Тема 11. Интерфейс SWING

Съдържание

- Модел на Swing
  - Същност;
  - Схема на модела;
  - Паралелизъм в архитектурата на MVC.
- Имплементация
  - Основни пакети;
  - Йерархията на някои от класовете на JFC;
  - Елементи;
  - Събитиен модел;
- Моделиране на таблични интерфейсни класове с технология SWING
  - Добавяне на таблицата в прозорец;
  - Установяване и смяна на ширината на колоните;
  - Установяване на потребителска селекция;
- Изграждане на модел на таблицата
  - Редактори и интерпретатори на типовете;
  - Създаване на комбиниран списък за редакция на данни;

Модел на Swing

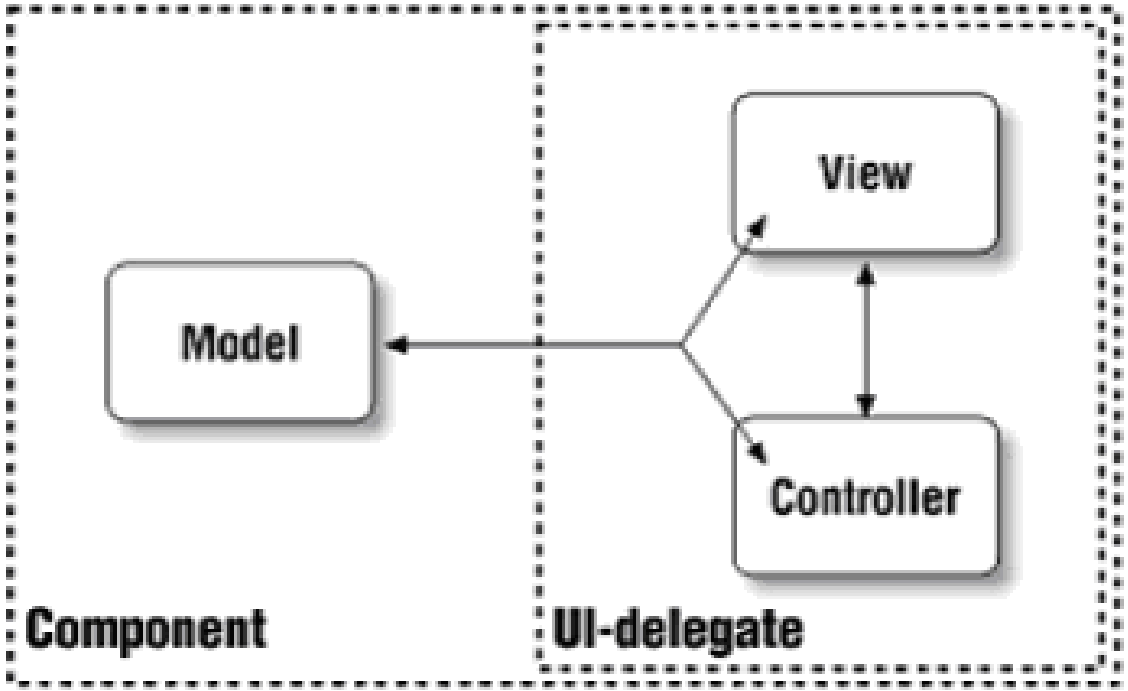
Същност на модела

В Swing се използва опростен вариант на модел-начин на изобразяване-контролер (MVC) наречен **делегиран модел**. Той включва:

- Модел;
- Изобразяване + обработване на потребителските събития-(UI delegate).

Особености на MVC в Swing. Схема на модела

Схема на модела:



Особености на MVC в Swing. Схема на модела

Разделяне на отговорностите:

- Моделът отговаря за съхраняване на информацията за състоянието на компонента;
- Общият делегиран потребителски интерфейс е отговорен за начина на изобразяване и за потребителските въздействия с помощта на пакета AWT;

Следствие:

- двупосочна връзка между модела и общия елемент;

Предимство: един и същ модел може да работи с много делегирани интерфейси – допуска се паралелизъм.

