

Лекция 15.2

Многократно използване на GUI компоненти

Основни теми

- Многократно използване на графичните компоненти
- Менажери на разположението

FlowLayout

BorderLayout

GridLayout

JTextArea **КОМПОНЕНТИ**

Задачи

Литература:

– *Java How to Program, Sixth Edition, глава 11*

1 Групиране на GUI компоненти и реализация на функционалност в JPanel

Всяка JComponent-а е контейнер

- Съдържа в себе си други компоненти**
- Реализира желана за изпълнение функционалност**

Позволява създаване на библиотеки от готови компоненти и “вмъкване” на компоненти при нужда

За групиране и компоненти най- често се използва JPanel като контейнер за други компоненти или реализация на желано поведение (“функционалност”) на компонента

1 Групиране на GUI компоненти и реализация на функционалност в JPanel

Пример:

Създаваме група от компоненти за пресмятане обща стойност на покупка по въведени на цена и количество на стока

Целта е да използваме тези компоненти в други приложения, включително аплети.

Започваме със *създаването на Java приложение, което ще създаде необходимата библиотека* от графични компоненти

1 Групиране на GUI компоненти и реализация на функционалност в JPanel

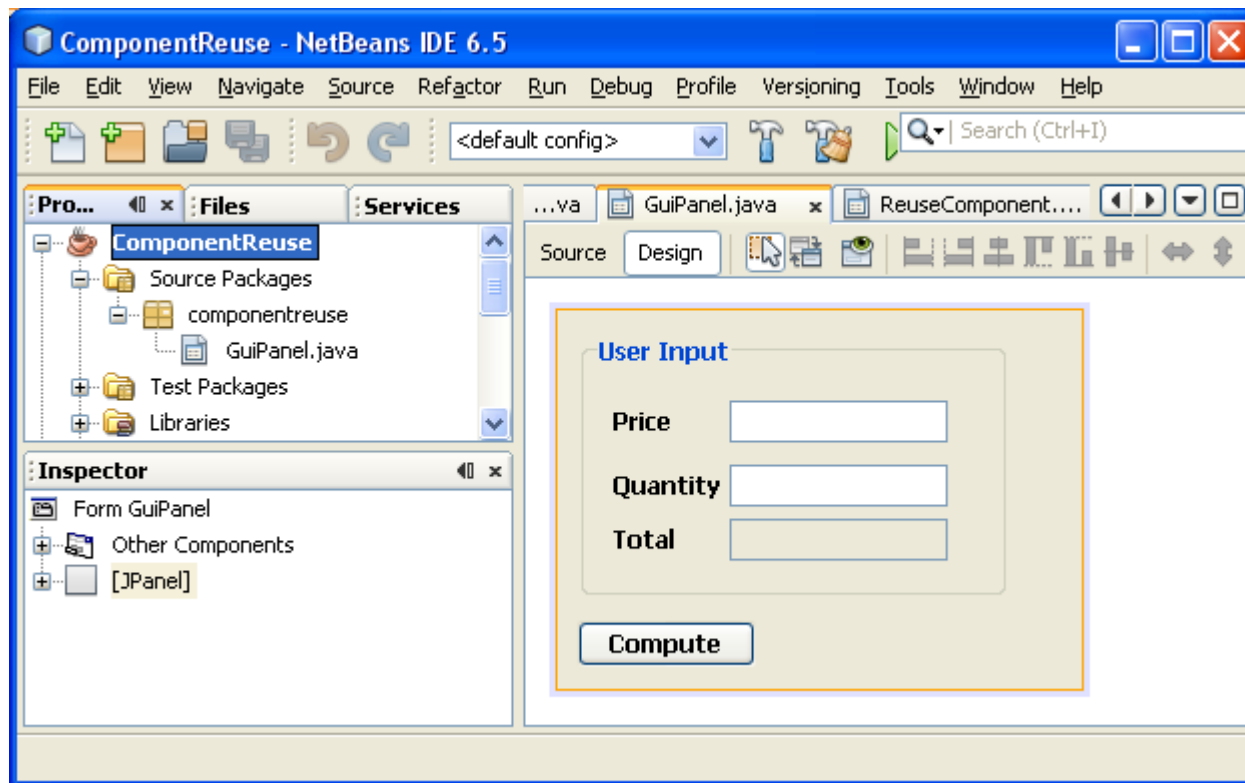
Създаване на библиотека от графични компоненти:

- същата последователност от действия, както създаване на всяка друга библиотека (виж лекция 11.1)
- **разликата е**, че **вместо обикновен Java файл използваме JPanel Form** за визуално редактиране на графичния контекст

Пример:

