат графичен потребителски интерфейс (GUI - Graphical User Interface) и така имаме много по-голям избор за вход от потребителя. В Java към момента има 3 библиотеки, които ни позволяват да разработваме програми с графичен интерфейс чрез използването на компоненти(Като бутон, поле за въвеждане на текст и други):

- AWT - първичната библиотека на Java за създаване на GUI тя вече не се използва в чистият си вид понеже е остаряла с времето. Тя има недостатъци като:
  - Компонентите и зависят от платформата на изпълнение
  - Съдържа не достатъчен набор от компоненти
  - Компонентите и са сравнително тежки за визуализиране
- Swing - библиотека разработена на основата на AWT, която е има следните преимущества спрямо AWT:
  - Компонентите и не зависят от платформата - изглеждат по различен начин на Windows, Linux, Mac
  - Леки за визуализиране компоненти
  - Изградена по модела MVC - Model View Controller - който разделя логиката на програмата от визуалните елементи и ги свързва помежду с контролер.
  - Разполага с много повече компоненти от AWT, като таблици, листи, скрол панели и други.
- JavaFx - най-новата библиотека за разработване на визуални приложения, която се налага на пазара
- като стандарт(на нея ще обърнем внимание в следващата тема).

Понеже AWT е остаряла библиотека, ние ще започнем от Swing, като тя използва някой от класовете на AWT специфично за събития, за които ние ще трябва да следим(натискане на бутон и др.).

Йерархията от класовете в библиотеката на Swing изглежда по следният начин:



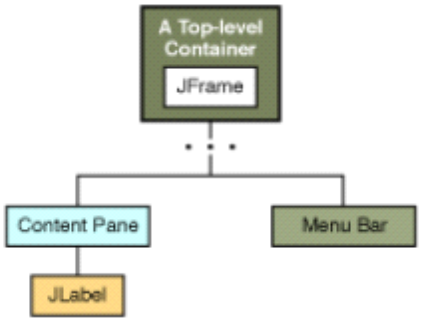
Swing ни предоставя три основни контейнера от най-високо ниво за нашите приложения:

- JFrame - главният контейнер за създаване на Swing приложения - ние ще се фокусираме само на него
- JDialog - диалогов прозорец - използва се за малки прозорци за визуализиране или въвеждане на някаква информация
- JApplet - малко приложение, което се пуска в уеб страница автоматично - вече е остаряла технология и не се използва

Тези класове, които наследяват JComponent представляват всичките компоненти, които може да използвате при построяването на вашите приложния.
Някои неща за компонентите и контейнерите:

- За да се появят на екрана графичните компоненти трябва да са част от графичната йерархия - което е дървовидна структура от графични елементи

- Всеки компонент може да се съдържа само веднъж в йерархията. Ако се добави втори път, той се премахва от първоначалното му място
- Всеки контейнер разполага с панел за съдържание (content pane), който съдържа графичните елементи.
- Може да се добави лента с меню в контейнерите от най-високо ниво

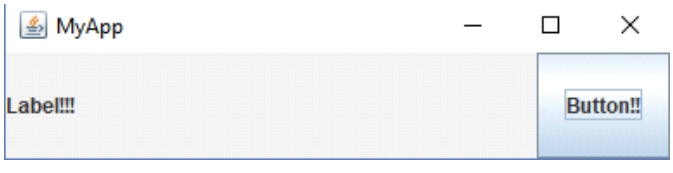


Ето така изглежда една съвсем проста йерархия на графично приложение.