Особености на MVC в Swing. Схема на модела

Пример: Изобразяване на данни в таблица и графика:

- Общ модел и множество от начини на извеждане;
- Променяне на данните на едно място- изображенията се обновяват в съответствие с новото състояние на модела;
- Позволява променяне на изгледите на компонента, добавяне на нови такива и др. преработки, без да се засяга работата на модела.

Особености на MVC в Swing. Паралелизъм в архитектурата на MVC

Например-документ:

- Редактиращо изобразяване;
- Изобразяване на печат;
- Отделни начини за изобразяване на всяка от страниците на документа.

Всеки потребителски интерфейс се занимава с това, което трябва да знае за неговата специфична задача, но всяко от тях взаимодейства с общия модел. Накрая всяко изобразяване се свързва с потребителски интерфейс, реализиран посредством обекта контролер.

Особености на MVC в Swing. Паралелизъм в архитектурата на MVC

Действие:

- Моделът има N начина на изобразяване. Моделът е един, към който има множество начини на изобразяване и асоциирани към тях класове контролер;
- Контролерите получават потребителските действия, като на всяко събитие, което е необходимо на приложението се изпраща съобщение за промяна към модела;
- Промяната на модела може да се предизвика и от събития, получени от асинхронни промени в базата от данни. Независимо от причината, когато моделът се промени, веднага се уведомяват всички изобразявания.

Особености на MVC в Swing. Паралелизъм в архитектурата на MVC

В Java, контролера и изобразяването са комбинирани:

- Реализация чрез пакет за интерфейс- например интерфейс Swing;
- Приложението се разделя на **модел** и асоциирани **View/Controllers**, представени като интерфейсни компоненти на Swing и техните слушатели (**listeners**).

Особености на MVC в Swing. Паралелизъм в архитектурата на MVC

Независимо от фактическото изпълнение на View/Controllers, главната идея на MVC трябва да отдели модела от изобразяването му на потребителски интерфейс.

Предимства на подхода:

- Моделът може да се разработи напълно **независимо от начина**, **по който ще се изобразява**;
- При такава независимост на модела има голяма **гъвкавост при разработката на начините**, които могат да се разработят за показване на модела;
- **Новите представяния** могат да се създават заедно с целия потребителски интерфейс и независимо от модела.

Имплементация

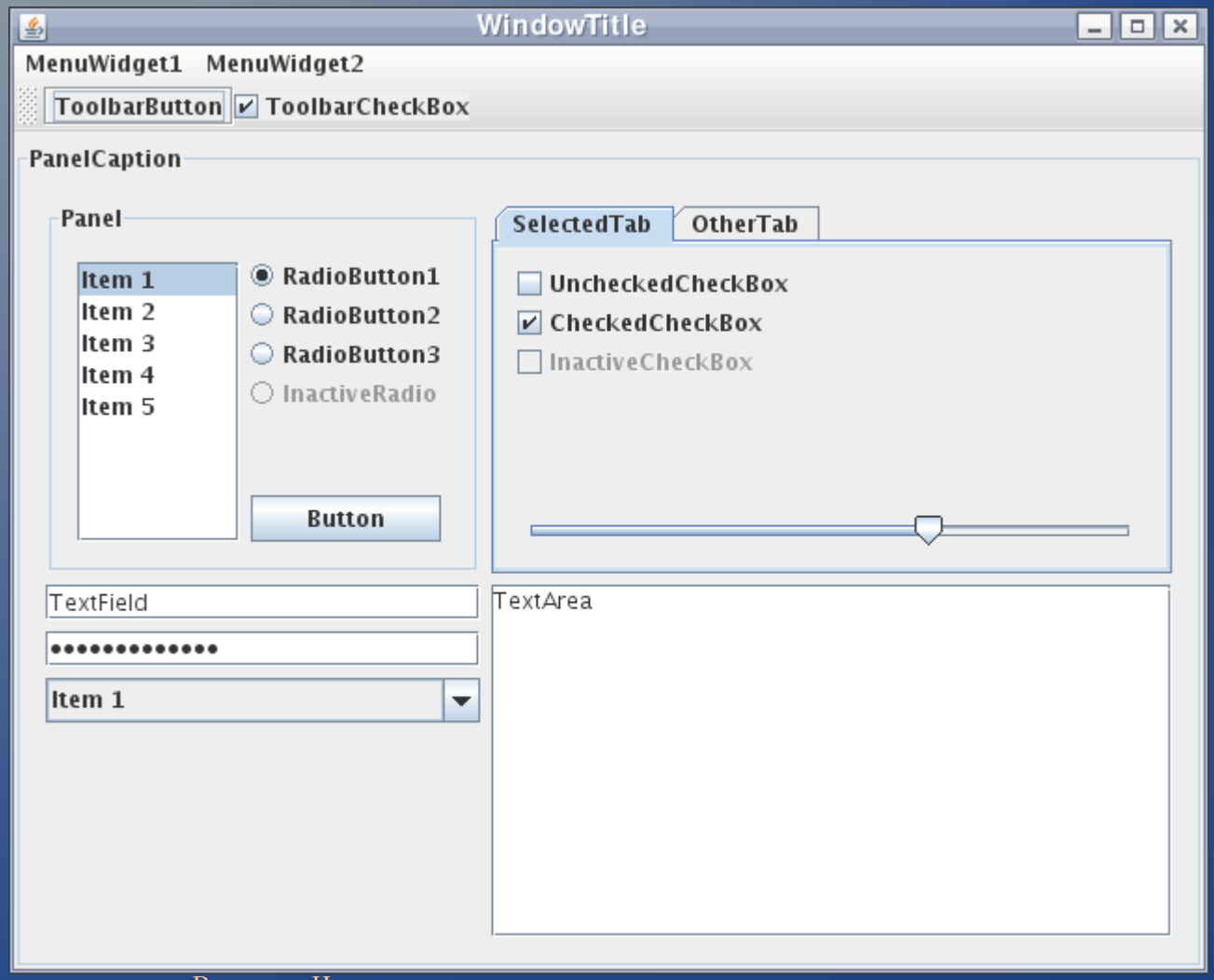
В Java View и Controller се представят от слушатели **listeners**.