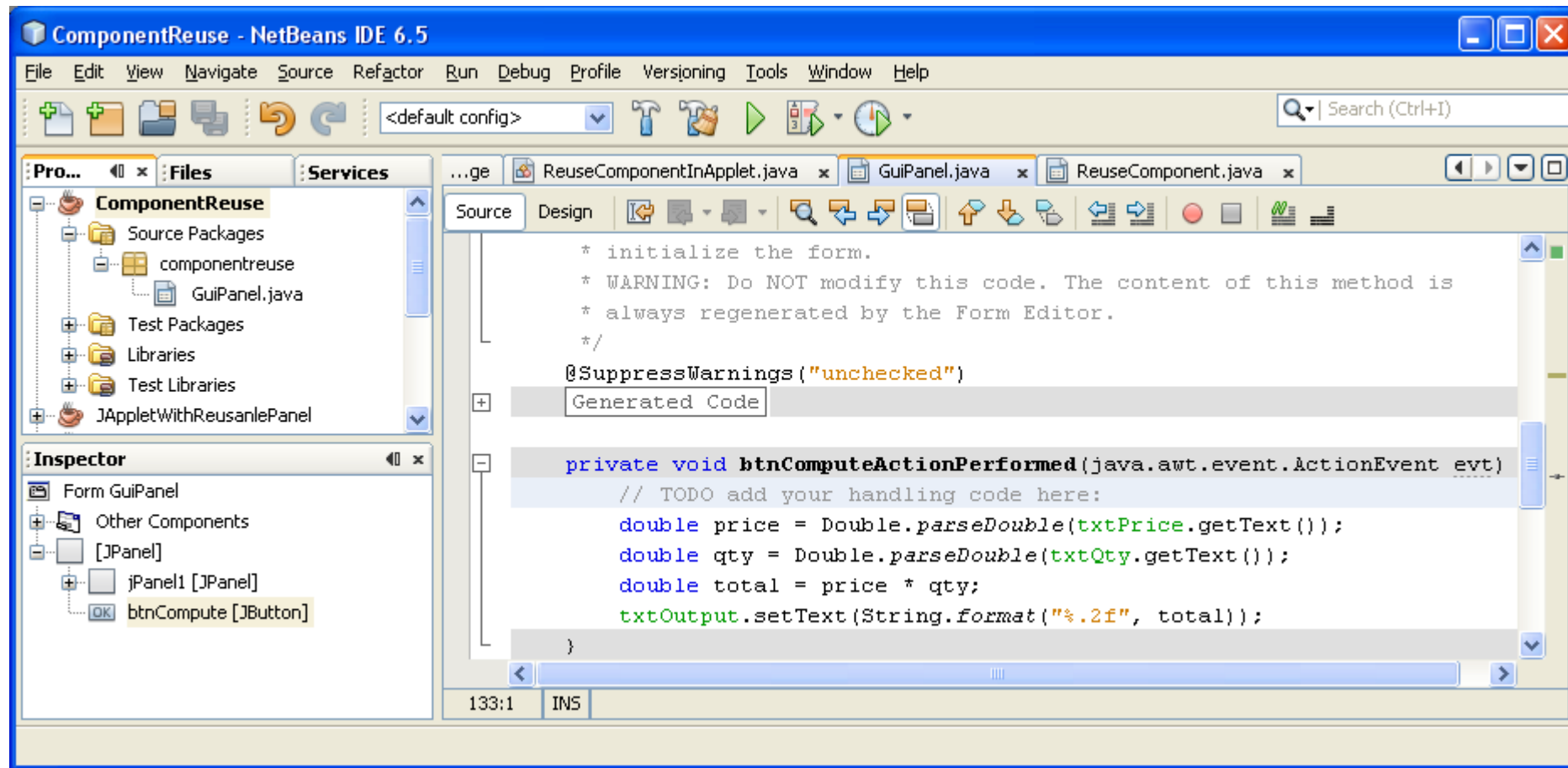
Нека е създаден Java Application проект, съдържащ JPanel Form , показан на следващия слайд

JPanel Form с графични компоненти и реализирано събитие Action за бутона Compute



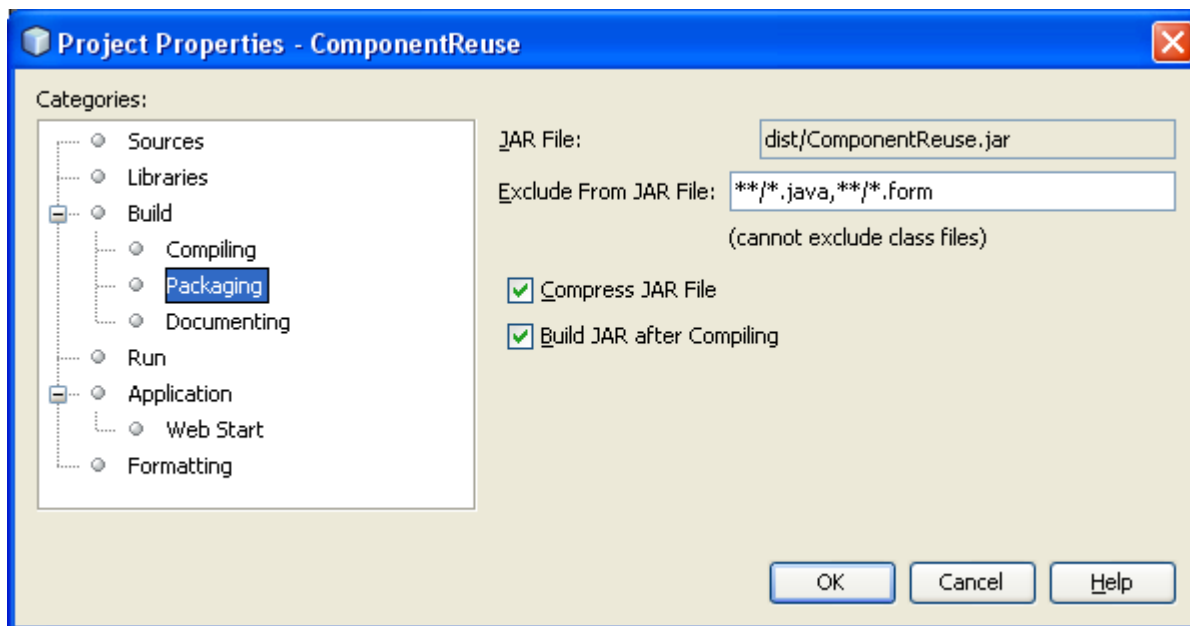
JPanel Form с графични компоненти и реализирано събитие Action за бутона Compute

