BLM 2562 OBJECT ORIENTED PROGRAMMING LECTURE NOTES

Assist. Prof. Dr. Yunus Emre SELÇUK

GUI PROGRAMMING WITH JAVA
Part I – Frames and Panels

1

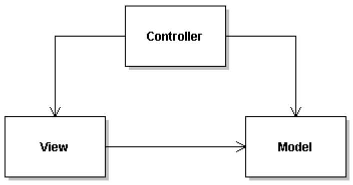
GUI PROGRAMMING WITH JAVA

- General information:
  - GUI editors in Java are not as easy to use as Microsoft's programming environments.
  - Learning curve of GUI operations in Java is rather steep.
  - Built-in GUI packages in Java are:
    - The java.awt package: It has become obsolote
    - The javax.swing package:
      - Current one since Java 1.2
      - Based on classes in java.awt.
      - Class names begin with J
        - javax.swing.JFrame
        - java.awt.Frame
        - Don't forget the J!
  - There are other 3rd party packages as well
    - such as SWT of the eclipse foundation
    - not to be covered in this course

2

GUI PROGRAMMING WITH JAVA

- General architecture of the Swing framework is based on the Model-View-Controller (MVC) design pattern.
- What is a design pattern?
  - It is a general reusable solution to a commonly occurring problem within a given context in software design.
  - A design pattern is not a finished design that can be transformed directly into code.
  - It can rather be seen as a template for leading you to a "good" design.
- What is MVC?



- Model class:
  - Represents raw information (data)
- View class:
  - Represents different representations of the data
- Controller class:
  - Receives user commands and handles them by reading and/or modifying data
- For more information, please refer to the classic GoF book and others.

3

GUI PROGRAMMING WITH JAVA

- Creating a basic window:
  - The class javax.swing.JFrame represents a basic window.
  - Create your own windows by inheriting from this class.

```
package oop08;
import javax.swing.*;
@SuppressWarnings("serial")
public class SimpleWindow extends JFrame {
   public static final int DEFAULT_WIDTH = 300;
   public static final int DEFAULT_HEIGHT = 200;
   public SimpleWindow( ) {
      setSize( DEFAULT_WIDTH, DEFAULT_HEIGHT );
   }
   public static void main(String[] args) {
      SimpleWindow window = new SimpleWindow( );
      window.setTitle("A Simple Window");
      window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      window.setVisible(true);
   }
}
```

4

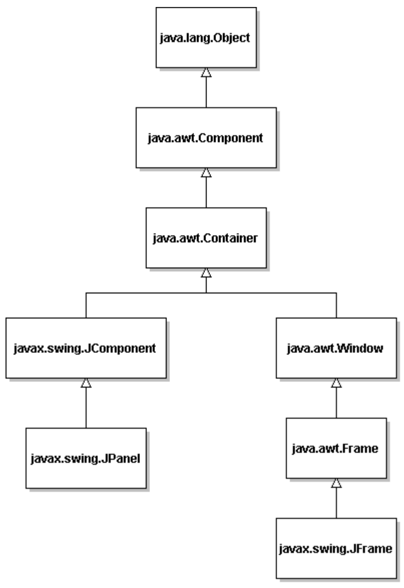
GUI PROGRAMMING WITH JAVA

- What have we just done?
  - We must set the size of the window in its constructor or in the main method by using the **setSize(int width, int height)** method.
    - Otherwise, its size will be 0 by 0!
  - We must show the window by using the **setVisible(boolean isVisible)** method.
    - Otherwise, it will not be shown in the screen.
  - We can set it's title by using the **setTitle(String title)** method.
  - We must use **setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)**
    - Otherwise the program will continue to execute (check the red stop button in eclipse)

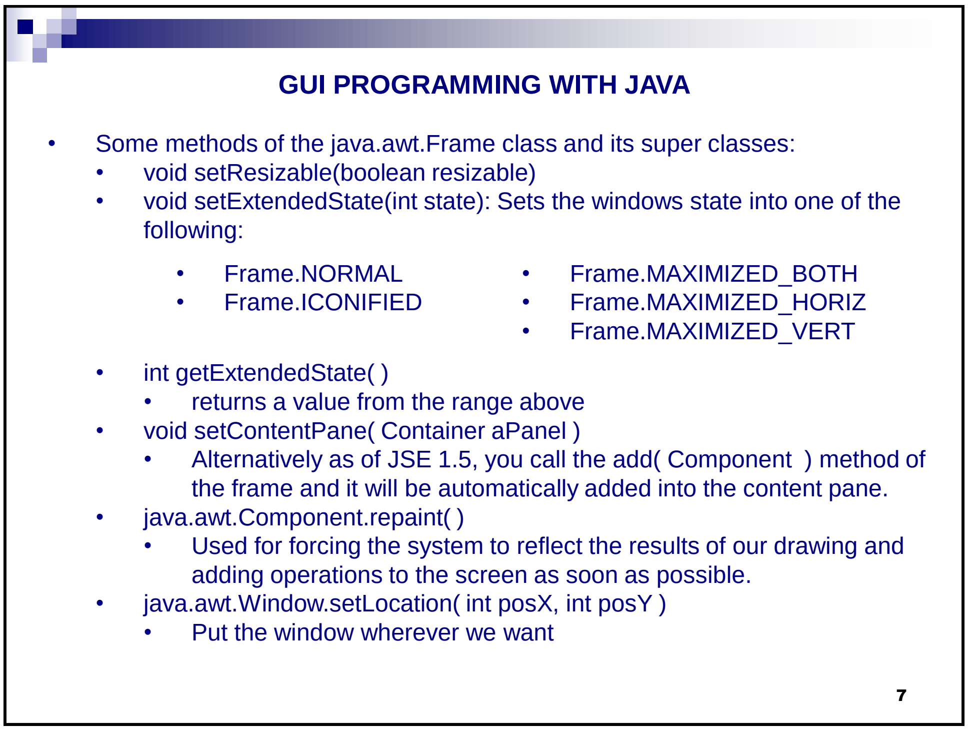
5

GUI PROGRAMMING WITH JAVA

- Frames and Panels:
  - The frame contains a top-level panel for you to put contents in it.
    - Hence its named the content panel.
    - A panel can contain other panels as well.
    - This opens up some possibilities in layout organization and code reuse.
  - The inheritance tree is shown on the left.



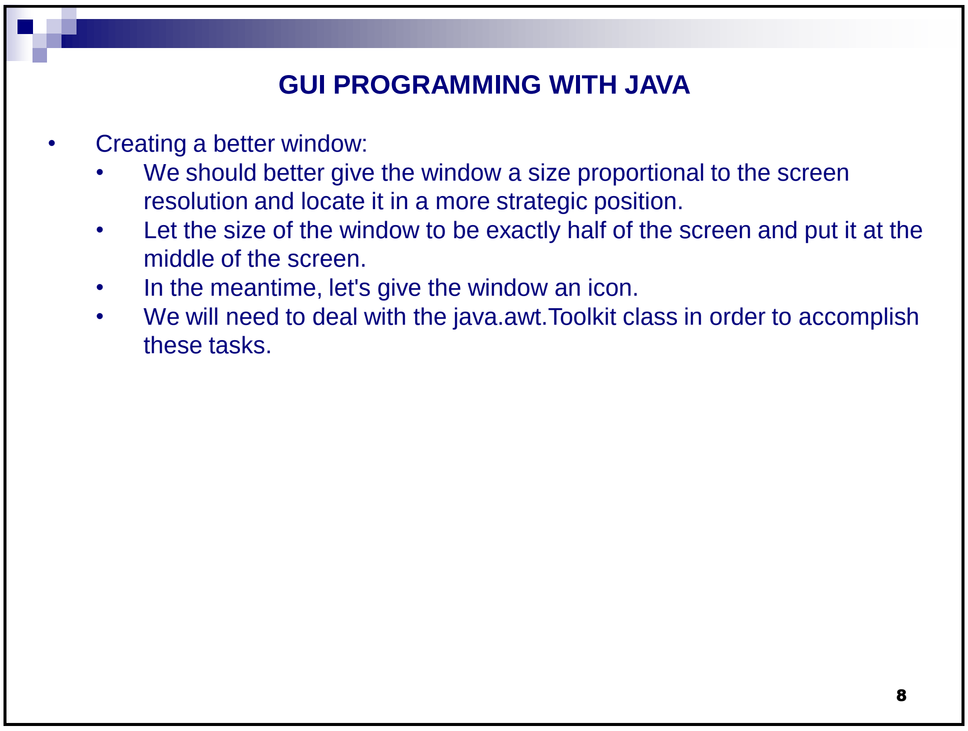
6



GUI PROGRAMMING WITH JAVA

- Some methods of the java.awt.Frame class and its super classes:
  - void setResizable(boolean resizable)
  - void setExtendedState(int state): Sets the windows state into one of the following:
    - Frame.NORMAL
    - Frame.ICONIFIED
    - Frame.MAXIMIZED_BOTH
    - Frame.MAXIMIZED_HORIZ
    - Frame.MAXIMIZED_VERT
  - int getExtendedState( )
    - returns a value from the range above
  - void setContentPane(Container aPanel)
    - Alternatively as of JSE 1.5, you call the add(Component) method of the frame and it will be automatically added into the content pane.
  - java.awt.Component.repaint( )
    - Used for forcing the system to reflect the results of our drawing and adding operations to the screen as soon as possible.
  - java.awt.Window.setLocation(int posX, int posY)
    - Put the window wherever we want

7



GUI PROGRAMMING WITH JAVA

- Creating a better window:
  - We should better give the window a size proportional to the screen resolution and locate it in a more strategic position.
  - Let the size of the window to be exactly half of the screen and put it at the middle of the screen.
  - In the meantime, let's give the window an icon.
  - We will need to deal with the java.awt.Toolkit class in order to accomplish these tasks.