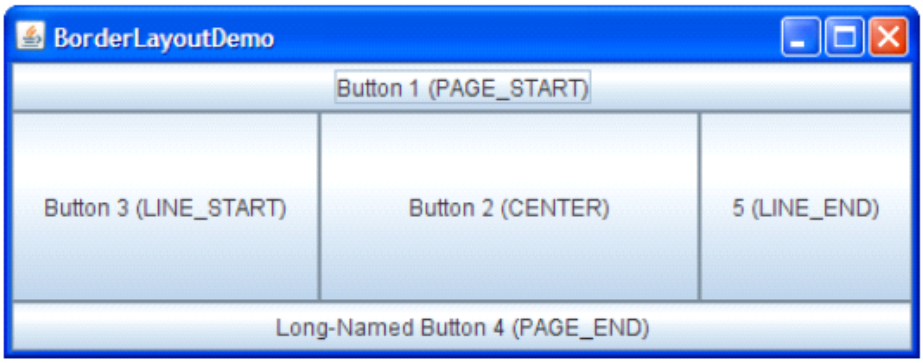
За създаването на графични приложения най-често се наследява JFrame класа(може и да се използва самият клас без да се наследява).

Първо приложение на Swing:

```
//Файл: MyApp.java
public class MyApp extends JFrame  { // наследяваме класа Jframe
    //Обявяване на компоненти
    JLabel label = new JLabel("Label!!!"); // поле, което е тип JLabel
    JButton button = new JButton("Button!!"); // поле, което е тип JButton
    public MyApp(String title) { // Конструктор
        super(title); // викаме конструктура на JFrame

        add(label,BorderLayout.LINE_START); // поставяме label-а в content pane на нашият клас и му
        задаваме да се напира в лявата част на BorderLayout
        add(button,BorderLayout.LINE_END); // поставяме buttona-а в content pane на нашият клас и му
        задаваме да се напира в дясната част на BorderLayout
    }

    public static void main(String [] args) {

        JFrame frame = new MyApp("MyApp"); // създаване на обект от нашият клас като му подаваме името
        на прозореца - тук се извиква конструктора и добавя компонентите ни
        frame.setDefaultCloseOperation(EXIT_ON_CLOSE); // задаване на операция при натискане на хикса
        горе в дясно
        frame.setSize(400,100); // задаване на размер на прозореца
        frame.setVisible(true); // показване на прозореца
    }


}
```



Забелязваме, че използваме методи като add(), които са обявени в JFrame и няма нужда да ги обявяваме ние.
Като добавяме, показваме къде точно да се намират нашите компоненти в нещо наречено BorderLayout.
Разполагането на елементи в BorderLayout изглежда по следният начин.



BorderLayout е част от класове наречени Layout Managers, те контролират как ще разполагате информацията на екрана си. Тези класове са намират в AWT библиотеката.

- BorderLayout* - по подразбиране за JFrame
- BoxLayout*
- CardLayout
- FlowLayout*
- GridBagLayout
- GridLayout*
- GroupLayout
- SpringLayout

Всичките се използват в определени ситуации, но тези които съм отбелязал с * се използват по-често.
Може да разгледате как изглежда всеки един от тях от тук:
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

Второ приложение:

За второто приложение ще използваме FlowLayout, който просто слага елементите на един ред освен ако, не могат да се поберат в контейнера. Задаваме го като извикаме метода:

```
setLayout(LayoutManager manager)
```

Който също е обявен в JFrame класа.

```
//Файл: ButtonApp
import java.awt.FlowLayout;
import java.awt.event.ActionListener;

import javax.swing.*;
public class ButtonApp extends JFrame { // наследява JFrame
    private JButton helloButton = new JButton("Hello"),
            howdyButton = new JButton("Howdy"); // създаване на два бутона с текст Hello и Howdy
    private JTextField textField = new JTextField(15); // създаване на текстово поле с 15 колони размер
    private ActionListener buttonListener = new ButtonListener(textField); // това го игнорирайте за
    сега

    public ButtonApp(String name) {
        super(name);
        helloButton.addActionListener(buttonListener); // игнорирайте
        howdyButton.addActionListener(buttonListener); // игнорирайте
        setLayout(new FlowLayout()); // това задава основният Layout на приложението ни
        //добавяне на бутоните и текстовото поле в панела с съдържание на приложението (content pane)
        add(helloButton);
        add(howdyButton);
        add(textField);
    }

    public static void main(String[] args) { // използваме main метода за да стартираме приложението
    JFrame frame = new ButtonApp("ButtonApp"); // създаваме  обект от нашият клас и задаване на
    заглавие на прозореца
    frame.setDefaultCloseOperation(EXIT_ON_CLOSE); // добавяме операция при натискане на хикса(горе в
    дясно) на прозореца
    frame.setSize(400,100); // обявяваме размера на прозореца
    frame.setVisible(true); // и го показваме на екрана
    }

}
```