7



1 Групиране на GUI компоненти и реализация на функционалност в JPanel

Настройваме свойствата по създаване на пакет (библиотека на Java), както обикновено (лекция 11.1)



1 Групиране на GUI компоненти и реализация на функционалност в JPanel

Изпълнявате **Build**, за да се създаде нужния JAR файл, съдържащ така дефинираната графична компонента (**JPanel Form** в случая)

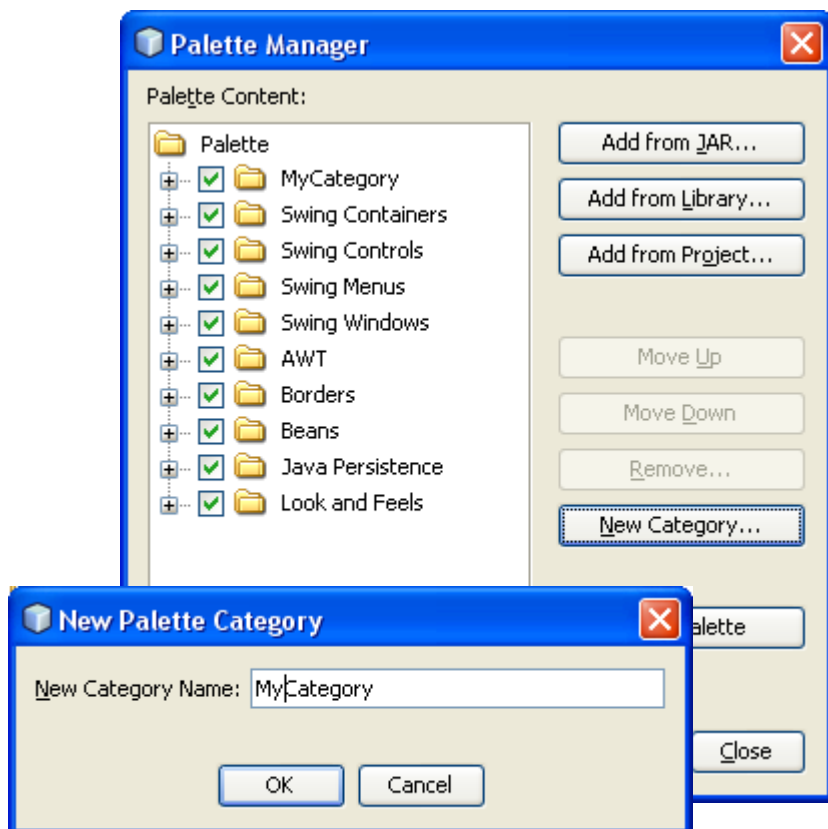
Остава **да се добави тази компонента** към **Palette** с другите Swing графични компоненти

1 Групиране на GUI компоненти и реализация на функционалност в JPanel

За целта:

- отваряте създадената графична компонента (**JPanel Form** в случая) в Design
- кликнете с десния бутон на мишката върху **Palette** (ако не се вижда, Ctrl-Shift- 8) и изберете **Palette Manager**
- използвайте **Palette Manager** за създаване на нова категория (бутон **New Category**) от компоненти, примерно, **My Category**

1 Групиране на GUI компоненти и реализация на функционалност в JPanel



1 Групиране на GUI компоненти и реализация на функционалност в JPanel

Остава да добавите вашата GUI компонента в тази нова категория

- кликнете с десния бутон на мишката върху **java** **файла** (в случая, **GuiPanel.java**) и изберете **Tools->Add To Palette** от менюто
- изберете желаната категория, от **Swing** компоненти от диалоговия прозорец, примерно, **My Category** и **потвърдете** с бутона **OK**