Предимство: независимостта на разработване на модела от видовете изобразявания.

- Потребителските интерфейси (GUI - Graphic User Interface) съдържат стандартни графични компоненти:
  - бутони;
  - списъци;
  - менюта и др.

Имплементация

Обща визуализация на различни платформи:



Имплементация. Основни пакети

За изграждане на GUI Java 2 предлага пакети, обединени под общото название JFC (Java Foundation Classes). Основни пакети:

- **java.awt**. Осигурява необходимата инфраструктура на GUI и всички видове GUI-компоненти.
- **java.awt.event**. Реализира събитийния модел на Java. Осигурява обработката на събитията, генерирани в резултат от манипулациите на потребителя с GUI, като например натискаане на бутон, придвижване на мишката и др.
- **javax.swing**. Предлага нови, усъвършенствани и по-естетични GUI-компоненти в сравнение с **java.awt**, наричани Swing-компоненти.

Имплементация. Основни пакети

- Пакетите **java.awt** и **java.awt.event**, заедно с пакета **java.awt.datatransfer**, са известни под общото название AWT (Abstract Window Toolkit). В Java 2 се използва версията AWT 1.1. Пакетът **javax.swing** и още 15 други пакета са известни под общото название Swing.

Имплементация. Основни пакети

Имената на Swing-компонентите:

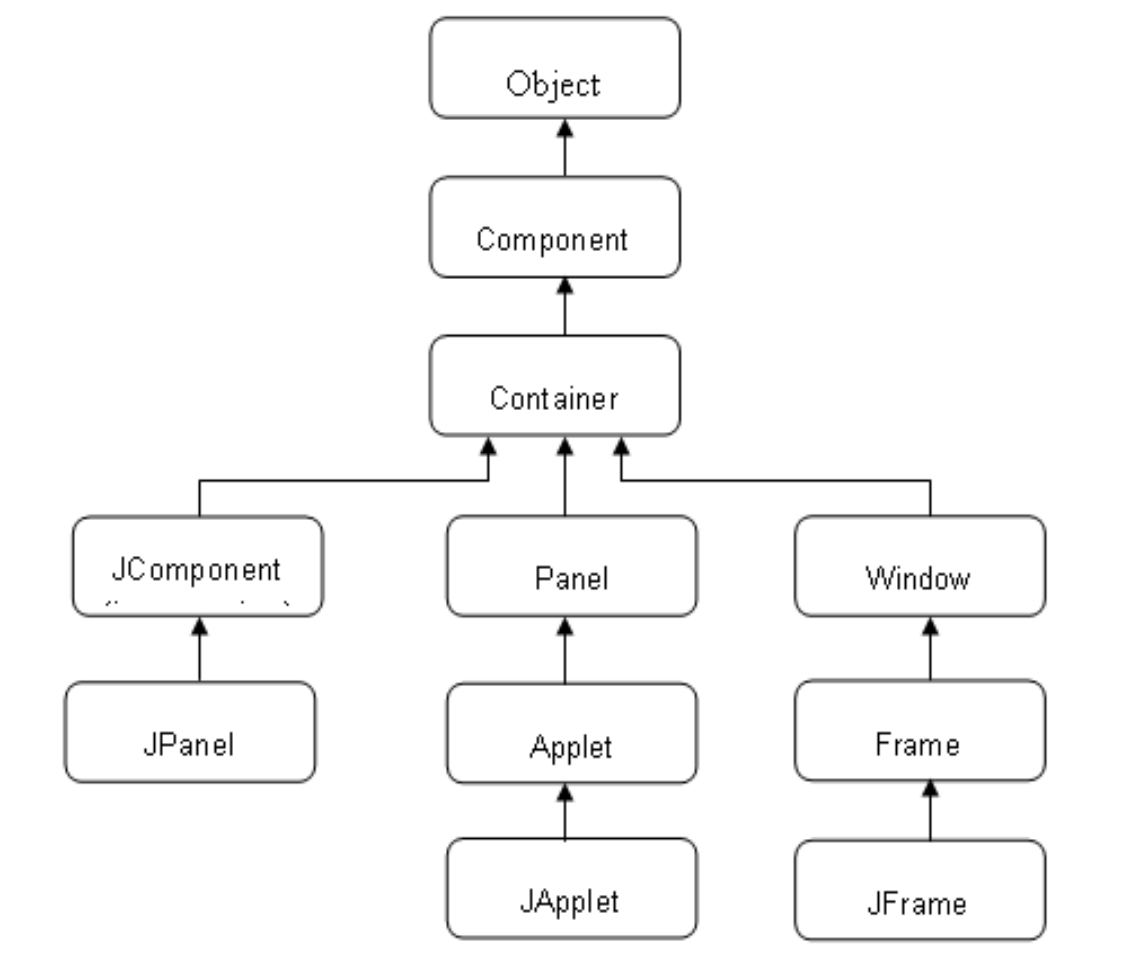
- Образувани са от тези на AWT с началната им буква “J”. Например: AWT **Button** - Swing **JButton**.
- Компонентите **JApplet** и **JFrame**, представляват контейнери от най-високо ниво. Swing-компонентите на аплетите се разполагат в тях. Те притежават четирислойна структура от повърхности, от които за програмиста най-важен е четвъртият слой, наричан повърхност на съдържанието - *content pane*.

Имплементация. Основни пакети

Пример: Един бутон се разполага в **JApplet** чрез следния код:

```
JApplet applet = new JApplet();
. . . . . .
Container contentPane =
  applet.getContentPane();
JButton button = new JButton("OK");
contentPane.add(button);
```

Имплементация.
Йерархията на някои от класовете на JFC



Имплементация. Елементи

Върху *content pane* могат да се разполагат и панели. Те се създават посредством класа **JPanel**. Панелите се характеризират с две свойства:

- Притежават повърхност, върху която може да се изобразява текст и графика;
- Те са контейнери, които могат да съдържат в себе си GUI-компоненти.