8

GUI PROGRAMMING WITH JAVA

- Getting a Toolkit instance and learning the screen size:

```
Toolkit kit = Toolkit.getDefaultToolkit();
Dimension screenSize = kit.getScreenSize();
int screenHeight = screenSize.height;
int screenWidth = screenSize.width;
```

- Loading an image and designating it as an icon:

```
Image img = kit.getImage("icon.jpg");
setIconImage(img);
```

- Regarding to the relative location of the image:
  - Put in the project directory if working with an IDE

9

GUI PROGRAMMING WITH JAVA

- Putting them alltogether:

```
package oop08;
import java.awt.*;
import javax.swing.*;
@SuppressWarnings("serial")
public class SizedWindowWithIcon extends JFrame {
    public SizedWindowWithIcon( ) {
        Toolkit kit = Toolkit.getDefaultToolkit();
        Dimension screenSize = kit.getScreenSize();
        int screenHeight = screenSize.height;
        int screenWidth = screenSize.width;
        setSize(screenWidth / 2, screenHeight / 2);
        setLocation(screenSize.width / 4, screenSize.height / 4);
        Image img = kit.getImage("icon.jpg");
        setIconImage(img);
        setTitle("A Sized Window With Icon");
    }
    public static void main(String[] args) {
        SizedWindowWithIcon window = new SizedWindowWithIcon( );
        window.setVisible(true);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

10

GUI PROGRAMMING WITH JAVA

- The correct way to initialize Swing:
  - Swing components are rather complex, therefore we need to initialize GUI components in a separate thread.
  - This should be done from the *event dispatch thread.*
    - i.e., the thread of control that passes events such as mouse clicks and keystrokes to the user interface components.
  - Here is the code fragment used for this purpose:

```
EventQueue.invokeLater(new Runnable() {
    public void run() {
        /* statements */
    }
});
```

- Now, what was that?
  - We have just created *an anonymous inner class which implements the Runnable interface*.
  - You can omit this magic code and go ahead as our previous examples and everything can go normal, but better be safe than sorry.

11

GUI PROGRAMMING WITH JAVA

- Drawing something *into the panel*:
  - You cannot draw into the frame directly, you draw into its top-level panel.
  - You should create a panel by inheriting from JPanel and overriding its paintComponent method.
  - Moreover, a panel is responsible from painting all components inside it.
    - The easiest way to do this is by calling the super.paintComponent method as the first command in the paintComponent method.
  - Drawing is done by using the parameter of type Graphics.
    - Displaying text is considered a special kind of drawing.
    - The Graphics class has a drawString method for this purpose:
      - g.drawString(String text, int xPos, int yPos)

```
class NotHelloWorldPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawString("Not a Hello, World program", 75, 100);
    }
}
```

12

GUI PROGRAMMING WITH JAVA

- Putting them alltogether:

```
package oop08;
import javax.swing.*;
import java.awt.*;
@SuppressWarnings("serial")
public class NotHelloWorld {
   public static void main(String[] args) {
      EventQueue.invokeLater(new Runnable() {
         public void run() {
            NotHelloWorldFrame frame = new NotHelloWorldFrame();
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);
         }
      });
   }
}
//to be continued in the next slide
```

13

GUI PROGRAMMING WITH JAVA

- Putting them alltogether (cont'd):

```
//A frame that contains a message panel
class NotHelloWorldFrame extends JFrame {
    public NotHelloWorldFrame(){
        setTitle("NotHelloWorld");
        setSize(300, 200);
        NotHelloWorldPanel panel = new NotHelloWorldPanel();
        add(panel);
    }
}
//A component that displays a message.
class NotHelloWorldPanel extends JPanel {
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawString("Not a Hello, World program", 60, 80);
    }
}
```

- PS: We have coded all classes in NotHelloWord.java

14

GUI PROGRAMMING WITH JAVA

- Handcoding coordinates of the text is somewhat awkward
  - Those who need to learn more about fonts, font metrics and details of working with 2D objects such as lines, rectangles, etc. can refer to the rest of the Section 7 of Core Java Vol.I
- We will proceed by learning how to react user input in the next week.

15

BLM 2562 OBJECT ORIENTED PROGRAMMING LECTURE NOTES
Assist. Prof. Dr. Yunus Emre SELÇUK

GUI PROGRAMMING WITH JAVA
Part II – Event Handling

16

GUI PROGRAMMING WITH JAVA

- The examples in the previous section do not interact with user.
  - We need to take input from the user by means of graphical controls such as text boxes, buttons, etc.
  - But before that, we need to know about events and how to process them.
- All operating systems (OS) with GUI support work in terms of *events* and *actions*.
  - Each movement of the mouse, each click, each key press and release, is an event.
  - Whenever an event occurs, someone should make an action.
    - The OS catches all events and make and action.
    - Moreover, OS lets programming languages to intervene, i.e. it lets us to *handle* those *events*.
    - Different programming languages have different approaches for *event handling*.

17

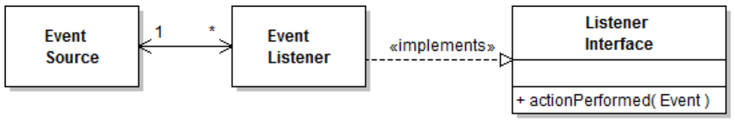
GUI PROGRAMMING WITH JAVA

- Event handling with Java:
  - Different types of event sources create different types of events.
    - GUI components such as JButton, JCheckBox, etc.
    - OS itself: Mouse and keyboard events, etc.
  - Event listeners listen to events of their preference.
    - Listeners should implement sub-interfaces of java.util.EventListener interface which are in java.awt.event package.
    - Code for handling events are written in the method(s) defined in those interfaces.
  - Programmer relates event sources with event listeners.
    - The general format is:
      - eventSourceObj.**add**EventName**Listener(**aListenerObj**);**

18

GUI PROGRAMMING WITH JAVA

- Relating event sources with event listeners:
  - Listeners of GUI events are associated with the GUI objects which is the source of that event.
  - Listeners of OS events are associated with JFrame or JPanel objects.
  - There can be *..* relation between event sources and listeners, but 1..* is suggested.

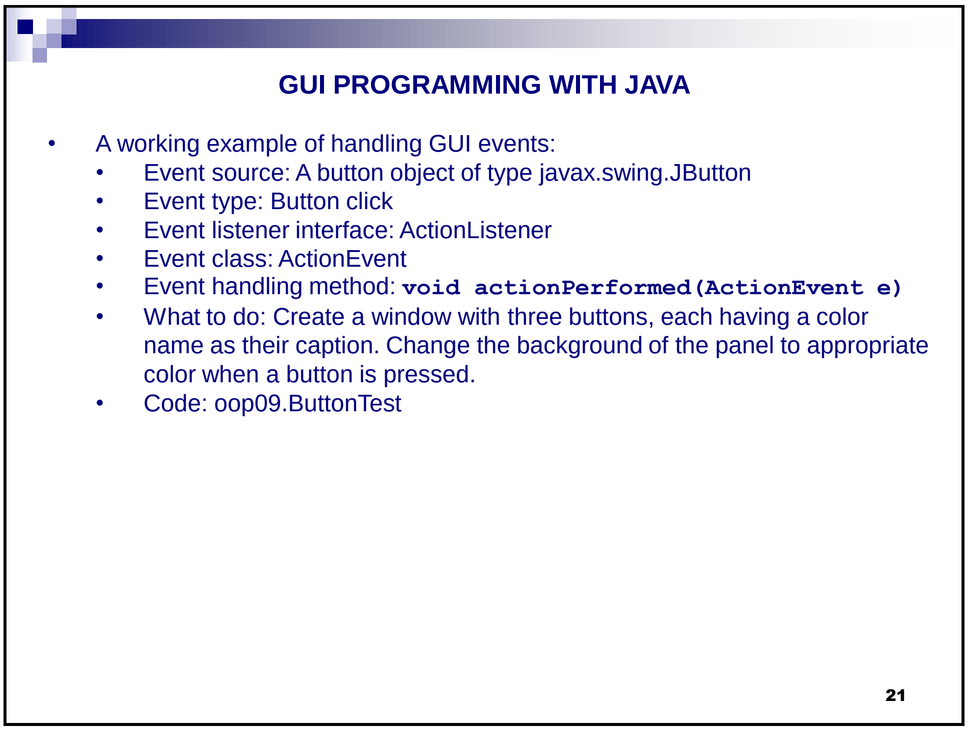


19

GUI PROGRAMMING WITH JAVA

- Events needed frequently …
  - ActionEvent
  - AdjustmentEvent
  - FocusEvent
  - ItemEvent
  - KeyEvent
  - MouseEvent
  - MouseWheelEvent
  - WindowEvent
- … and their listeners
  - ActionListener
  - AdjustmentListener
  - FocusListener
  - ItemListener
  - KeyListener
  - MouseListener
  - MouseWheelListener
  - WindowListener
  - WindowFocusListener
  - WindowStateListener
- All events and interfaces are within the java.awt.event package.

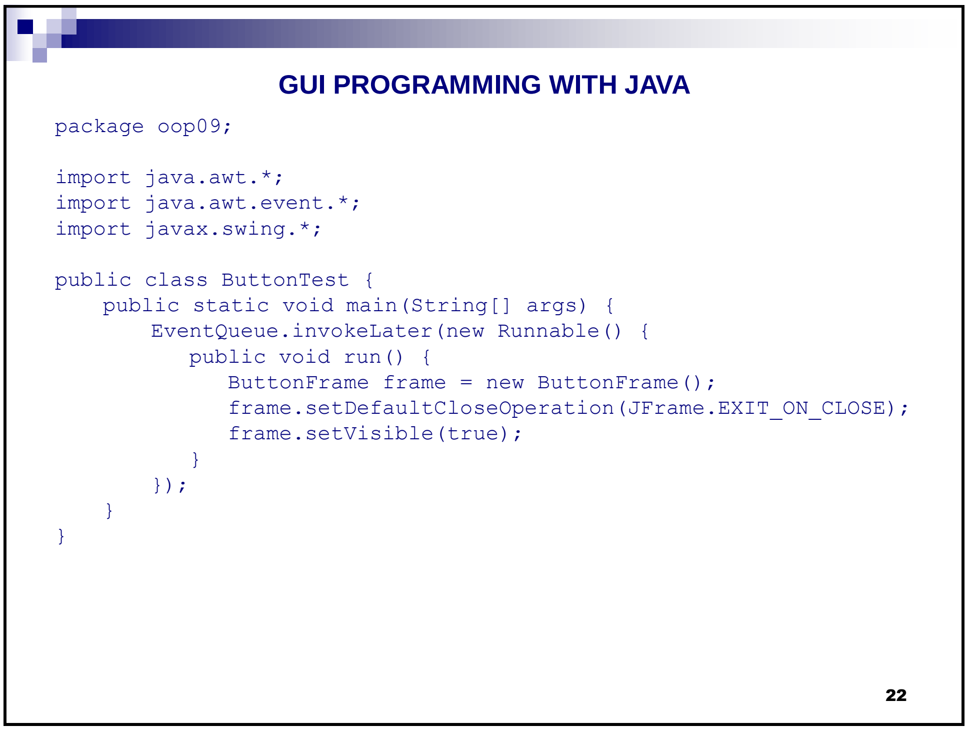
20



GUI PROGRAMMING WITH JAVA

- A working example of handling GUI events:
  - Event source: A button object of type javax.swing.JButton
  - Event type: Button click
  - Event listener interface: ActionListener
  - Event class: ActionEvent
  - Event handling method: **`void actionPerformed(ActionEvent e)`**
  - What to do: Create a window with three buttons, each having a color name as their caption. Change the background of the panel to appropriate color when a button is pressed.
  - Code: oop09.ButtonTest

21



GUI PROGRAMMING WITH JAVA