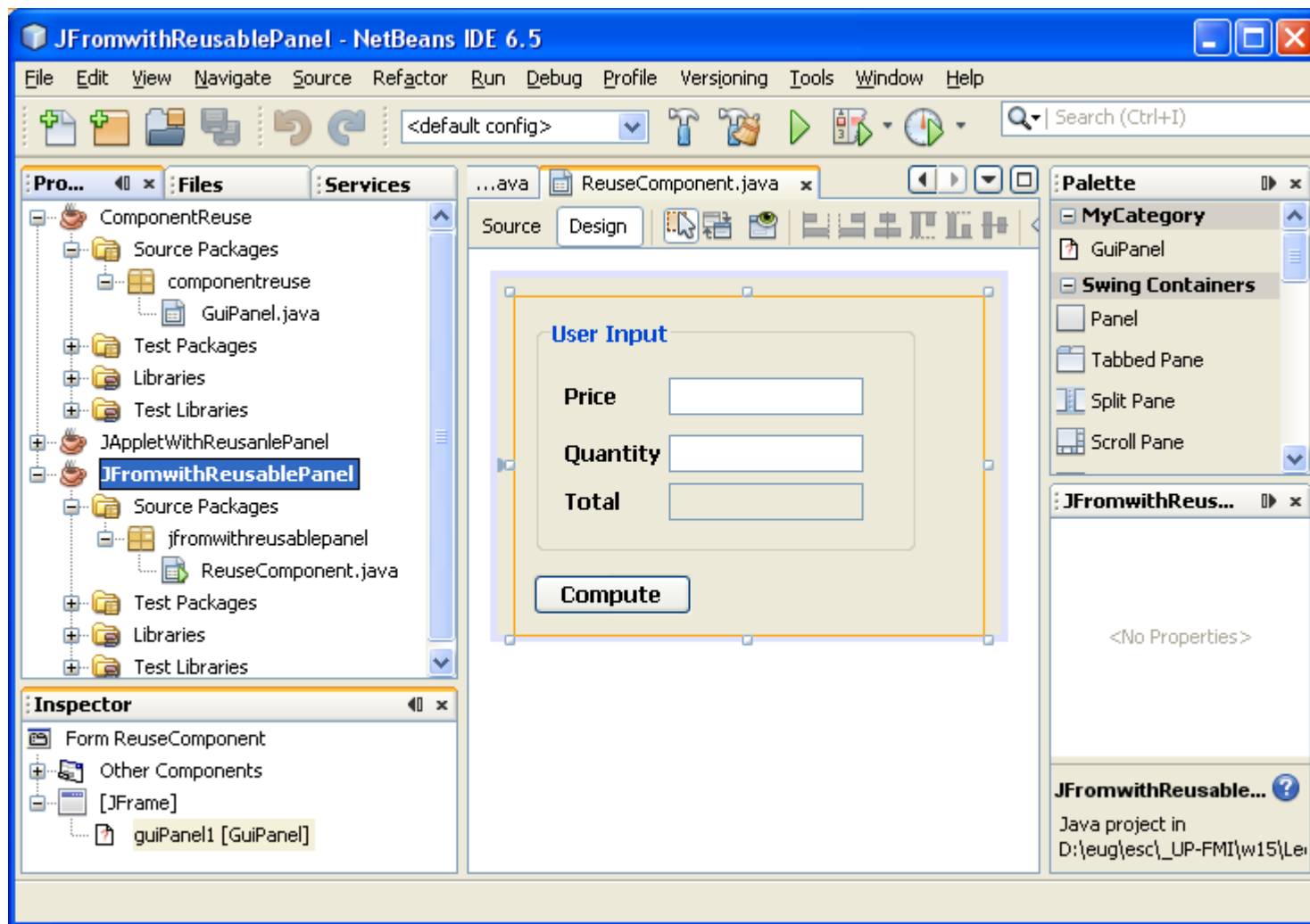
Така създадохме графична компонента и я добавихме към **Palette** от **Swing** компоненти

1.1 Използване на GUI компоненти в JFrame Form приложение

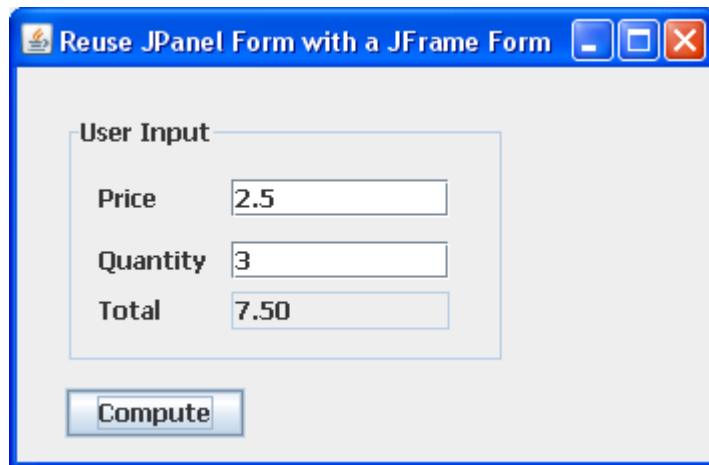
За целта:

1. Създайте Java приложение и добавете **JFrame Form** клас , например **ReuseComponent.java** (**виж лекция 14.1**)
2. **Изберете** потребителската компонента от Palette, както всяка друга компонента и я **добавете** към **JFrame Form**
3. **Build** и **Run** Java приложението с така създадената **JFrame Form**