Имплементация. Елементи

Включване на панел в контейнер от тип **JApplet** се извършва с метода на Container - add:

```
Container contentPane = applet.getContentPane();
JPanel panel = new JPanel();
contentPane.add(panel);
```

Имплементация. Събитиен модел

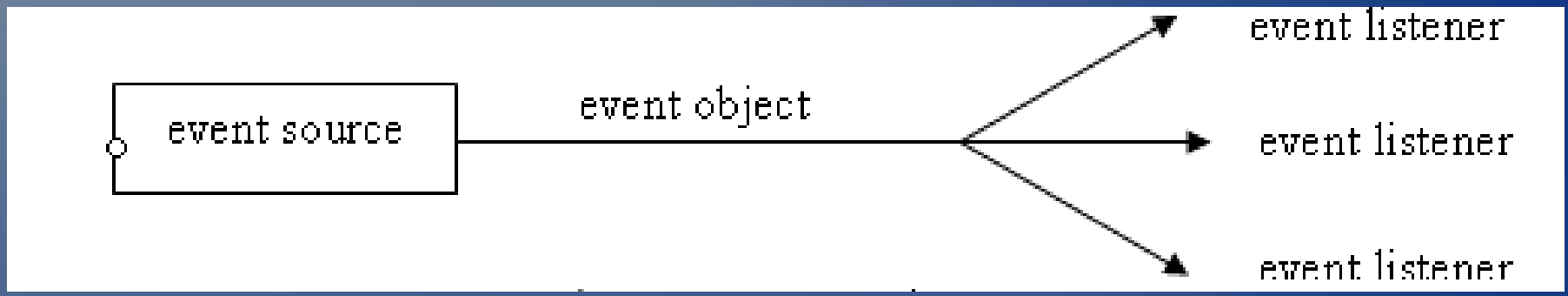
Събитийният модел на Java е валиден и за аплетите. Схема на взаимодействие:

Потребителят въздейства с графичния интерфейс:

- Swing-компонентите генерират събития, които се обработват;
- Компонетите са *източници на събития* (event sources);
- Обработката на събитията се извършва от съответстващите им *обработчици на събития* (event handlers)-наричат се и "*слушатели*" (event listeners)

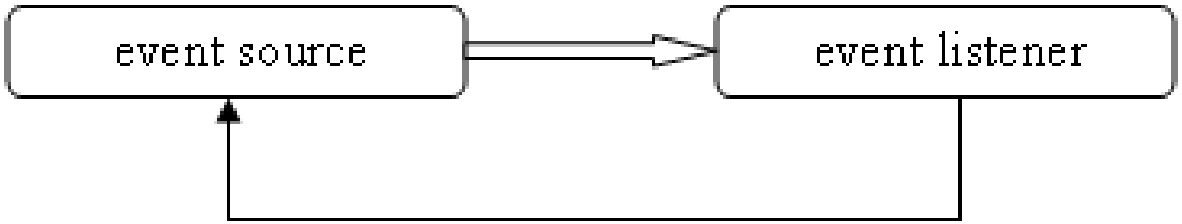
Имплементация. Събитиен модел

Всяко събитие се представя от обект (event object), който съдържа в себе си информация относно събитието:



Имплементация. Събитиен модел

За да може даден източник да знае на кого от слушателите да изпрати съобщение за възникнало събитие, необходимо е слушателят за този вид събития да бъде регистриран в източника:



Имплементация. Събитиен модел

- Един и същ източник може да генерира различни по вид събития:
  - Източникът, в зависимост от вида на събитието, трябва да може да изпраща съобщения на различни слушатели;
  - За да може да се осъществява това, в източника се регистрират едновременно повече слушатели.

Имплементация. Събитиен модел (изисквания към слушателя)

За всеки обработчик (слушател) на събития трябва да са изпълнени три условия:

1. Слушателят за определен вид събитие трябва да бъде обект на клас, който имплементира интерфейс, съответстващ на този вид събитие. Такъв тип интерфейси, предназначени за събития, се наричат с общото название *интерфейси за подслушване* (listener interface). В декларацията на класа на слушателя следва да се укаже, че класът имплементира нужния интерфейс за подслушване, например **ActionListener** :

public class MyClass **implements ActionListener** { . . . }

или че наследява клас, който имплементира същия интерфейс.

Имплементация. Събитиен модел (изисквания към слушателя)

2. Обработчикът (слушателят) трябва да е регистриран в източника на събитието.