```java
package oop09;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ButtonTest {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
           public void run() {
              ButtonFrame frame = new ButtonFrame();
              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
              frame.setVisible(true);
           }
        });
    }
}
```

22

GUI PROGRAMMING WITH JAVA

```
@SuppressWarnings("serial")
class ButtonFrame extends JFrame {
    private JPanel buttonPanel;
    public static final int DEFAULT_WIDTH = 300;
    public static final int DEFAULT_HEIGHT = 200;

    public ButtonFrame() {
        setTitle("ButtonTest");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

        // create buttons
        JButton yellowButton = new JButton("Yellow");
        JButton blueButton = new JButton("Blue");
        JButton redButton = new JButton("Red");

        buttonPanel = new JPanel();

        // add buttons to panel
        buttonPanel.add(yellowButton);
        buttonPanel.add(blueButton);
        buttonPanel.add(redButton);

        //continues in the next slide
```

GUI PROGRAMMING WITH JAVA

```
        // add panel to frame
        add(buttonPanel);

        // create button actions
        ColorAction yellowAction = new ColorAction(Color.YELLOW);
        ColorAction blueAction = new ColorAction(Color.BLUE);
        ColorAction redAction = new ColorAction(Color.RED);

        // associate actions with buttons
        yellowButton.addActionListener(yellowAction);
        blueButton.addActionListener(blueAction);
        redButton.addActionListener(redAction);
    } //end of ButtonFrame constructor.

    //An action listener that sets the panel's background color.
    private class ColorAction implements ActionListener {
        private Color backgroundColor;
        public ColorAction(Color c){ backgroundColor = c; }
        public void actionPerformed(ActionEvent event){
            buttonPanel.setBackground(backgroundColor);
        }
    }/* coded as an inner class. You can write an alternative
        version with regular classes (ButtonTestV2) */
}
```

GUI PROGRAMMING WITH JAVA

- A working example of handling Window events:
  - Code: oop09.WindowEventTest
  - The WindowListener and WindowStateListener interfaces has been implemented

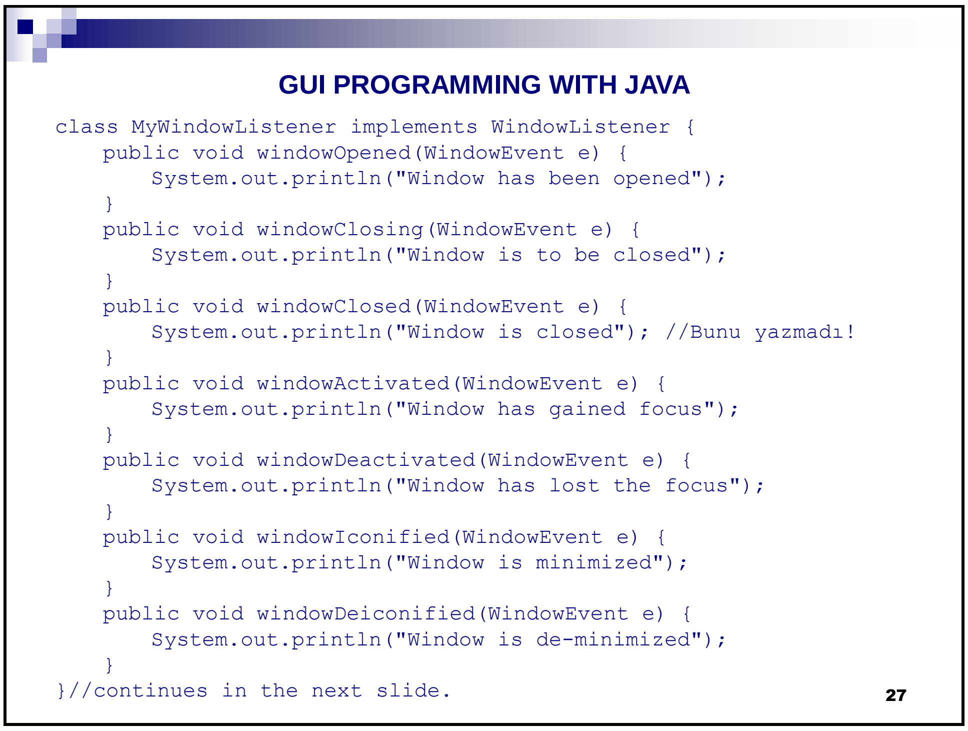
```
package oop09;
import java.awt.EventQueue;
import java.awt.event.*;
import javax.swing.*;

public class WindowEventTest {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                CaptureWindowEventsFrame frame =
                new CaptureWindowEventsFrame();
                frame.setVisible(true);
            }
        });
    }
}
//continues in the next slide.
```

25

GUI PROGRAMMING WITH JAVA

```
class CaptureWindowEventsFrame extends JFrame {
    private static final long serialVersionUID = 1L;
    public static final int DEFAULT_WIDTH = 300;
    public static final int DEFAULT_HEIGHT = 200;
    public CaptureWindowEventsFrame(){
        setTitle("CaptureWindowEventsTest");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        addWindowListener( new MyWindowListener( ) );
        addWindowStateListener( new MyWindowStateListener( ) );
    }
}
//continues in the next slide.
```

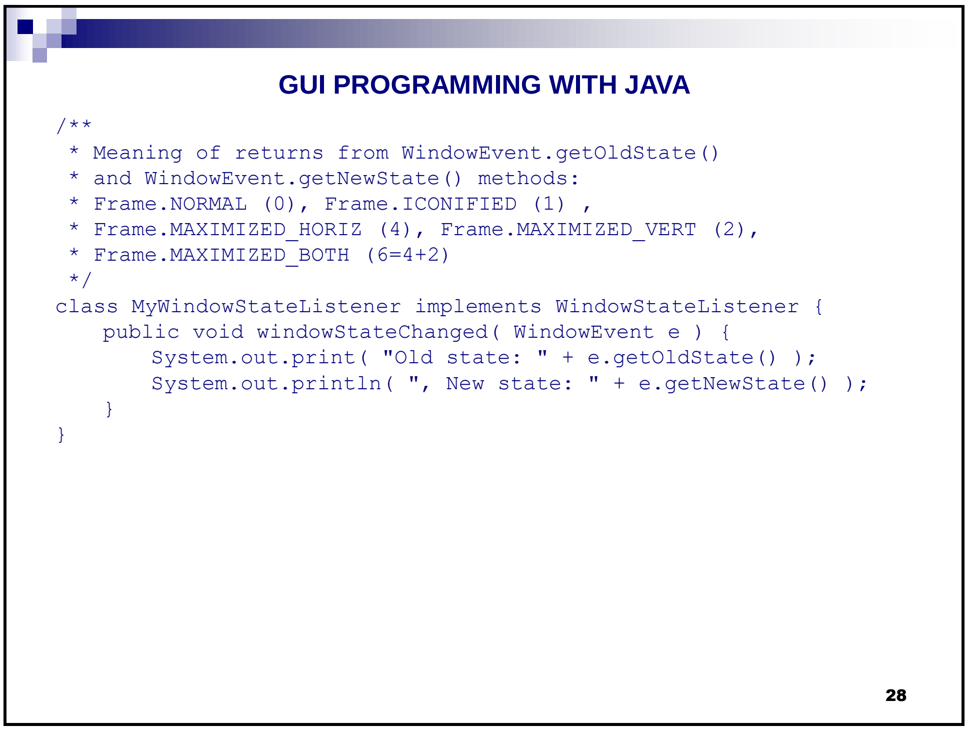
26



GUI PROGRAMMING WITH JAVA

```
class MyWindowListener implements WindowListener {
    public void windowOpened(WindowEvent e) {
        System.out.println("Window has been opened");
    }
    public void windowClosing(WindowEvent e) {
        System.out.println("Window is to be closed");
    }
    public void windowClosed(WindowEvent e) {
        System.out.println("Window is closed"); //Bunu yazmadı!
    }
    public void windowActivated(WindowEvent e) {
        System.out.println("Window has gained focus");
    }
    public void windowDeactivated(WindowEvent e) {
        System.out.println("Window has lost the focus");
    }
    public void windowIconified(WindowEvent e) {
        System.out.println("Window is minimized");
    }
    public void windowDeiconified(WindowEvent e) {
        System.out.println("Window is de-minimized");
    }
}//continues in the next slide.
```

27



GUI PROGRAMMING WITH JAVA