1.1 Използване на GUI компоненти в Jframe Form приложение



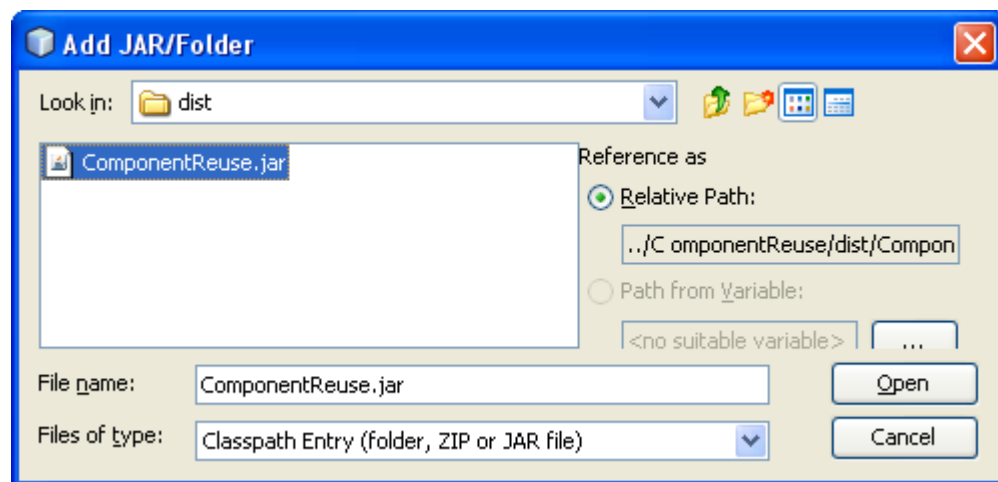
1.1 Използване на GUI компоненти в JFrame Form приложение



1.2 Използване на GUI компоненти в JApplet приложение

За целта:

1. Създайте Java приложение и добавете JApplet клас, например `ReuseComponentInApplet.java` (виж лекция 15.1)
2. **Настройте** Java приложение да използва JAR файла от приложението, в което създадохме `JPanel` компонентата (виж лекция 11.1)



1.2 Използване на GUI компоненти в JApplet приложение

3. Добавете в JApplet-a **import** към съответния пакет ,например,

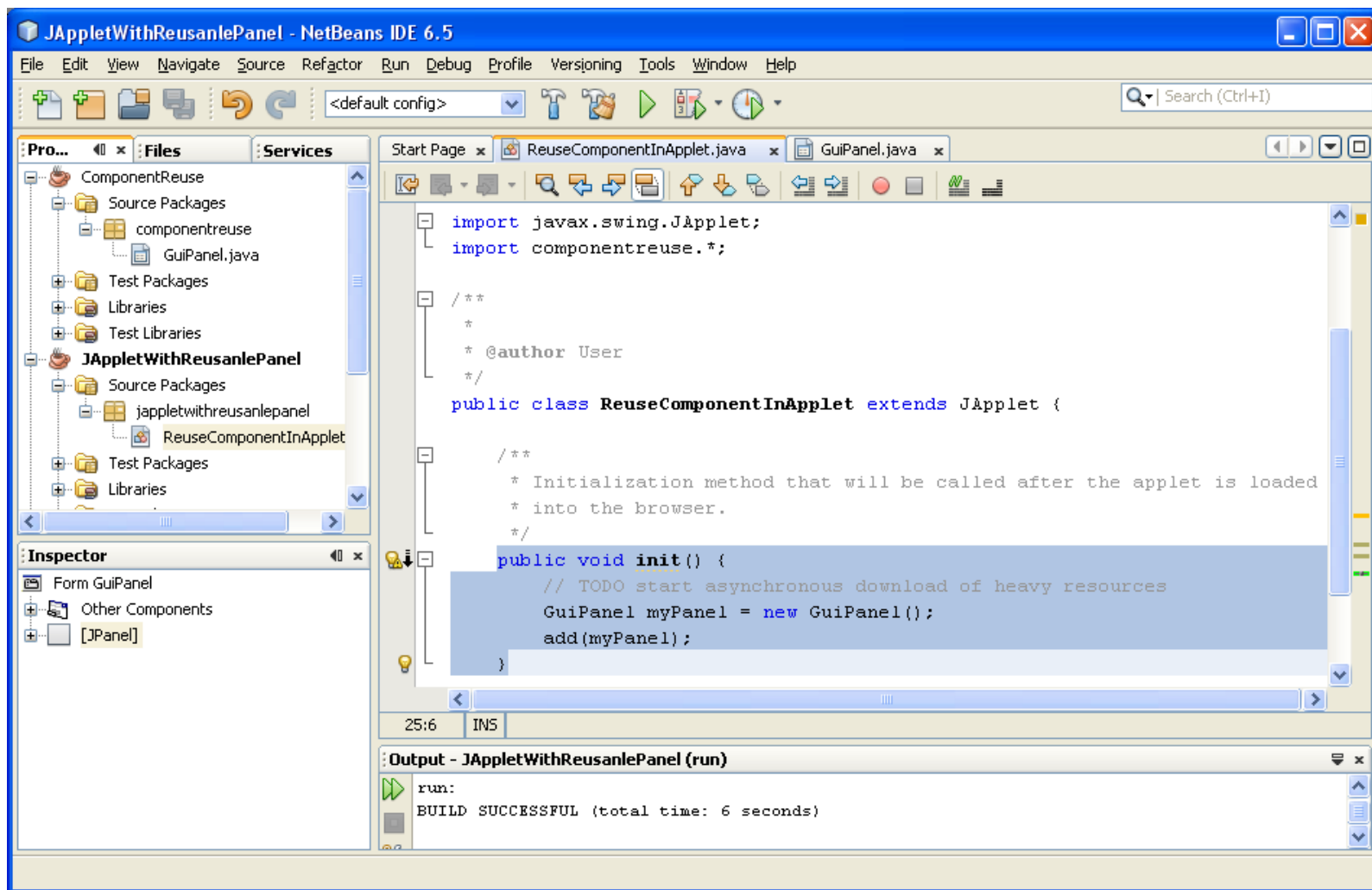
```
import componentreuse.*;
```

4. Създайте инстанция от графичната компонента и я добавете към JApplet-a, например,