Осъществява се по следния примерен начин:

button.addActionListener(**референция на обекта-слушател на MyClass**);

Имплементация. Събитиен модел (изисквания към слушателя)

3. В класа на обекта-слушател (в случая в MyClass) трябва да са предефинирани методите на слушателския интерфейс:

Пример:

```
public void actionPerformed(ActionEvent e) {
    // Код, обработващ събитието
}
```

Имплементация. Събитиен модел

Основни стъпки при имплементацията

1. Създаване на компонент
   - например JButton;
2. Добавяне на подходящо място в GUI
   - например JPanel;
3. Регистрация на слушател, който получава управлението, когато компонент генерира събитие:
   - например ActionListener за натискане);
4. Определяне на метод (callback), който се извиква, когато слушател е уведомен :
   - например actionPerformed за ActionListener.

Имплементация. Събитиен модел

Пример, в който обект на класа MyClass, представен чрез **this**, се регистрира в обекта button, който е източник на събития от тип **ActionEvent**:

```
public class MyClass extends JApplet implements ActionListener {
    MyClass() {    // Конструктор
        // добавяне на класа като слушател на бутон “button”
        button.addActionListener(this);
        . . . . . .
    }
    public void actionPerformed(ActionEvent e) {
        // код за обработка на събитието натискане на бутон
    }
    . . . . . .
}
```

Имплементация. Събитиен модел

- Реализация чрез анонимни класове (класове, дефинирани вътре в други класове).

В този случай в декларацията на потребителския клас не е необходимо да се указва, че той реализира слушателския интерфейс. Пример:

```
public class MyClass extends JApplet {
    . . . . . .
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            . . . . . .
        }
    });
    . . . . . .
}
```

Имплементация. Събитиен модел

Интерфейсът **ActionListener** се използва за събития, възникнали в резултат на:

- натискане на бутон на мишката;
- след натискане на клавиша <Enter> в края на въведен текст (в текстово поле от тип **JTextField**);
- след избор на елемент от меню.

Имплементация. Събитиен модел

В пакета **java.awt.event** се съдържат общо 11 основни слушателски интерфейса:

- **ActionListener-actionPerformed( );**
- **AdjustmentListener-adjustmentValue-Changed( );**
- **ComponentListener**
  - **componentResized( )**
  - **componentMoved( )**
  - **componentShown( )**
  - **componentHidden( );**
- **ContainerListener-componentAdded( ), componentRemoved( );**
- **KeyListener-клавиатура: keyTyped( ),keyPressed( ),keyReleased( );**
- **MouseListener-бутони и др.;**
- **MouseMotionListener-за движения:mouseDragged( ), mouseMoved( );**
- **MenuListener-menuCanceled( ),menuDeselected( ),menuSelected( );**
- **FocusListener-focusGained( ), focusLost( );**
- **ItemListener-itemStateChanged( );**
- **WindowListener: windowOpened( ),windowClosing( ),windowClosed( ) …**

Имплементация. Събитиен модел

Онези от тях, които съдържат по-голям брой методи, са разширени с *адаптерни класове*:

- **ComponentAdapter**;
- **ContainerAdapter**;
- **KeyAdapter**;
- **MouseAdapter**;
- **MouseMotionAdapter**;
- **FocusAdapter**;
- **WindowAdapter**

Имплементация. Събитиен модел

Адаптерите изпълняват помощна роля.

Обосновка за създаване:

- Броят на методите, дефинирани в интерфейса е голям, програмистът трябва да осигури “празна” имплементация на методите (методите не съдържат код).
- В създавания клас, имплементиращ интерфейс, вместо да бъдат имплементират всички методи на интерфейса, достатъчно е да се наследи адаптерния клас и да се припокрият само онези методи, които са необходими в програмата.

Имплементация. Събитиен модел

Адаптер постановка, пример:

MouseMotionListener-два методаза движения:

- **mouseDragged( );**
- **mouseMoved( );**

Приложението изисква обработка единия-**mouseDragged( );**

Решение:

- Наследява се **MouseMotionAdapter;**
- Имплементира се **mouseDragged( )- предефинира празната имплементация на MouseMotionAdapter;**
- **Включва се пакет на адаптерните класове: java.awt.event**

Имплементация. Събитиен модел

За интерфейса **ActionListener** няма дефиниран адаптерен клас, понеже той дефинира само един единствен метод – **actionPerformed(ActionEvent evt)**. Този метод трябва да се пренапише в новосъздаваните класове (да бъде предефиниран).