```
/**
 * Meaning of returns from WindowEvent.getOldState()
 * and WindowEvent.getNewState() methods:
 * Frame.NORMAL (0), Frame.ICONIFIED (1) ,
 * Frame.MAXIMIZED_HORIZ (4), Frame.MAXIMIZED_VERT (2),
 * Frame.MAXIMIZED_BOTH (6=4+2)
 */
class MyWindowStateListener implements WindowStateListener {
    public void windowStateChanged( WindowEvent e ) {
        System.out.print( "Old state: " + e.getOldState() );
        System.out.println( ", New state: " + e.getNewState() );
    }
}
```

28

GUI PROGRAMMING WITH JAVA

- About adapter classes:
  - Remember the `MyWindowListener` class implementing `WindowListener`.
  - It has 7 methods corresponding to 7 events.
  - What if you need to handle just 1 event?
  - In this case, create your listener class by inheriting from java.awt.event.WindowAdapter class.
    - This class has default implementations for all events.
    - You just override the method you need.
  - Check out this adapter:

```
package oop09;
import java.awt.event.*;
class MyWindowAdapter extends WindowAdapter {
    public void windowActivated(WindowEvent e) {
        System.out.println("Window has gained focus");
    }
    public void windowDeactivated(WindowEvent e) {
        System.out.println("Window has lost the focus");
    }
}
```

29

GUI PROGRAMMING WITH JAVA

- About adapter classes (cont'd):
  - Now we can create WindowEventTestV2 using the newly created adapter as a classroom exercise.
  - There are other adapters as well.
    - Those are left for you to explore.

30

GUI PROGRAMMING WITH JAVA

- Events are divided into two groups:
  - High-level (Logical) events:
    - These purpose of these events is to initiate a process related to business logic.
    - Clicking a button, resizing a window, advancing the scroll bar, etc.
  - Low-level events:
    - Keyboard and mouse events are of this type.
    - These come together and constitute high-level events.
    - For example, in order to click a button:
      - Move the mouse on top of the button.
      - Press the left mouse button.
      - Release the left mouse button.

GUI PROGRAMMING WITH JAVA

- Handling keyboard events:
  - Done via KeyListener interface or KeyAdapter class.
  - Right and left Shift/Control/Alt/AltGr keys are special keys.
    - You may need to control whether a key is pressed with a special key or not.
    - Pressing c is not the same as pressing CTRL-C
  - Example: oop09.KeyboardEventsTest

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class KeyboardEventsTest {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                CaptureKeyboardEventsFrame frame = new
                                CaptureKeyboardEventsFrame();
                frame.setVisible(true);
            }
        });
    }
}
```

GUI PROGRAMMING WITH JAVA

```
@SuppressWarnings("serial")
class CaptureKeyboardEventsFrame extends JFrame {
    public static final int DEFAULT_WIDTH = 300;
    public static final int DEFAULT_HEIGHT = 200;
    public CaptureKeyboardEventsFrame(){
        setTitle("CaptureKeyboardEventsTest");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        addKeyListener(new SpecialKeyHandler() );
        addKeyListener(new KeyHandler() );
    }
}
```

- Notice that we have associated listeners with the frame.
  - Listeners can also be associated with panels, but this is rather tricky and not covered here.

33

GUI PROGRAMMING WITH JAVA

```
class KeyHandler implements KeyListener {
    int lastKey; char lastChar; static String keyText;
    public void keyPressed(KeyEvent event) {
        int keyCode = event.getKeyCode();
        if( keyCode != lastKey) {
                System.out.print( "Key pressed, code: " + keyCode );
                lastKey = keyCode;
        }
    }
    public void keyReleased(KeyEvent event) {
        int keyCode = event.getKeyCode();
        System.out.println( "\nKey released, logical code: " +
        KeyEvent.getKeyText(keyCode) );
    }
    public void keyTyped(KeyEvent event) {
        char keyChar = event.getKeyChar();
        if( keyChar != lastChar ) {
                System.out.print( "\nKey typed: " + keyChar );
                lastChar = keyChar;
        }
        else
                System.out.print( keyChar );
    }
}
```

34

GUI PROGRAMMING WITH JAVA

```
class SpecialKeyHandler extends KeyAdapter {
    public void keyTyped(KeyEvent event) {
        if( event.isShiftDown() )
                System.out.print("***Shift key is used***");
        if( event.isControlDown() )
                System.out.print("***Control key is used***");
        if( event.isAltDown() )
                System.out.print("***Alt key is used***");
        if( event.isAltGraphDown() )
                System.out.print("***AltGR key is used***");
        if( event.isMetaDown() )
                System.out.print("***Meta key is used***");
    }
}
```

35

GUI PROGRAMMING WITH JAVA

- Handling mouse events:
  - Pressing the buttons of the mouse and moving the mouse are handled via different listeners.
  - Clicks are handled via MouseListener interface or MouseAdapter class.
  - Movements are handled via MouseMotionListener interface or MouseMotionAdapter class
  - Meanwhile, pressing of special keyboard keys can also be checked.
  - Example: oop09.MouseEventsTest

```
package oop09;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class MouseEventsTest {
  public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
       public void run() {
         CaptureMouseEventsFrame frame = new
                 CaptureMouseEventsFrame();
         frame.setVisible(true);
       }
    });
  }
}
```

36

```
@SuppressWarnings("serial")
class CaptureMouseEventsFrame extends JFrame {
    public static final int DEFAULT_WIDTH = 300;
    public static final int DEFAULT_HEIGHT = 200;
    public CaptureMouseEventsFrame(){
        setTitle("CaptureMouseEventsTest");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        addMouseListener( new MouseHandler() );
        addMouseMotionListener( new MouseMotionHandler() );
    }
}
class MouseMotionHandler implements MouseMotionListener {
    public void mouseDragged(MouseEvent event) {
        double x,y;
        x = event.getPoint().getX();
        y = event.getPoint().getY();
        System.out.println("A mouse drag at ("+x+","+y+")");
    }
    public void mouseMoved(MouseEvent event) {
        double x,y;
        x = event.getPoint().getX();
        y = event.getPoint().getY();
        System.out.println("A mouse move at ("+x+","+y+")");
    }
}
```

```
class MouseHandler extends MouseAdapter {
   public void mouseClicked(MouseEvent event) {
      double x = event.getPoint().getX(); double y = event.getPoint().getY();
      int z = event.getClickCount();
      System.out.println("A mouse click at ("+x+","+y+")x"+z);
      int m = event.getButton();
      if( ( m == MouseEvent.BUTTON1) )
         System.out.println("This was a left-click.");
      if( ( m == MouseEvent.BUTTON3) )
         System.out.println("This was a right-click.");
      if( ( m == MouseEvent.BUTTON2) )
         System.out.println("This was a middle-click.");
      int e = event.getModifiersEx();
      if( (e & InputEvent.SHIFT_DOWN_MASK) != 0 )
         System.out.println("The shift key was also pressed");
      if( (e & InputEvent.CTRL_DOWN_MASK) != 0 )
         System.out.println("The control key was also pressed");
      if( (e & InputEvent.ALT_DOWN_MASK) != 0 )
         System.out.println("The alt key was also pressed.");
      if( (e & InputEvent.ALT_GRAPH_DOWN_MASK) != 0 )
         System.out.println("The altgr key was also pressed.");
      if( (e & InputEvent.META_DOWN_MASK) != 0 )
         System.out.println("The meta key was also pressed.");
      System.out.println("Summary of mods in click: " +
         InputEvent.getModifiersExText(e));
    }
}
```

BLM 2562 OBJECT ORIENTED PROGRAMMING LECTURE NOTES

Assist. Prof. Dr. Yunus Emre SELÇUK

GUI PROGRAMMING WITH JAVA
Part III – Layout Management

GUI PROGRAMMING WITH JAVA

LAYOUT MANAGEMENT

- You can determine the positions of components in panel by using *layout managers*.
- The java.awt package includes various layout managers, each having different rules and therefore answering to different needs.
- We will examine some of those layout managers this week.
- General usage:
  - Assign a layout manager to a panel by using the Container.setLayout method
  - Add a GUI component to the panel by using the Container.add(aComponent, layoutRule) method.
  - By adding child panels into a parent and using different layout managers, you can fulfill complex layout requirements.

GUI PROGRAMMING WITH JAVA

FLOW LAYOUT:

- FlowLayout is the default layout manager
  - Adds components horizontally next to each other as long as there is enough space.
  - Advances to next line when no place is left
  - Example: oop10.Layout01FlowLayout
    - Same as the previous button example except the use of the layout manager.

41

GUI PROGRAMMING WITH JAVA

```
package oop10;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * FlowLayout manager lines the components horizontally until
 * there is no room and then starts a new row of components.
 */
public class Layout01FlowLayout {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
           public void run() {
                Frame01 frame = new Frame01();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
           }
        });
    }
}
```

42

GUI PROGRAMMING WITH JAVA

```
class Frame01 extends JFrame
{
    private static final long serialVersionUID = 1L;
    public static final int DEFAULT_WIDTH = 300;
    public static final int DEFAULT_HEIGHT = 200;

    public Frame01()
    {
        setTitle("ButtonTest");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        FlowLayoutPanel panel = new FlowLayoutPanel();
        add(panel);
    }
}
```

43

GUI PROGRAMMING WITH JAVA

```
class FlowLayoutPanel extends Jpanel {
    public FlowLayoutPanel() {
        //Bileşenleri eklemeden önce setLayout kullan.
        setLayout(new FlowLayout(FlowLayout.RIGHT));
        /*Alternatif değerler:
         * FlowLayout.LEFT
         * FlowLayout.CENTER (default),
         * FlowLayout.RIGHT
         *Alternatif kurucu:
         * FlowLayout(int align, int hgap, int vgap)
         * align: önceki değerler.
         * hgap: bileşenler arası yatay aralık.
         * vgap: dikey aralık.
         * Negatif aralıklar örtüşen bileşenlere sebep olur.
         * */

        JButton yellowButton = new JButton("Yellow");
        JButton blueButton = new JButton("Blue");
        JButton redButton = new JButton("Red");
```

44

GUI PROGRAMMING WITH JAVA

```
        JButton yellowButton = new JButton("Yellow");
        JButton blueButton = new JButton("Blue");
        JButton redButton = new JButton("Red");
        add(yellowButton);
        add(blueButton);
        add(redButton);

        ColorAction yellowAction = new ColorAction(Color.YELLOW);
        ColorAction blueAction = new ColorAction(Color.BLUE);
        ColorAction redAction = new ColorAction(Color.RED);

        yellowButton.addActionListener(yellowAction);
        blueButton.addActionListener(blueAction);
        redButton.addActionListener(redAction);
    }
```

45

GUI PROGRAMMING WITH JAVA

```
    private class ColorAction implements ActionListener
    {
        private Color backgroundColor;
        public ColorAction(Color c) {
                backgroundColor = c;
        }

        public void actionPerformed(ActionEvent event) {
                setBackground(backgroundColor);
        }
    }
}
```

46

GUI PROGRAMMING WITH JAVA

BORDER LAYOUT

- Introduces 5 areas as layout rules: CENTER, EAST, WEST, NORTH, SOUTH
- This layout manager enlarges the component so that it covers the entire area.
  - If we just need to position the component but not to enlarge it, we can put it in a panel with another layout manager and add that panel as a child of the first panel.
- Example: oop10.Layout02BorderLayout
  - This example uses an inner class while the previous example uses a regular class in the same java file. We could also use a regular class in its own java file. The result does not matter.
  - In order to show that the buttons work, the frame title is also changed in the button events.