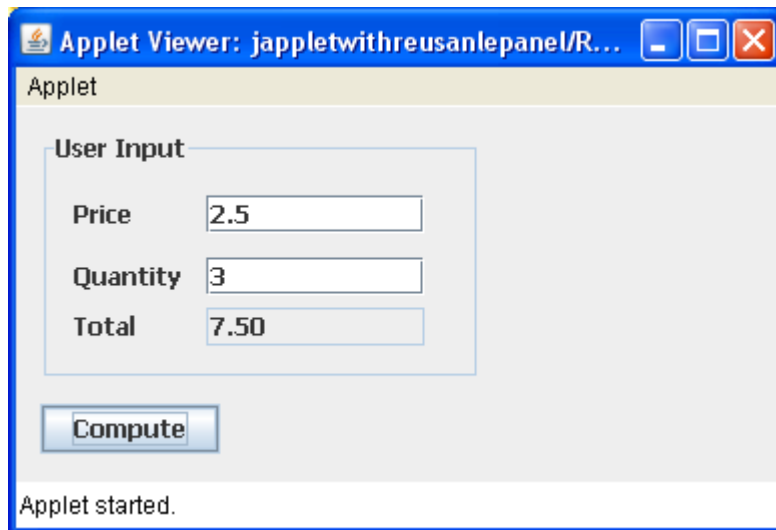
```
public void init() {  
    // TODO start asynchronous download of heavy resources  
    GuiPanel myPanel = new GuiPanel();  
    add(myPanel);  
}
```

5. Изпълнете аплета

1.2 Използване на GUI компоненти в JApplet приложение



1.2 Използване на GUI компоненти в JApplet приложение



2 Менажери за управление на разположението

- Производни на interface `LayoutManager`
 - Позволяват да се дефинира разположението на графичните компоненти (*текущо състояние*)
 - Позволява да се дефинират основни операции за получаване на желаното разположение (*поведение*)

Стил на програмиране 11.16

Повечето IDE позволяват да се използват GUI графични редактори за работа с графични компоненти

Позволяват по-голяма свобода при дефинирането на разположението отколкото вградените менажери за разположение в Java.

2.1 FlowLayout

- FlowLayout
 - Най- прост менажер за управление на разположението на компонентите добавени в контейнера
 - Компонетите се разполагат отляво надясно в реда на добавянето им в контейнера
 - Компонентите могат допълнително да се подравняват отляво, центрират и подравняват отдясно.

Менажер	Описание
FlowLayout	Подразбиращ се за <code>java.awt.FlowLayout</code> . Компонетите се разполагат отляво надясно в реда на добавянето им в контейнера. Може да се задава и реда на подреждане в <code>Container</code> при добавянето им с метода <code>add</code> , който взима 3 аргумент <code>Component</code> и целочислен индекс за позиция по ред.
BorderLayout	Подразбиращ се <code>java.awt.BorderLayout</code> . Подрежда компонентите в следните области : <code>NORTH</code> , <code>SOUTH</code> , <code>EAST</code> , <code>WEST</code> и <code>CENTER</code> . Във всяка от тях може да има само по една компонента
GridLayout	Подрежда компонентите по ред и колонка.

Fig. 11.38 | Менажери за разположение на компоненти.

Outline

BorderLayout Frame.java

(1 от 2)

```

1  // Fig. 11.41: BorderLayoutFrame.java
2  // Demonstrating BorderLayout.
3  import java.awt.BorderLayout;
4  import java.awt.event.ActionListener;
5  import java.awt.event.ActionEvent;
6  import javax.swing.JFrame;
7  import javax.swing.JButton;
8
9  public class BorderLayoutFrame extends JFrame implements ActionListener
10 {
11     private JButton buttons[]; // array of buttons to hide portions
12     private final String names[] = { "Hide North", "Hide South",
13         "Hide East", "Hide West", "Hide Center" };
14     private BorderLayout layout; // borderlayout object
15
16     // set up GUI and event handling
17     public BorderLayoutFrame()
18     {
19         super( "BorderLayout Demo" );
20
21         layout = new BorderLayout( 5, 5 ); // 5 pixel gaps
22         setLayout( layout ); // set frame layout
23         buttons = new JButton[ names.length ]; // set size of array
24
25         // create JButtons and register listeners for them
26         for ( int count = 0; count < names.length; count++ )
27         {
28             buttons[ count ] = new JButton( names[ count ] );
29             buttons[ count ].addActionListener( this );
30         } // end for

