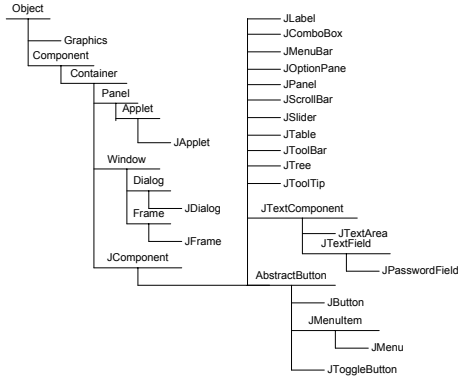
Графичен потребителски интерфейс (GUI)

Приложения с GUI

1. **Създават прозорци върху екрана**
2. **Съдържат визуални елементи за:**
   - **изобразяване на информация;**
   - **взаимодействие на потребителя с програмата чрез мишка или клавиатура;**
3. **Класове от пакетите** java.awt **или** javax.swing

Йерархия на класовете



Компонента

1. **Обект, който представя графична информация или позволява на потребителя да взаимодейства с програмата по някакъв начин**
2. **Примери: бутони, текстови полета, етикети, плъзгачи, менюта**
3. **Генерира събития**
4. **Програмите отговарят на събитията чрез слушателски класове**
5. **Всяко събитие има съответстващ адаптерен клас**
6. **Подходящият слушател се добавя към компонентата, генерираща събитието**

Клас Component **(**java.awt**) – супер клас за всички графични компоненти.**

public void paint(Graphics page)

Отпечатва компонентата с графично съдържание page**.**

public void repaint()

Пречертава компонентата.

public void update(Graphics page)

Обновява компонентата, използвайки графичното съдържание page**.**

public Color getBackground()
public Color getForeground()

public void setBackground(Color color)
public void setForeground(Color color)

Връща/установява цвета на фона/цвета за изчертаване на компонентата.

public Dimension getSize()
public void setSize(Dimension dim)
public void setSize(int width, int height)

Връща/установява размера на компонентата.

public void setLocation(int x,int y)

Премества компонентата в ново местоположение с горен ляв ъгъл, определен от x **и** y **параметрите в координатното пространство на компонентата-родител.**

public synchronized void add(PopupMenu popmenu)
public synchronized void remove(PopupMenu popmenu)

Добавя/премахва издигащото се меню popmenu **към компонентата.**

public Graphics getGraphics()

Връща графичното съдържание на компонентата.

public void setSize(int width, int height)

Установява размера на компонентата с ширина width **и височина** height**.**

public void setVisible(boolean b)

Изобразява/скрива компонентата според стойността на b **(**true**/**false**).**

Събития с компоненти

ComponentEvent

FocusEvent

KeyEvent

MouseEvent

MouseMotionEvent

MouseWheelEvent

Добавяне на слушатели на събитията

public void addXXListener(XXListener l)

XX – **име на събитие**

Клас JComponent **(**javax.swing**)**

Базов клас за всички Swing компоненти (без контейнерите на по-високо ниво).

Контейнер

1. **Компонента, която може да съдържа други компоненти**
2. **Контейнерите и свързаните с тях менажери на разположението определят организирането и изобразяването на компонентите (интерфейс** LayoutManager**).**
3. **Видове контейнери:**

– **аплет;**

– **фрейм – прозорец, който може да се разположи на произволно място върху екрана;**

– **панел – организира групи от компоненти в по-голям контейнер; не може да бъде изобразяван и трябва да се добави към съществуващ контейнер.**

Фрейми и аплети – контейнери от горно ниво.

Клас Container **(**java.awt**) – супер клас за контейнери**

public Component add(Component item)
public Component add(Component item, int index)
public Component add(String str, Component item)

Добавя компонента item **към контейнера в определено място** index **(към края по подразбиране) с име** str**.**

public Component getComponent(int index)
public Component getComponent(int x, int y)
public Component getComponent(Point p)

Връща компонента, разположена в определената точка point **или** index**.**

public LayoutManager getLayout()

public void setLayout(LayoutManager layout)

Връща/установява менажера на разположението за контейнера.

Клас JFrame **(**javax.swing**) – представя прозорец**

public JFrame(String title)

Конструкторът създава прозорец със заглавие title**.**

public void setDefaultCloseOperation(int operation)

Установява операцията при затваряне на прозореца.

operation =
- JFrame.DO_NOTHING_ON_CLOSE – **нищо**
- JFrame.HIDE_ON_CLOSE – **скрий прозореца**
- JFrame.DISPOSE_ON_CLOSE – **освободи прозореца**
- JFrame.EXIT_ON_CLOSE – **завърши приложението**

public Container getContentPane()

Връща обект от класа Container **със съдържанието на фрейма.**

public void setContentPane(Container contentPane)

Установява съдържанието на фрейма.