GUI PROGRAMMING WITH JAVA

```
package oop10;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * NORTH, SOUTH: Yükseklik sabit, genişlik pencere genişliği.
 * EASTI, WEST: Yükseklik pencere yüksekliği, genişlik sabit.
 * CENTER: Diğer konumlardan geri kalan alanın tümü.
 */
    public class Layout02BorderLayout extends JFrame {
    private static final long serialVersionUID = 1L;
    public static final int DEFAULT_WIDTH = 300;
    public static final int DEFAULT_HEIGHT = 200;

    public Layout02BorderLayout() {
        setTitle("ButtonTest");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        BorderLayoutPanel panel = new BorderLayoutPanel();
        add(panel);
    }
```

GUI PROGRAMMING WITH JAVA

```
public static void main(String[] args)    {
   EventQueue.invokeLater(new Runnable() {
      public void run() {
        Layout02BorderLayout frame = new Layout02BorderLayout();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
      }
   });
}

class BorderLayoutPanel extends JPanel
{
    private static final long serialVersionUID = 1L;
    public BorderLayoutPanel()
    {
        setLayout(new BorderLayout());
        //veya BorderLayout( int hgap, int vgap )

        JButton yellowButton = new JButton("Yellow");
        JButton blueButton = new JButton("Blue");
        JButton redButton = new JButton("Red");
        JButton greenButton = new JButton("Green");
        JButton whiteButton = new JButton("White");
```

49

GUI PROGRAMMING WITH JAVA

```
        /* Renk değişikliği belli olsun istiyorsan
         * CENTER konumdakini remarkla.
         * Diğer konumlarda eleman
         * olmazsa o kısım hiç gösterilmiyor. */
        add(yellowButton, BorderLayout.WEST);
        add(blueButton, BorderLayout.CENTER);
        add(redButton, BorderLayout.EAST);
        add(greenButton, BorderLayout.NORTH);
        add(whiteButton, BorderLayout.SOUTH);

        ColorAction yellowAction = new ColorAction(Color.YELLOW);
        ColorAction blueAction = new ColorAction(Color.BLUE);
        ColorAction redAction = new ColorAction(Color.RED);
        ColorAction greenAction = new ColorAction(Color.GREEN);
        ColorAction whiteAction = new ColorAction(Color.WHITE);

        yellowButton.addActionListener(yellowAction);
        blueButton.addActionListener(blueAction);
        redButton.addActionListener(redAction);
        greenButton.addActionListener(greenAction);
        whiteButton.addActionListener(whiteAction);
    }
```

50

GUI PROGRAMMING WITH JAVA

```
        private class ColorAction implements ActionListener
        {
            private Color backgroundColor;
            public ColorAction(Color c) {
                backgroundColor = c;
            }

            public void actionPerformed(ActionEvent event) {
                setBackground(backgroundColor);
                setTitle("ButtonTest: (" + backgroundColor.getRed()
                + "," + backgroundColor.getGreen()
                + "," + backgroundColor.getBlue() + ")");
            }
        }
    }
}
```

51

GUI PROGRAMMING WITH JAVA

GRID LAYOUT

- This layout manager divides the area into rows and columns, i.e. cells, of equal size.
- Components are extended to cover their cells.
- As the window is resized, the cells are resized as well.
- You cannot target a special row and column when adding components.
  - Hint: You can leave a cell empty by adding an empty child panel into that cell.
- Example: oop10.Layout03GridLayout

52

GUI PROGRAMMING WITH JAVA

```
package oop10;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Alanı satır ve sütunlara böler. Bileşenler eşit boyutta olur,
 * pencere büyüdükçe o oranda bileşenler de büyür.
 * Ortada biryerleri boş bırakmak istiyorsan oraya boş bir panel
 * eklemelisin.
 */
public class Layout03GridLayout {
    public static void main(String[] args)    {
       EventQueue.invokeLater(new Runnable() {
          public void run() {
               Frame03 frame = new Frame03();
               frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
               frame.setVisible(true);
          }
       });
    }
}
```

53

GUI PROGRAMMING WITH JAVA

```
class Frame03 extends JFrame
{
    private static final long serialVersionUID = 1L;
    public static final int DEFAULT_WIDTH = 300;
    public static final int DEFAULT_HEIGHT = 200;

    public Frame03()
    {
        setTitle("ButtonTest");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        GridLayoutPanel panel = new GridLayoutPanel();
        add(panel);

        /* Pencere boyutunu içindeki bileşenlerin tercih
         * edilen veya varsayılan boyutlarını dikkate
         * alacak şekilde, layout manager kurallarına da
         * dikkat ederek yeniden boyutlandırır.
         * */
        pack();
    }
}
```

54

GUI PROGRAMMING WITH JAVA

```
class GridLayoutPanel extends JPanel
{
    private static final long serialVersionUID = 1L;
    public GridLayoutPanel()
    {
        setLayout(new GridLayout(2,2));
        /* GridLayout( int satir, int sutun );
         * GridLayout( int row, int col, int hgap, int vgap );
         * */

        JButton yellowButton = new JButton("Yellow");
        JButton blueButton = new JButton("Blue");
        JButton redButton = new JButton("Red");
        JButton greenButton = new JButton("Green");
        JButton whiteButton = new JButton("White");

        add(yellowButton);
        add(blueButton);
        add(redButton);
        add(greenButton);
        add(whiteButton);
```

55

GUI PROGRAMMING WITH JAVA

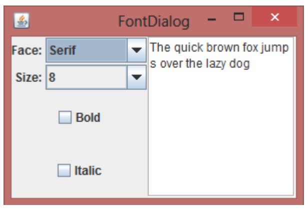
```
        ColorAction yellowAction = new ColorAction(Color.YELLOW);
        ColorAction blueAction = new ColorAction(Color.BLUE);
        ColorAction redAction = new ColorAction(Color.RED);
        ColorAction greenAction = new ColorAction(Color.GREEN);
        ColorAction whiteAction = new ColorAction(Color.WHITE);

        yellowButton.addActionListener(yellowAction);
        blueButton.addActionListener(blueAction);
        redButton.addActionListener(redAction);
        greenButton.addActionListener(greenAction);
        whiteButton.addActionListener(whiteAction);
    }
    private class ColorAction implements ActionListener {
        private Color backgroundColor;
        public ColorAction(Color c) {
                backgroundColor = c;
        }
        public void actionPerformed(ActionEvent event) {
                setBackground(backgroundColor);
        }
    }
}
```

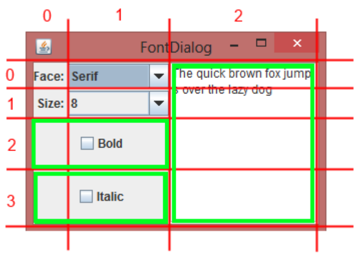
56

GUI PROGRAMMING WITH JAVA

GRIDBAG LAYOUT

- Using this layout manager is similar to designing HTML pages by using tables.
- Design the GUI on paper first, then divide it into rows and columns.
- Large components can extend multiple rows and/or columns.
- You can define constraints for individual cells:
  - To enlarge or not to enlarge the component it contains.
  - Alignment of the component within the cell (anchor)
  - Padding value





- Red: Cell borders
- **Green: Merged cells**

GUI PROGRAMMING WITH JAVA

GRIDBAG LAYOUT

- Best practice:
  - GridBagLayout object is not used directly as its cumbersome to do so.
  - We write a convenience class which models the constraints mentioned before and add the components with instances of that class.
  - Our convenience class is named GBC.
- Example: oop10.Layout04GridBagLayout
  - Focus on layout at the moment, although you can see a preview of using some GUI components.