```

Декларира BorderLayout променлива

Създава BorderLayout

Задава начин на разположение на
компонентите

Регистрира обработване на
събитие



Outline

```

31 add( buttons[ 0 ], BorderLayout.NORTH ); // add button to north
32 add( buttons[ 1 ], BorderLayout.SOUTH ); // add button to south
33 add( buttons[ 2 ], BorderLayout.EAST ); // add button to east
34 add( buttons[ 3 ], BorderLayout.WEST ); // add button to west
35 add( buttons[ 4 ], BorderLayout.CENTER ); // add button to center
36 } // end BorderLayoutFrame constructor
37
38 // handle button events
39 public void actionPerformed((ActionEvent event) )
40 {
41     // check event source and layout content pane correspondingly
42     for ( JButton button : buttons )
43     {
44         if ( event.getSource() == button )
45             button.setVisible( false ); // hide button clicked
46         else
47             button.setVisible( true ); // show other buttons
48     } // end for
49
50     layout.layoutContainer( getContentPane() ); // layout content pane
51 } // end method actionPerformed
52 } // end class BorderLayoutFrame

```

Добавя бутони с константи за
разположение

Прави бутона който е избран невидим

Прави другите бутони видими

Обновява разположението
(след промяната)

BorderLayout
Frame.java

(2 от 2)



Outline

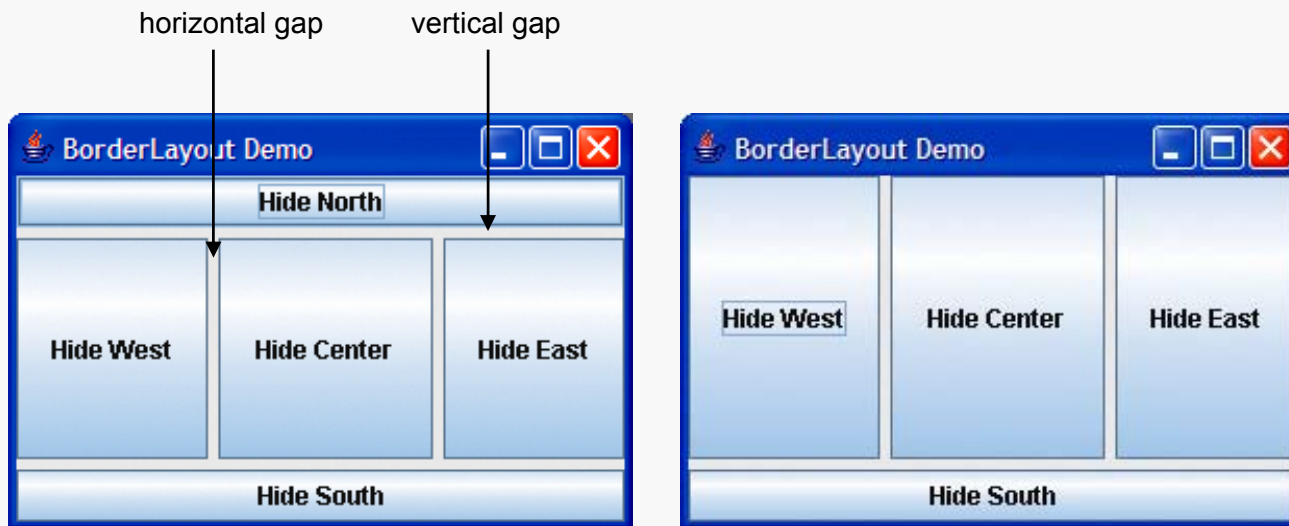
BorderLayout Demo.java

(1 of 2)

```

1 // Fig. 11.42: BorderLayoutDemo.java
2 // Testing BorderLayoutFrame.
3 import javax.swing.JFrame;
4
5 public class BorderLayoutDemo
6 {
7     public static void main( String args[] )
8     {
9         BorderLayoutFrame borderLayoutFrame = new BorderLayoutFrame();
10        borderLayoutFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11        borderLayoutFrame.setSize( 300, 200 ); // set frame size
12        borderLayoutFrame.setVisible( true ); // display frame
13    } // end main
14 } // end class BorderLayoutDemo

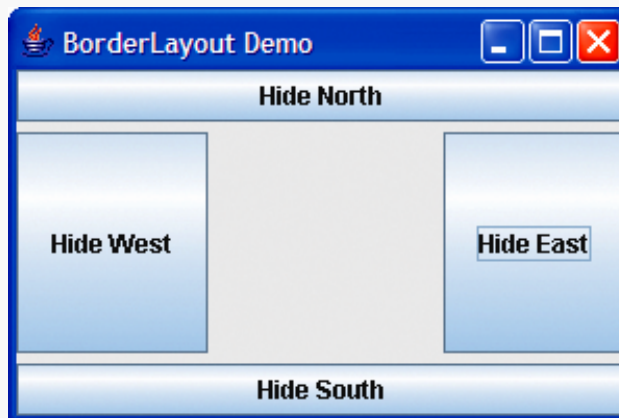
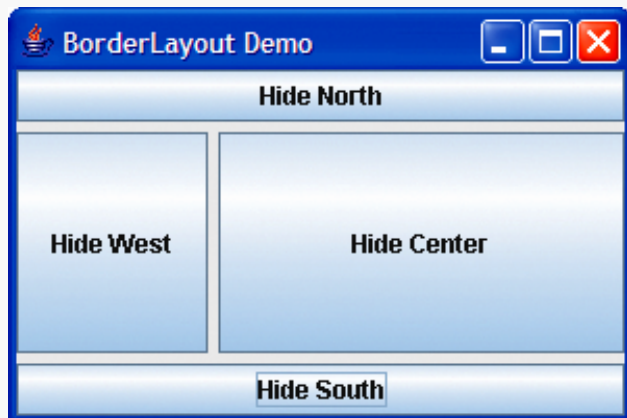
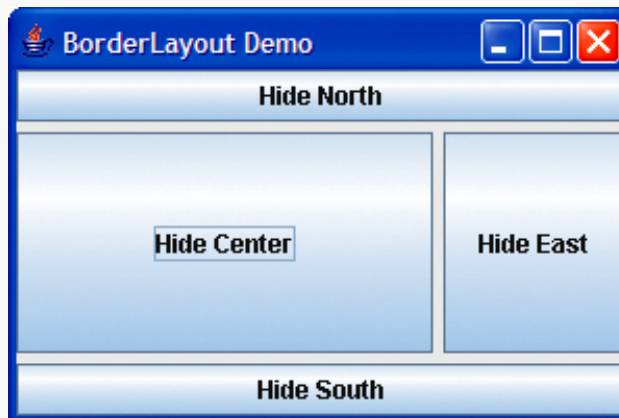
```