Имплементация. Събитиен модел

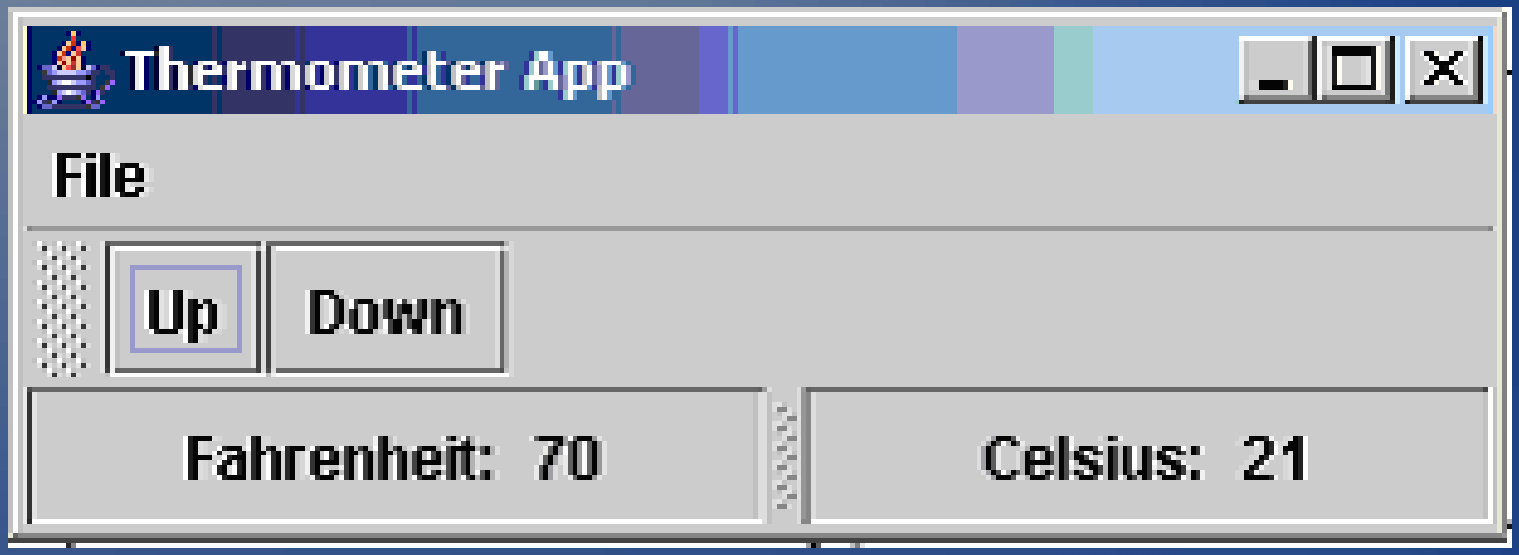
Приложенията, основани на SWING се състоят от слушатели на събития за различните компоненти, използвани в начините на изобразяване

Рамкова програма на модела

Всяко приложение се състои от:

- рамка с обозначение, работна област, меню, бутони Tool bar.
- В работната област се разполагат компоненти-текстови кутии, етикети, списъчни кутии бутони от различен тип и др.

Пример:



Рамкова програма на модела

Основни стъпки при създаване на “стандартно” приложение:

1. Създаване на обект от клас JFrame;
2. Създаване на обект от клас JMenuBar и определяне на елементите му;
3. Добавяне на JMenuBar в JFrame
4. Създаване на обект от клас JPanel
5. Дефиниране на съдържанието на JPanel
6. Добавете съдържанието на JPanel в JFrame
7. Pack и извеждане на JFrame