GUI PROGRAMMING WITH JAVA

```
package oop10;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * GridBagConstraints için daha kolay kullanım sunan bir yardımcı sınıf yazılabilir.
 * Paketteki GBC sınıfı da böyle bir sınıftır.
 * GBC.gridx, gridy:
 * Bileşenin sol üst köşesi için ızgara koordinatları.
 * Izgaranın sol üstü (0,0), analitik sistemde 4. bölge.
 * GBC.gridwidth, gridheight:
 * Bileşenin kaç sütun ve kaç satır kaplayacağı.
 * HTML colspan ve rowspan gibi.
 * GBC.weightx, weighty:
 * Kalan boş alanın yüzde kaçının o sütun/satırın o yöndeki uzamasına atanacağı.
 *  Eğer alan hiç büyümesin istiyorsan o yöndekine 0 ver. Aksi halde 100 ver.
 *  Ara değerler pek deneme-yanılma için harcanan zamana değmiyor.
 * ÖNERİ: Her alana 100 ver, sonra çeşitli pencere ebatlarında büyümenin rahatsız
 *  ettiği bileşen ve yönler için 0 ver.
 *  Dikkat: Alan diyoruz, yani sütun veya satır. Bileşen değil.
 */
```

59

GUI PROGRAMMING WITH JAVA

```
/**
* GBC.fill:
*  Bileşenin büyüyüp tüm alanı kapsamasını istemiyorsan kullan.
*  Bu kısıtlama büyümeye izin verilen yönleri belirtir.
*  Değerler: GBC.NONE, GBC.HORIZONTAL, GBC.VERTICAL, GBC.BOTH
* GBC.anchor:
*  GBC.fill ayarları nedeniyle büyümeyen bileşenin o alanda
*  hangi konumda durmasını istiyorsan orayı ver.
*  Değerler: GBC.CENTER (default), GBC.NORTHEAST, ... yani tüm ana ve ara yönler.
*  1. gösterim:
*      NORTHWEST                 NORTH            NORTHEAST
*      WEST                      CENTER           EAST
*      SOUTHWEST                 SOUTH            SOUTHEAST
*  2. gösterim:
*      FIRST_LINE_START          LINE_START       FIRST_LINE_END
*      PAGE_START                CENTER           PAGE_END
*      LAST_LINE_START           LINE_END         LAST_LINE_END
* Yastıklama (dış):
*  Bileşenin etrafında ek boş alan olsun istiyorsan GBC.insets üyesini oluştur.
*  java.awt.Insets.Insets( int top, int left, int bottom, int right );
* GBC.ipadx, ipady:
*  İç yastıklama için. Bileşenin minimum boyutları senin için çok küçükse bu
*  değerleri ver, bileşenin min. boyutu eski min. boyut + yastık değeri olur.
*/
```

60

GUI PROGRAMMING WITH JAVA

```
public class Layout04GridBagLayout {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                 Frame06 frame = new Frame06();
                 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                 frame.setVisible(true);
             }
        });
    }
}

/**
 A frame that uses a grid bag layout to arrange font selection components.
 */
class Frame06 extends JFrame
{
    private static final long serialVersionUID = 1L;
    public static final int DEFAULT_WIDTH = 300;
    public static final int DEFAULT_HEIGHT = 200;
    private JComboBox<String> face;
    private JComboBox<Integer> size;
    private JCheckBox bold;
    private JCheckBox italic;
    private JTextArea sample;
```

GUI PROGRAMMING WITH JAVA

```
    public Frame06()
    {
        setTitle("FontDialog");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

        GridBagLayout layout = new GridBagLayout();
        setLayout(layout);

        ActionListener listener = new FontAction();

        // construct components
        JLabel faceLabel = new JLabel("Face: ");
        face = new JComboBox<String>(new String[] { "Serif", "SansSerif",
                "Monospaced", "Dialog", "DialogInput"  });
        face.addActionListener(listener);

        JLabel sizeLabel = new JLabel("Size: ");
        size = new JComboBox<Integer>(new Integer[] {
                8, 10, 12, 15, 18, 24, 36, 48 });
        size.addActionListener(listener);
        bold = new JCheckBox("Bold");
        bold.addActionListener(listener);
        italic = new JCheckBox("Italic");
        italic.addActionListener(listener);
```

GUI PROGRAMMING WITH JAVA

```
        sample = new JTextArea();
        sample.setText("The quick brown fox jumps over the lazy dog");
        sample.setEditable(false);
        sample.setLineWrap(true);
        sample.setBorder(BorderFactory.createEtchedBorder());

        // add components to grid, using GBC convenience class
        add(faceLabel, new GBC(0, 0).setAnchor(GBC.EAST));
        add(face, new GBC(1, 0).setFill(GBC.HORIZONTAL).
                setWeight(100, 0).setInsets(1));
        add(sizeLabel, new GBC(0, 1).setAnchor(GBC.EAST));
        add(size, new GBC(1, 1).setFill(GBC.HORIZONTAL).
                setWeight(100, 0).setInsets(1));
        add(bold, new GBC(0, 2, 2, 1).setAnchor(GBC.CENTER).
                setWeight(100, 100));
        add(italic, new GBC(0, 3, 2, 1).setAnchor(GBC.CENTER).
                setWeight(100, 100));
        add(sample, new GBC(2, 0, 1, 4).setFill(GBC.BOTH).
                setWeight(100, 100));
    }
```

63

GUI PROGRAMMING WITH JAVA

```
    /**
     An action listener that changes the font of the
     sample text.
     */
    private class FontAction implements ActionListener
    {
        public void actionPerformed(ActionEvent event)
        {
                String fontFace = face.getItemAt( face.getSelectedIndex() );
                int fontStyle = (bold.isSelected() ? Font.BOLD : 0)
                        + (italic.isSelected() ? Font.ITALIC : 0);
                int fontSize = size.getItemAt( size.getSelectedIndex() );
                Font font = new Font(fontFace, fontStyle, fontSize);
                sample.setFont(font);
                sample.repaint();
        }
    }
}
```

64

GUI PROGRAMMING WITH JAVA

```
package oop10;

/*
GBC - A convenience class to tame the GridBagLayout
Copyright (C) 2002 Cay S. Horstmann (http://horstmann.com)
This program is free software; in terms of the GNU license.*/

import java.awt.*;

/**
This class simplifies the use of the GridBagConstraints
class.
*/
public class GBC extends GridBagConstraints {
    private static final long serialVersionUID = 1L;

    /**
     Constructs a GBC with a given gridx and gridy position and
     all other grid bag constraint values set to the default.
     @param gridx the gridx position
     @param gridy the gridy position
     */
    public GBC(int gridx, int gridy)   {
        this.gridx = gridx;
        this.gridy = gridy;
    }
```

65

GUI PROGRAMMING WITH JAVA

```
    /**
     Constructs a GBC with given gridx, gridy, gridwidth, gridheight
     and all other grid bag constraint values set to the default.
     @param gridx the gridx position
     @param gridy the gridy position
     @param gridwidth the cell span in x-direction
     @param gridheight the cell span in y-direction
     */
    public GBC(int gridx, int gridy, int gridwidth, int gridheight)   {
        this.gridx = gridx;
        this.gridy = gridy;
        this.gridwidth = gridwidth;
        this.gridheight = gridheight;
    }
    /**
     Sets the anchor.
     @param anchor the anchor value
     @return this object for further modification
     */
    public GBC setAnchor(int anchor)   {
        this.anchor = anchor;
        return this;
    }
```

66

GUI PROGRAMMING WITH JAVA

```
/**
 Sets the fill direction.
 @param fill the fill direction
 @return this object for further modification
 */
public GBC setFill(int fill)   {
    this.fill = fill;
    return this;
}

/**
 Sets the cell weights.
 @param weightx the cell weight in x-direction
 @param weighty the cell weight in y-direction
 @return this object for further modification
 */
public GBC setWeight(double weightx, double weighty)   {
    this.weightx = weightx;
    this.weighty = weighty;
    return this;
}
```

67

GUI PROGRAMMING WITH JAVA

```
/**
 Sets the insets of this cell.
 @param distance the spacing to use in all directions
 @return this object for further modification
 */
public GBC setInsets(int distance)   {
    this.insets = new Insets(distance, distance, distance, distance);
    return this;
}

/**
 Sets the insets of this cell.
 @param top the spacing to use on top
 @param left the spacing to use to the left
 @param bottom the spacing to use on the bottom
 @param right the spacing to use to the right
 @return this object for further modification
 */
public GBC setInsets(int top, int left, int bottom, int right)   {
    this.insets = new Insets(top, left, bottom, right);
    return this;
}
```

68

GUI PROGRAMMING WITH JAVA

```
    /**
     Sets the internal padding
     @param ipadx the internal padding in x-direction
     @param ipady the internal padding in y-direction
     @return this object for further modification
     */
    public GBC setIpad(int ipadx, int ipady)   {
        this.ipadx = ipadx;
        this.ipady = ipady;
        return this;
    }
}
```

69

BLM 2562 OBJECT ORIENTED PROGRAMMING LECTURE NOTES
Assist. Prof. Dr. Yunus Emre SELÇUK

GUI PROGRAMMING WITH JAVA
Part IV – Fundamental GUI Components

70

GUI PROGRAMMING WITH JAVA

LABELS AND TEXTBOXES

- The **javax.swing.JLabel** instances are used for labels
  - Constructors:
    - JLabel(String text);
    - JLabel(Icon icon);
    - JLabel(String text, int align);
      - align: JLabel.LEFT, JLabel.RIGHT, JLabel.CENTER
  - Some useful methods:
    - setFont(Font);
    - setText(String);
    - String getText( );

GUI PROGRAMMING WITH JAVA

LABELS AND TEXTBOXES

- The **javax.swing.JTextField** instances are used for one-lined texts
  - Constructors:
    - JTextField(String text);
    - JTextField(int columnCount); //not precise
    - JTextField(String text, int columnCount);
  - Some useful methods :
    - setFont(Font);
    - setText(String);
    - String getText( );
    - setEditable(boolean);
    - setEnabled(boolean);
- javax.swing.JTextField and JLabel inherit many useful methods from javax.swing.text.JTextComponent.
- JLabel and JTextField example: oop11.UIElements01_TextField_Label

GUI PROGRAMMING WITH JAVA

LABELS AND TEXTBOXES

```
package oop11;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class UIElements01_TextField_Label extends JFrame {
  private static final long serialVersionUID = 1L;
  public static final int DEFAULT_WIDTH = 400;
  public static final int DEFAULT_HEIGHT = 200;

  public static void main(String[] args) {
      EventQueue.invokeLater(new Runnable() {
          public void run() {
               UIElements01_TextField_Label frame = new
                      UIElements01_TextField_Label();
               frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
               frame.setVisible(true);
          }
      });
  }
```

73

GUI PROGRAMMING WITH JAVA

LABELS AND TEXTBOXES

```
  public UIElements01_TextField_Label( ) {
    setTitle("TextField_Label_Test");
    setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    Dimension screenSize =
        Toolkit.getDefaultToolkit().getScreenSize();
    setLocation(screenSize.width/10,screenSize.height/10);
    AnaPanel panel = new AnaPanel();
    add(panel);
  }
  //Panel class is coded as an inner class
  class AnaPanel extends JPanel {
    private static final long serialVersionUID = 1L;
    private JTextField TFsol;
    private JTextField TFsag;
    private JLabel Lsol;
    private JLabel Lsag;
    private JButton Bswap;
```