В тази програма използваме системата за управление на събития(като натискане на бутон), която е наследена от AWT.
За да можем да контролираме тези събития трябва да имплементираме интерфейс в случая ActionListener (може да е ChangeListener, MouseListener и още около 72 други). И да зададен нашият клас като Listener на бутона ни.
Този ред създава обекта, който ще използваме.

```
private ActionListener buttonListener = new ButtonListener(textField);
```

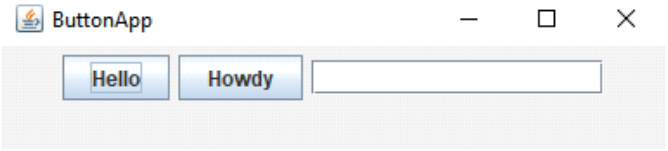
Тези редове го добавят на бутоните ни.(В случая двата бутона правят еднакви действия за това може да си позволим да е един и същи обект/клас)

```
helloButton.addActionListener(buttonListener);
howdyButton.addActionListener(buttonListener);
```

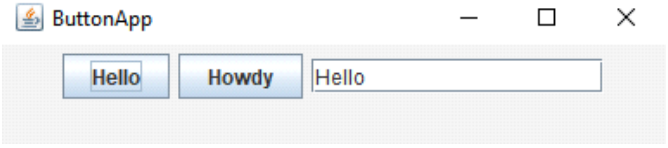
А ето и самият клас, който сме написали ние за да можем да го използваме, като слушател за събития. Когато имплементираме интерфейса ActionListener, той ни казва, че трябва да имплементираме метода:

```
public void actionPerformed(ActionEvent e);
```

```
//Файл:ButtonListener
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

public class ButtonListener implements ActionListener {

    private JTextField controlledField; // поле

    public ButtonListener(JTextField controlledField) { // конструктор
    this.controlledField = controlledField;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String name =  ((JButton)e.getSource()).getText();//Взимаме текста от бутона, който е натиснат.
           controlledField.setText(name);//Задаваме текста на управляваното поле.
    }

}
```



След натискане на Hello бутона:



Някои по интересни компоненти, които можете да използвате:

- JButton
- JCheckList
- JComboBox
- JSlider
- JLabel
- JTextArea

Може да разгледате повече компоненти тук:
http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

Когато искаме малко по-сложен Layout на нашето приложение използваме комбинация от JPanel-ли и поставяме на тях различни Layout-ти.

Трето приложение:

Програма за поръчка на пица

Файл: PizzaApp.java, Файл: OrderActionListener.java

```
public class PizzaApp extends JFrame  {
    private JTextArea orderSummary = new JTextArea(10,20);
    private JSlider pizzaSize; // Ще създадем обект при натстройването
    private JComboBox<String> pizzaType = new JComboBox<>();
    private JPanel leftPanel = new JPanel(),
            rightPanel = new JPanel(),
            toppingPanel = new JPanel();
    private JCheckBox addCheese = new JCheckBox("Add cheese?");
    private JCheckBox addOlives = new JCheckBox("Add olives?");
    private JCheckBox addPeppers = new JCheckBox("Add peppers?");
    private JCheckBox addPepperoni = new JCheckBox("Add pepperoni?");
    private JTextField comment = new JTextField();
    private JButton submitOrder = new JButton("Submit Order");
    private OrderActionListener orderListener = new OrderActionListener(submitOrder);
    public PizzaApp(String title) {
    private void setLayouts(){
    private void customizeElements(){
    private void customizePizzaSize(){
    private void customizePizzaType(){
    private void addElementsToInterface(){
    public static void main(String [] args) {

        JFrame frame = new PizzaApp("Pizza Order Form");
        frame.setDefaultCloseOperation(EXIT_ON_CLOSE);
        frame.setSize(700,300);
        frame.setVisible(true);
    }

}
```

Понеже кода на програмата е голям не слагам целият код, а само как изглежда класа.