Събития

1. **Всички събития за компоненти**
2. **Събития с прозорци**

– **клас** WindowEvent

– **интерфейс** WindowListener **или адаптерен клас** WindowAdapter

– **добавяне на слушател** l **за събитието** WindowsEvent **към компонентата**

public void addWindowListener (WindowListener l)

MEG2 Mariana Goranova; 24.8.2006 г.

```
public interface WindowListener {
  // Извиква се, когато прозорецът се установи като активен прозорец, т.е.
  // ще получава събития от клавиатурата.
  void windowActivated (WindowEvent e);
  // Извиква се, когато прозорецът се затвори при разрушаването му.
  void windowClosed (WindowEvent e);
  // Извиква се, когато прозорецът се затвори от системното си меню.
  void windowClosing (WindowEvent e);
  // Извиква се, когато прозорецът повече не е активен, т.е. престава да
  // получава събития от клавиатурата.
  void windowDeactivated (WindowEvent e);
  // Извиква се при промяна на прозореца от икона в нормално състояние.
  void windowDeiconified (WindowEvent e);
  // Извиква се при промяна на прозореца от нормално състояние в икона.
  void windowIconified (WindowEvent e);
  // Извиква се, когато прозорецът е станал видим за първи път.
  void windowOpened (WindowEvent e);
}
```

Клас SwingUtilities **(**javax.swing**)**

Съдържа помощни методи за Swing.

public static void invokeLater(Runnable doRun)

Предизвиква асинхронно изпълнение на метода doRun.run() **в нишката (клас** Thread**, реализиращ интерфейса** Runnable**), изпълняваща събитието.**

Клас JOptionPane **(**javax.swing**)**

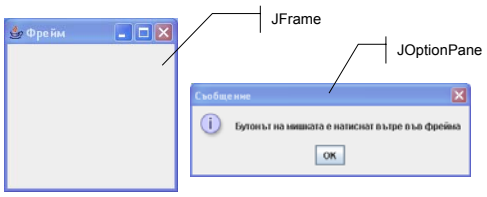
Представя стандартен диалогов прозорец за:

- **потвърждаване** showConfirmDialog
- **входен диалог** showInputDialog
- **за съобщение** showMessageDialog
- **за диалог с опции** showOptionDialog

public static void showMessageDialog(Component parentComponent,
Object message, String title, int messageType, Icon icon)

parentComponent – **компонента-родител**
message – **съобщение, което се изобразява**
title – **заглавие на диалоговия прозорец**
messageType – **тип на съобщението:** ERROR_MESSAGE
INFORMATION_MESSAGE
WARNING_MESSAGE
QUESTION_MESSAGE
PLAIN_MESSAGE
icon – **изобразявана икона**

Пример: Приложение, което обработва събитието натискане бутон на мишката – изобразява се диалогов прозорец със съобщението Бутонът на мишката е натиснат вътре във фрейма**.**




```
import java.awt.event.*;
import javax.swing.*;
public class FrameDemo extends JFrame  {
  public FrameDemo () {
    super("Фрейм");       // Извиква конструктора на суперкласа JFrame
    // Приложението завършва при затваряне на прозореца
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    // Добавя слушател за събитието „натискане бутона на мишката“
    this.addMouseListener(new MouseAdapter() {
      // Обработва събитието
      public void mousePressed(MouseEvent e) {
        JOptionPane.showMessageDialog(null,
        "Бутонът на мишката е натиснат вътре във фрейма","Съобщение",
        JOptionPane.INFORMATION_MESSAGE);
      }
    });
    // Изобразява прозореца
    this.setSize(200, 200);
    this.setVisible(true);
  }
```

```
  public static void main(String[] args) {
    // Изпълнява асинхронно приложението в нишка, изпълняваща
    // събитието
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        new FrameDemo();
      }
    });
  }
}
```

Панел

Клас JPanel **(**javax.swing**)**

Представя контейнер.

public JPanel()

Създава контейнер.

public void setBorder(Border border)

Установя рамка за панела.

public Component add(Component comp)

Добавя дадената компонента comp **към контейнера.**

Бутон

Клас JButton **(**javax.swing**)**

Създава бутон. При натискането му се генерира събитието ActionEvent**, изпраща се съобщението** actionPerformed **на интерфейса** ActionListener **за всички слушатели, които са се регистрирали чрез метода** addActionListener**.**

public JButton()

Конструира JButton **без етикет.**

public JButton(String label)

Конструира JButton **с определен етикет** label**.**

public void addActionListener(ActionListener l)

Добавя слушател l**, за да приеме събитията** ActionEvent **при натискане на бутона.**

Методът getSource **на събитието** ActionEvent

public Object getSource()

връща обекта-източник на събитието.

```
public interface ActionListener {
   // Извиква се, когато се случи събитието ActionEvent, показва източника е
   // на събитието и потребителят описва действието, което трябва да се
   // предприеме.
   void actionPerformed (ActionEvent e);
}
```

Етикет

Клас JLabel **(**javax.swing**)**

Изобразява област за къс текст, изображение или и двете. Не реагира на входни събития.

public JLabel(String text)

Създава етикет с определен текст text**.**

Менажери на разположението

Менажер на разположението

- **обект, който управлява разполагането на компонентите в контейнера;**
- **определя размера и разположението на всяка компонента;**
- **установява се за контейнер чрез метода**

  public void setLayout(LayoutManager layout)