74

GUI PROGRAMMING WITH JAVA

LABELS AND TEXTBOXES

```
//inner panel class continues
  public AnaPanel() {
      //JTextField( String text, int cols )
      //JTextField( int cols )
      TFsol = new JTextField("Sol",10);
      TFsag = new JTextField(10);
      // JLabel( String text );
      // JLabel( String text, int align );
      Lsol = new JLabel("Sol:");
      Lsag = new JLabel("Sağ:",JLabel.RIGHT);
      Lsol.setFont( new Font("Serif",Font.BOLD,12) );
      Lsag.setFont( new Font("Monospaced",Font.ITALIC,12) );
      Bswap = new JButton("Değiştir!");
      //JTextComponent.setEditable( boolean isEditable );
      TFsag.setEditable(false);
      add(Lsol); add(TFsol); add(Bswap); add(Lsag); add(TFsag);
      SwapAction action = new SwapAction();
      Bswap.addActionListener(action);
  }
```

75

GUI PROGRAMMING WITH JAVA

LABELS AND TEXTBOXES

```
  //inner panel class continues
    //action class is inner class of the panel class
    private class SwapAction implements ActionListener {
      public void actionPerformed(ActionEvent event) {
         String StrTmp;
         StrTmp = TFsol.getText();
         TFsol.setText( TFsag.getText() );
         TFsag.setText(StrTmp);
      }
    }
  }
}
```

76

GUI PROGRAMMING WITH JAVA

FORMATTED TEXTBOXES

- You can guarantee the contents of a textbox to be of a desired type by using **javax.swing.JFormattedTextField** instances, which collaborates with **java.text.NumberFormat** instances.
- You can refer to Core Java Vol.I or other sources for details.

TEXTBOXES FOR PASSWORDS

- Passwords must not be openly displayed.
- **javax.swing. JPasswordField** instances hide their contents by displaying * instead of its characters.
- Contents of a JPasswordField can be obtained by using char arrays only.
  - The reason for not using Strings is security: Strings reside in the memory until the garbage collector executes.
- Code snippet (refer to Core Java Vol.I or other sources for details) :

```
char[] buf; String parola;           //... When you are done:
JPasswordField PFcevap;              Arrays.fill(buf,'X');
buf = PFcevap.getPassword( );        parola = new String(buf);
parola = new String(buf);            //(for security reasons)
//use the password ...
```

77

GUI PROGRAMMING WITH JAVA

TEXT AREAS AND SCROLL BARS

- **javax.swing.JTextArea** instances are used for multiple-lined text.
- In order to add scroll bar(s), we "decorate" a JTextArea instance with a **javax.swing.JScrollPane** instance.
  - Examine the decorator design pattern in order to understand the rationale of this approach.
- Contents of the text area can be wrapped by words, letters or not wrapped at all.
- Example: oop11.UIElements02_TextArea

```
package oop11;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class UIElements02_TextArea extends JFrame {
   private static final long serialVersionUID = 1L;
   public static final int DEFAULT_WIDTH = 400;
   public static final int DEFAULT_HEIGHT = 200;
   private JTextArea textArea;
   private JScrollPane scrollPane;
   private JPanel buttonPanel;
   private JButton wrapButton, wrapModeButton;
```

78

GUI PROGRAMMING WITH JAVA

TEXT AREAS AND SCROLL BARS

```
public static void main(String[] args) {
   EventQueue.invokeLater(new Runnable() {
     public void run() {
       UIElements02_TextArea frame = new UIElements02_TextArea();
          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          frame.setVisible(true);
     }
   });
}
public UIElements02_TextArea() {
   setTitle("TextArea Test");
   setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
   Dimension screenSize =
            Toolkit.getDefaultToolkit().getScreenSize();
   setLocation(screenSize.width/10,screenSize.height/10);
   buttonPanel = new JPanel();
//constructor continues in the next slide
```

79

TEXT AREAS AND SCROLL BARS

```
//constructor is continued from the next slide
    //add a button to append text into the text area
    JButton insertButton = new JButton("Insert");
    buttonPanel.add(insertButton);
    insertButton.addActionListener( new ActionListener() {
       public void actionPerformed(ActionEvent event) {
         textArea.append("Bu yoğurdu sarımsaklasak da mı "
            + " saklasak, sarımsaklamasak da mı saklasak? ");
       }
    });
    // add button to turn line wrapping on and off
    wrapButton = new JButton("Wrap");
    buttonPanel.add(wrapButton);
    wrapButton.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent event) {
         boolean wrap = !textArea.getLineWrap();
         /* Sözcük kaydırmayı açar veya kapatır.*/
         textArea.setLineWrap(wrap);
         scrollPane.revalidate();
         wrapButton.setText(wrap ? "No Wrap" : "Wrap");
      }
    });
//constructor continues in the next slide
```

80

GUI PROGRAMMING WITH JAVA

TEXT AREAS AND SCROLL BARS

```
//constructor is continued from the next slide
    // add button to alter line wrapping style
    wrapModeButton = new JButton("Wrap by words");
    buttonPanel.add(wrapModeButton);
    wrapModeButton.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent event) {
          boolean wrap = !textArea.getWrapStyleWord();
          /* Kaydırma için kelime sınırlarının mı (true)
          * yoksa pencere sınırlarının mı (false)
          * kullanılacağını belirler.*/
          textArea.setWrapStyleWord(wrap);
          scrollPane.revalidate();
          wrapModeButton.setText(!wrap ? "Wrap by words"
                     : "Wrap by characters");
       }
    });
    add(buttonPanel, BorderLayout.SOUTH);
//constructor continues in the next slide
```

81

```
//constructor is continued from the next slide
    // add a text area with scroll bars:
       textArea = new JTextArea("Bu deneme mesajıdır.", 8, 40);
    /* javax.swing.JTextArea
     * JTextArea( int rows, int cols )
     * JTextArea( String text, int rows, int cols )
     * text: İlk olarak gözükecek metin.
     * rows: tercih edilen satır sayısı.
     * cols: tercih edilen sütun sayısı. */

    scrollPane = new JScrollPane(textArea);
    /* javax.swing.JScrollPane
     * Bir bileşene kaydırma çubuğu eklemek için
     * o bileşeni bu şekilde JScrollPane içine eklersin. */

    /* İçinde bileşen barındıran kaydırma panelini de
     * eklemeyi unutma.*/
    add(scrollPane, BorderLayout.CENTER);

    /* JTextComponent metodları kalıtımla mevcut.
     * \n gibi karakterlerde sorun yok. */
    textArea.setText(textArea.getText()
            +"\nÖzel\tkarakterler\tdenemesi!\n");
 }//end of constructor
}//end of class
```

82

GUI PROGRAMMING WITH JAVA

CHECK BOXES

- **javax.swing.JCheckBox** instances are used for check boxes
- Constructor: JCheckBox(String label);
- Get the state of the checkbox: boolean isSelected( );
- Example: oop11.UIElements03_CheckBox

```
package oop11;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class UIElements03_CheckBox extends JFrame {
   private static final long serialVersionUID = 1L;
   public static final int DEFAULT_WIDTH = 400;
   public static final int DEFAULT_HEIGHT = 200;
   private JLabel label;
   private JCheckBox bold;
   private JCheckBox italic;
   private static final int FONTSIZE = 12;
```