Outline

BorderLayout
Demo.java

(2 от 2)



2.2 Използване на панели

- По- сложен модел на графичен интерфейс изисква групиране на компоненти по логическото им предназначение
- Може да се използват многократно

Outline

Panel Frame. java

(1 от 2)

```
1 // Fig. 11.45: PanelFrame.java
2 // Using a JPanel to help lay out components.
3 import java.awt. GridLayout;
4 import java.awt. BorderLayout;
5 import javax.swing. JFrame;
6 import javax.swing. JPanel;
7 import javax.swing. JButton;
8
9 public class PanelFrame extends JFrame
10 {
11     private JPanel buttonJPanel; // panel to hold buttons
12     private JButton buttons[]; // array of buttons
13
14     // no-argument constructor
15     public PanelFrame()
16     {
17         super( "Panel Demo" );
18         buttons = new JButton[ 5 ]; // create buttons array
19         buttonJPanel = new JPanel(); // set up panel
20         buttonJPanel.setLayout( new GridLayout( 1, buttons.length ) );
21     }
22 }
```

Декларира JPanel за бутони

Създава JPanel

Задава подреждане в един ред



Outline

Panel Frame. java

(2 от 2)

```
22 // create and add buttons
23 for ( int count = 0; count < buttons.length; count++ )
24 {
25     buttons[ count ] = new JButton( "Button " + ( count + 1 ) );
26     buttonJPanel.add( buttons[ count ] ); // add button to panel
27 } // end for
28
29 add( buttonJPanel , BorderLayout.SOUTH ); // add panel to JFrame
30 } // end Panel Frame constructor
31 } // end class Panel Frame
```

Добавя бутон към панела



Добавя панела към приложението



Outline

Panel Demo. java

```
1 // Fig. 11.46: Panel Demo.java
2 // Testing Panel Frame.
3 import javax.swing.JFrame;
4
5 public class Panel Demo extends JFrame
6 {
7     public static void main( String args[] )
8     {
9         Panel Frame panel Frame = new Panel Frame();
10        panel Frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11        panel Frame.setSize( 450, 200 ); // set frame size
12        panel Frame.setVisible( true ); // display frame
13    } // end main
14 } // end class Panel Demo
```



Задачи

1. **Напишете многократно използвана графична компонента, която позволява да се пресметне и изведе месечната вноска по сключен договор за кредит, при което се въвежда сумата на кредита, годишната лихва и брой години за изплащане на кредита. Демонстрирайте приложението на графичната компонента в Java аplet и JFrame приложения**
2. **Напишете многократно използвана графична компонента, която представя калкулатор. Демонстрирайте приложението на графичната компонента в Java аplet и JFrame приложения**

Задачи

3. Напишете Java *JFrame* приложение, което изобразява панел в средата, на който е изписан символа 'А' (писането на Текст върху графична компонента става с командата *drawString(x,y, stringToDraw)* на *Graphics* обект *g* по подобие на *drawOval()*, *drawrect()*, *drawLine()* и пр.) Нека при натиснат на бутон на мишката върху символа 'А' този символ да се мести заедно с курсора на мишката върху панела до отпускане на бутона на мишката. *В долната част на прозореца добавете етикет, който да извежда текущите координати при движение на мишката*
4. Напишете Java *JFrame* приложение , което е вариант на задача 1, при който символа се мести нагоре, надолу, наляво и надясно в панела при използване на съответните стрелки от клавиатурата. Като упътване използвайте, че методът *getKeyCode()* на *KeyEvent* връща *KeyEvent.VK_DOWN*, *KeyEvent.VK_UP*, *KeyEvent.VK_LEFT*, *KeyEvent.VK_RIGHT* целочислени константи при натискане на стрелките нагоре, надолу, наляво и надясно. *В долната част на прозореца добавете етикет, който да извежда текущите координати при натискане на стрелките*