Интерфейс LayoutManager **(**java.awt**)**

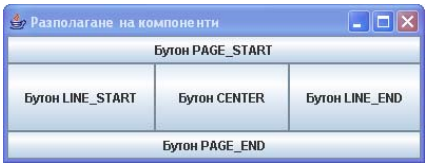
- **дефинира интерфейс за класове, които разполагат компонентите в контейнери.**

Видове менажери

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- SpringLayout

Клас BorderLayout (java.awt)

Разполага компонентите в пет области: горна, долна, лява, дясна и централна, които се определят чрез константите: PAGE_START, PAGE_END, LINE_START, LINE_END **и** CENTER **.**




```
import javax.swing.*;
import java.awt.*;
public class Layout extends JFrame {
  public Layout() {
    super("Разполагане на компоненти");
    // Приложението завършва при затваряне на прозореца
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    // Получава съдържанието на контейнера
    Container container = this.getContentPane();
    // Установява менажер на разположението
    container.setLayout(new BorderLayout());
    // Създава компоненти и ги поставя във фрейма
    JButton top = new JButton("Бутон PAGE_START");
    container.add(top, BorderLayout.PAGE_START);
    JButton bottom = new JButton("Бутон PAGE_END");
    container.add(bottom, BorderLayout.PAGE_END);
    JButton left = new JButton("Бутон LINE_START");
    container.add(left, BorderLayout.LINE_START);
    JButton center = new JButton("Бутон CENTER");
    container.add(center, BorderLayout.CENTER);
    JButton right = new JButton("Бутон LINE_END");
    container.add(right, BorderLayout.LINE_END);
```

```
    // Изобразява фрейма
    this.setSize(400, 150);
    this.setVisible(true);
  }
  public static void main(String[] args) {
    // Изпълнява асинхронно приложението в нишка, изпълняваща
    // събитието
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        new Layout();
      }
    });
  }
}
```

Клас FlowLayout **(**java.awt**)**

Разполага компонентите в поток от ляво надясно. Обикновено се използва за аранжиране на бутони в панел. Всяка линия се центрира. Това е подразбиращият се менажер за JPanel**.**

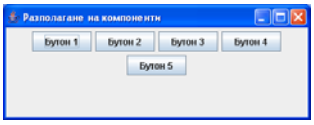
public FlowLayout()

public FlowLayout(int alignment)

public FlowLayout(int alignment, int horizontalGap, int verticalGap)

alignment – **подравняване: отляво, центрирано (по подразбиране), отдясно; определя се чрез константите:** LEADING, CENTER, TRAILING;

horizontalGap, verticalGap – **брой пиксели между компонентите (5 пиксела по подразбиране)**




```
// Установява менажер на разположението
container.setLayout(new FlowLayout());
// Създава компоненти и ги поставя във фрейма
JButton button1 = new JButton("Бутон 1");
container.add(button1);
JButton button2 = new JButton("Бутон 2");
container.add(button1);
JButton button3 = new JButton("Бутон 3");
container.add(button3);
JButton button4 = new JButton("Бутон 4");
container.add(button4);
JButton button5 = new JButton("Бутон 5");
container.add(button5);
```

Клас BoxLayout **(**javax.swing**)**

Разполага компонентите хоризонтално или вертикално.

public BoxLayout(Container target, int axis)

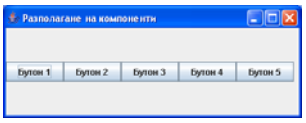
target – **контейнерът, който ще бъде аранжиран;**

axis – **начин на разполагане на компонентите, определя се чрез константите:**

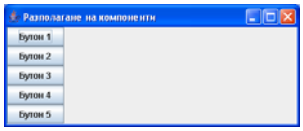
- X_AXIS – **хоризонтално разполагане отляво надясно;**
- Y_AXIS – **вертикално разполагане отгоре надолу;**
- LINE_AXIS – **както разположението на думи в ред, основано на свойството** ComponentOrientation **на контейнера;**
- PAGE_AXIS – **както разположението на редове в страница, основано на свойството** ComponentOrientation **на контейнера.**

```
// Установява менажер на разположението
container.setLayout(new BoxLayout(container, BoxLayout.LINE_AXIS));
```

BoxLayout.LINE_AXIS **или** BoxLayout.X_AXIS



BoxLayout.PAGE_AXIS **или** BoxLayout.Y_AXIS



Клас GridLayout **(**java.awt**)**

Разполага компонентите в мрежа от елементи с еднакви размери.

public GridLayout (int rows,int cols)

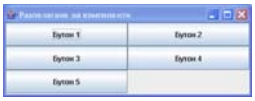
public GridLayout (int rows,int cols,int hgap,int vgap)

rows**,** cols**– брой реда/стълба; допуска** rows = 0 **или** cols = 0 **(не едновременно 0);** rows **е с приоритет пред** cols**;**

hgap**,** vgap **– хоризонтално/вертикално празно пространство.**

```
// Установява менажер на разположението
container.setLayout(new GridLayout(3,2));
```

new GridLayout(3,2)



new GridLayout(2,3) **или**
new GridLayout(2,2)

new GridLayout(1,0)

new GridLayout(0,1)