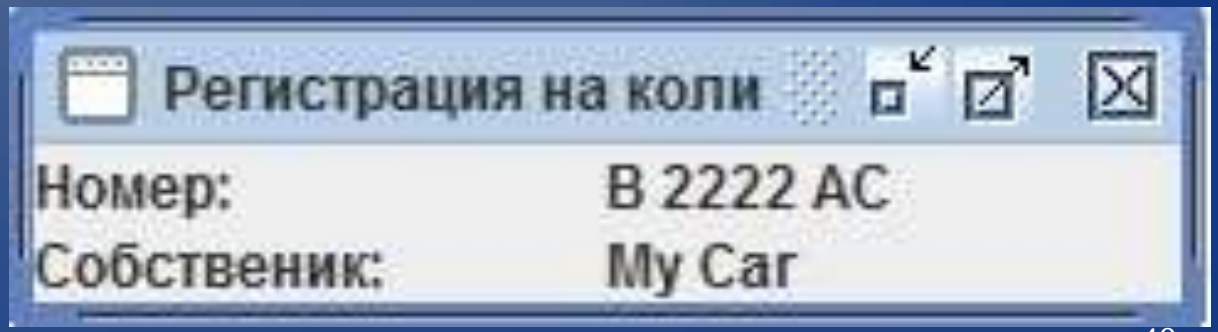
Рамкова програма на модела

Рамкова програмна част за практическото използване на JFrame

```
JFrame theFrame = new JFrame("име на приложението");   //1
JMenuBar  theMenuBar = new JMenuBar();                 //2
// код за дефиниране на елементите на менюто, обработващите му
// функции и др. Добавяне в обекта.
theFrame.setJMenuBar(theMenuBar);                      //3
JPanel   thePanel = new JPanel();                      //4
// код за дефиниране на съдържанието на панела, начина на извеждане
// дефиниране на потребителския интерфейс, команди за управление и др.
// (зависи от приложението)                            //5
theFrame.setContentPane(thePanel);                     //6
theFrame.pack();  theFrame.setVisible(true);           //7 извеждане
```

Рамкова програма на модела (пример за кодиране)

```
public class SwingCarVisualizer implements CarVisualizer {
   private JLabel plateLabel=new JLabel();
   private JLabel ownerLabel=new JLabel();
   public void createAndShowGUI() {
      JFrame.setDefaultLookAndFeelDecorated(true);
      JPanel panel=new JPanel(new GridLayout(0,2));
      panel.add(new JLabel("Номер:          "));  panel.add(plateLabel);
      panel.add(new JLabel("Собственик:     "));  panel.add(ownerLabel);
      JFrame frame = new JFrame("Регистрация на коли");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```



Рамкова програма на модела

Създаване на интерфейс за приложението. Цел:

- За да се използва многократно приложението;
- За да се гарантира исканата имплементация;
- За да се осигури разширяването му (конзолен изход)

Пример:

```
public interface CarVisualizer {
   void show(String plate,String owner);
}
```

Рамкова програма на модела

Имплементация на интерфейса в клас ConsoleCarVisualizer :

```
public class ConsoleCarVisualizer implements CarVisualizer
{
    public void show(String plate,String owner) {
      System.out.println("Plate: "+plate+" owner: "+owner);
   }

}
```

Рамкова програма на модела

Имплементация на интерфейса в клас SwingCarVisualizer:

```
public void show(String plate,String owner) {
   plateLabel.setText(plate);
   ownerLabel.setText(owner);
}
```

Рамкова програма на модела

Клас осигуряващ връзка между базата от данни и интерфейса за изобразяване:

```
public class Connnector {

 public static void join(CarLoader loader, CarVisualizer visualizer){
      visualizer.show(loader.getPlate(),loader.getOwner());
 }
   public static void main(String[] args) {
     final SwingCarVisualizer si=new SwingCarVisualizer();
     final ConsoleCarVisualizer ci=new ConsoleCarVisualizer();

      final DBCarLoader dbLoader=new DBCarLoader();
      si.createAndShowGUI();
     join(dbLoader,si);
     join(dbLoader,ci);
   }
}
```

Фак. номер _____Име_____

	Име на елемента	Стр	Изисквания за елемента от програмата
4			I. Да се състави интерфейс
4			II. Да се състави клас Частни полета: • **Публични**:
1			**Конструктори** – II.1
2			II.2 **Методи**:
2			II.3
2			II.4
2			II.5
2			II.6
4			III. Да се състави клас 2 Частни полета: •
7			**Публични членове. Конструктори**: III.1
6			**Методи**: III.2
1 1			III.3 III.4
6			III.5.
3			III.6
3			III.7
10			IV. **Главна функция** 1т - IV.1 1т - IV.2 3т - IV.3 3т IV.4 2т - IV.5

Примерен файл:

Въпроси ?