Интересните части:

На различните панели слагам различни Layout-и в случая слагаме BoxLayout-и но с различни настройки :

```
private void setLayouts(){
    setLayout(new FlowLayout());
    toppingPanel.setLayout(new BoxLayout(toppingPanel,BoxLayout.X_AXIS));
    leftPanel.setLayout(new BoxLayout(leftPanel,BoxLayout.Y_AXIS));
    rightPanel.setLayout(new BoxLayout(rightPanel,BoxLayout.Y_AXIS));
}
```

Добавянето на елементите в панелите и добавянето на панелите в самият JFrame:

```
private void addElementsToInterface(){

    leftPanel.add(new JLabel("Pizza type:"));
    leftPanel.add(pizzaType);
    leftPanel.add(new JLabel("Pizza Size:"));
    leftPanel.add(pizzaSize);
    leftPanel.add(new JLabel("Extra toppings:"));
    toppingPanel.add(addCheese);
    toppingPanel.add(addOlives);
    toppingPanel.add(addPeppers);
    toppingPanel.add(addPepperoni);
    leftPanel.add(toppingPanel);
    leftPanel.add(new JLabel("Comment:"));
    leftPanel.add(comment);
    leftPanel.add(submitOrder);

    rightPanel.add(new JLabel("Order Summary:"));
    rightPanel.add(orderSummary);
    add(leftPanel);
    add(rightPanel);

}
```

Добавяне на ActionListener без създаване на клас(inline)

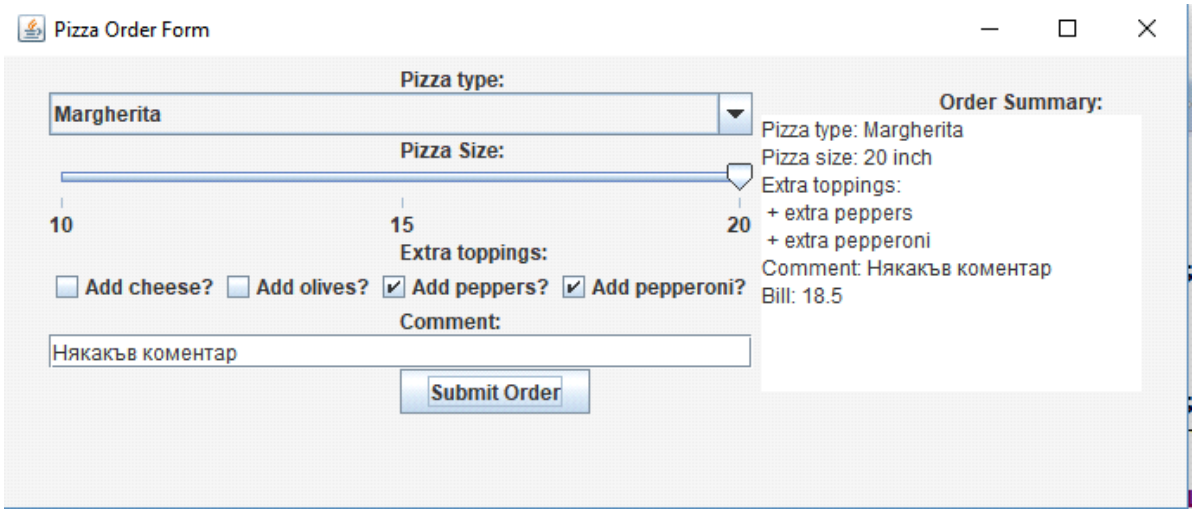
```
submitOrder.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {

            .... код

    }

});
```

Едит клас имплементира два Listener интерфейса(и активиране на бутон от кода):

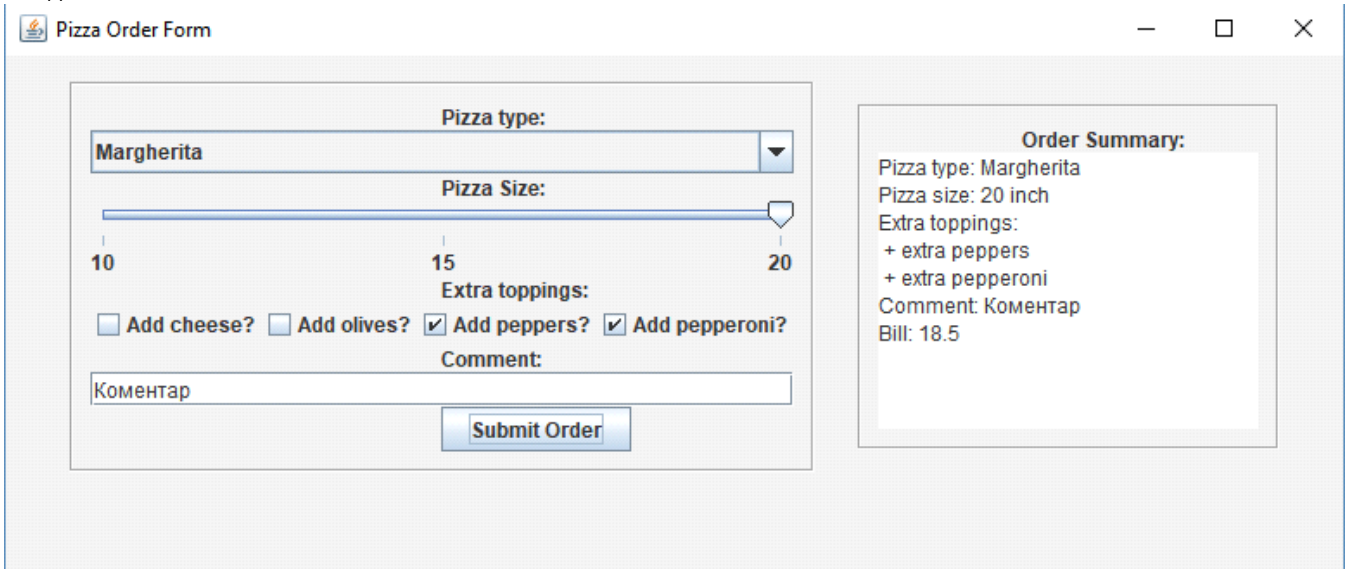
```java
public class OrderActionListener implements ActionListener, ChangeListener {
    JButton controlButton;
    public OrderActionListener(JButton controlButton){
        this.controlButton = controlButton;
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        controlButton.doClick();

    }
    @Override
    public void stateChanged(ChangeEvent arg0) {
        controlButton.doClick();

    }

}
```

активиране на бутон от код

Как изглежда приложението без Border-и



Със добавени Border-и



Начин за добавяне на Border-и:

```
private void addBorders(){
```

```
    leftPanel.setBorder(BorderFactory.createCompoundBorder(new EmptyBorder(10, 10, 10,
    10),BorderFactory.createCompoundBorder( new EtchedBorder(),new EmptyBorder(10, 10, 10, 10))));
    rightPanel.setBorder(BorderFactory.createCompoundBorder(new EmptyBorder(10, 10, 10,
    10),BorderFactory.createCompoundBorder( new EtchedBorder(),new EmptyBorder(10, 10, 10, 10))));
}
```

Използваме класа BorderFactory за да създадем Border с неговият статичен(може да се използва без да се създава обект) метод за създаване на композиция от Border-и.
EtchedBorder - вдлъбнат стил Border
EmptyBorder - празен Border като посочваме разстоянието в горе, ляво, долу , дясно
Може да разгледате още за тях:
https://docs.oracle.com/javase/tutorial/uiswing/components/border.html

Swing библиотеката за визуални приложения е голяма и трудно можем да я покрием цялата, но главните неща които трябва да се научат са:

- Containers
- Components
- Layout Managers
- Event Listeners

Допълнителни линкове към темата:
http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
https://docs.oracle.com/javase/tutorial/uiswing/TOC.html

Автор: Димитър Томов - xdtomov@gmail.com