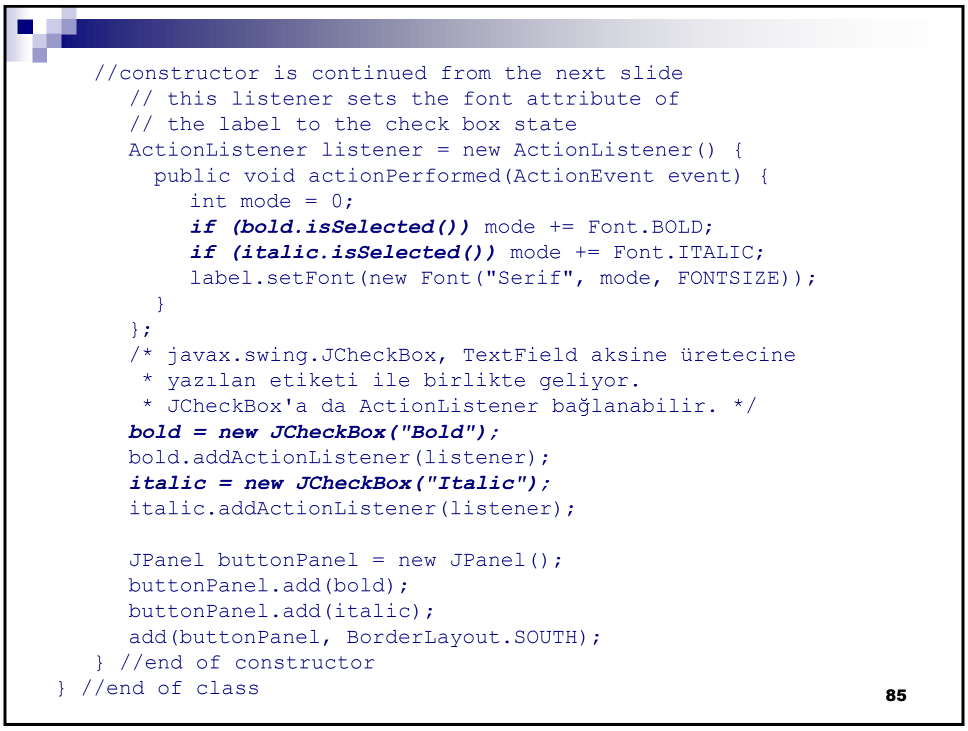
83

GUI PROGRAMMING WITH JAVA

CHECK BOXES

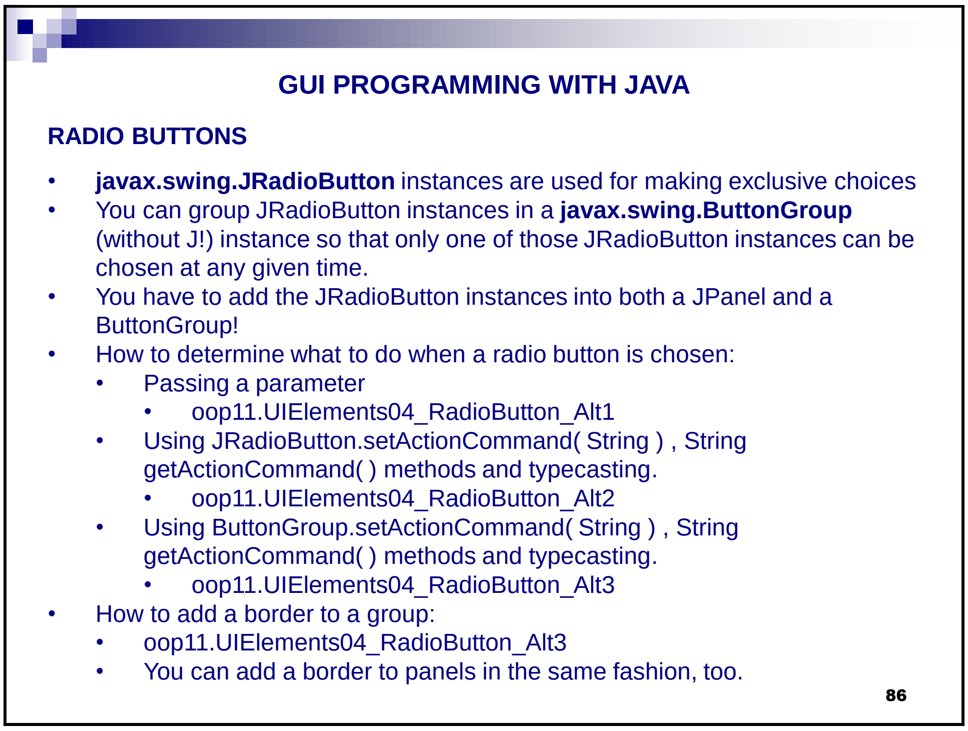
```
   public static void main(String[] args) {
      EventQueue.invokeLater(new Runnable() {
          public void run() {
          UIElements03_CheckBox frame = new UIElements03_CheckBox();
             frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             frame.setVisible(true);
          }
       });
   }
   public UIElements03_CheckBox() {
      setTitle("CheckBoxTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      Dimension screenSize =
               Toolkit.getDefaultToolkit().getScreenSize();
      setLocation(screenSize.width/10,screenSize.height/10);

      label = new JLabel("The quick brown fox jumps "
        + "over the lazy dog.");
      label.setFont(new Font("Serif", Font.PLAIN, FONTSIZE));
      add(label, BorderLayout.CENTER);
   //constructor continues in the next slide
```

84



```
   //constructor is continued from the next slide
      // this listener sets the font attribute of
      // the label to the check box state
      ActionListener listener = new ActionListener() {
        public void actionPerformed(ActionEvent event) {
           int mode = 0;
           if (bold.isSelected()) mode += Font.BOLD;
           if (italic.isSelected()) mode += Font.ITALIC;
           label.setFont(new Font("Serif", mode, FONTSIZE));
        }
      };
      /* javax.swing.JCheckBox, TextField aksine üretecine
       * yazılan etiketi ile birlikte geliyor.
       * JCheckBox'a da ActionListener bağlanabilir. */
      bold = new JCheckBox("Bold");
      bold.addActionListener(listener);
      italic = new JCheckBox("Italic");
      italic.addActionListener(listener);

      JPanel buttonPanel = new JPanel();
      buttonPanel.add(bold);
      buttonPanel.add(italic);
      add(buttonPanel, BorderLayout.SOUTH);
   } //end of constructor
} //end of class
```

85



GUI PROGRAMMING WITH JAVA

RADIO BUTTONS

- **javax.swing.JRadioButton** instances are used for making exclusive choices
- You can group JRadioButton instances in a **javax.swing.ButtonGroup** (without J!) instance so that only one of those JRadioButton instances can be chosen at any given time.
- You have to add the JRadioButton instances into both a JPanel and a ButtonGroup!
- How to determine what to do when a radio button is chosen:
  - Passing a parameter
    - oop11.UIElements04_RadioButton_Alt1
  - Using JRadioButton.setActionCommand(String) , String getActionCommand( ) methods and typecasting.
    - oop11.UIElements04_RadioButton_Alt2
  - Using ButtonGroup.setActionCommand(String) , String getActionCommand( ) methods and typecasting.
    - oop11.UIElements04_RadioButton_Alt3
- How to add a border to a group:
  - oop11.UIElements04_RadioButton_Alt3
  - You can add a border to panels in the same fashion, too.

86

RADIO BUTTONS (Alt.1)

```
package oop11;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

// Her düğmeye ayrı olay dinleyicisi atayan bir çözüm.
public class UIElements04_RadioButton_Alt1 extends JFrame {
   private static final long serialVersionUID = 1L;
   public static final int DEFAULT_WIDTH = 800;
   public static final int DEFAULT_HEIGHT = 200;
   private JPanel buttonPanel;
   private ButtonGroup group;
   private JLabel label;
   private static final int DEFAULT_SIZE = 12;
   public static void main(String[] args) {
      EventQueue.invokeLater(new Runnable() {
          public void run() {
             UIElements04_RadioButton_Alt1 frame = new
                       UIElements04_RadioButton_Alt1();
             frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             frame.setVisible(true);
          }
      });
   }
```

87

RADIO BUTTONS (Alt.1)

```
   public UIElements04_RadioButton_Alt1() {
      setTitle("RadioButtonTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      Dimension screenSize =
               Toolkit.getDefaultToolkit().getScreenSize();
      setLocation(screenSize.width/10,screenSize.height/10);
      label = new Jlabel
               ("The quick brown fox jumps over the lazy dog.");
      label.setFont(new Font("Serif", Font.PLAIN, DEFAULT_SIZE));
      add(label, BorderLayout.CENTER);

      // add the radio buttons
      buttonPanel = new JPanel();
      group = new ButtonGroup();
      /* Kendi metodumuz olan addRadioButton üç ana iş
       * yapacak: Düğmeyi gruba ekleme, düğmeyi panele
       * ekleme ve düğmeye ActionListener atama. */
      addRadioButton("Small", 8);
      addRadioButton("Medium", 12);
      addRadioButton("Large", 18);
      addRadioButton("Extra large", 36);
      add(buttonPanel, BorderLayout.SOUTH);
   }
```

88

RADIO BUTTONS (Alt.1)

```
   public void addRadioButton(String name, final int size){
      boolean selected = size == DEFAULT_SIZE;
      /* javax.swing.JRadioButton da üretecine
       * yazılan etiketi ile birlikte geliyor.
       * JRadioButton(String name, boolean isSelected)  */
      JRadioButton button = new JRadioButton(name, selected);
      // JRadioButton, grubuna ve paneline ayrı ayrı eklenmelidir.
      group.add(button);
      buttonPanel.add(button);
      button.setActionCommand(name);

      // this listener sets the label font size
      ActionListener listener = new ActionListener() {
         public void actionPerformed(ActionEvent event) {
           label.setFont(new Font("Serif", Font.PLAIN, size));
         }
      };
      button.addActionListener(listener);
   }
}
```

89

RADIO BUTTONS (Alt.2)

```
package oop11;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/**
 * Burada tüm düğmelere tek bir ActionListener örneği atayacağız.
 * Hangi düğmenin basıldığını tetiklenen olaydan öğrenip
 * seçilen değeri o düğmenin eylem komutundan öğreneceğiz.
 */
public class UIElements04_RadioButton_Alt2 extends JFrame {
   private static final long serialVersionUID = 1L;
   public static final int DEFAULT_WIDTH = 800;
   public static final int DEFAULT_HEIGHT = 200;
   private JPanel buttonPanel;
   private ButtonGroup group;
   private JLabel label;
   private static final int DEFAULT_SIZE = 12;
   private MyRadioButtonListener listener;

   @SuppressWarnings("unused")
   private MyRadioButton RBsmall, RBmed, RBlarge, RBxlarge;
```

90

RADIO BUTTONS (Alt.2)

```
public static void main(String[] args) {
   EventQueue.invokeLater(new Runnable() {
       public void run() {
          UIElements04_RadioButton_Alt2 frame = new
                   UIElements04_RadioButton_Alt2();
          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          frame.setVisible(true);
       }
    });
}
```

91

RADIO BUTTONS (Alt.2)

```
public UIElements04_RadioButton_Alt2() {
   setTitle("RadioButtonTestV2");
   setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
   Dimension screenSize =
     Toolkit.getDefaultToolkit().getScreenSize();
     setLocation(screenSize.width/10,screenSize.height/10);
     label = new Jlabel
            ("The quick brown fox jumps over the lazy dog.");
     label.setFont(new Font("Serif", Font.PLAIN, DEFAULT_SIZE));
     add(label, BorderLayout.CENTER);

   // add the radio buttons
   buttonPanel = new JPanel();
   group = new ButtonGroup();
   listener = new MyRadioButtonListener();

   RBsmall = new MyRadioButton("Small", 8);
   RBmed = new MyRadioButton("Medium", 12);
   RBlarge = new MyRadioButton("Large", 18);
   RBxlarge = new MyRadioButton("Extra large", 36);

   add(buttonPanel, BorderLayout.SOUTH);
}
```

92

RADIO BUTTONS (Alt.2)

```
/**
 * İç sınıf yapmasaydık size ve DEFAULT_SIZE bilgilerini
 * kurucuya parametre vermeliydik. */
private class MyRadioButton extends JRadioButton {
   private static final long serialVersionUID = 1L;

   public MyRadioButton( String name, int size ) {
      super(name, size == DEFAULT_SIZE);
      group.add(this);
      buttonPanel.add(this);
      addActionListener(listener);

      /* setActionCommand mutlaka çalıştırılmalı
       * çünkü radyo düğmesinin modelinde eylem
       * komutu null'dır. Halbuki JButton'da eylem
       * komutu düğmenin etiketi ile aynıydı. */
      setActionCommand(String.valueOf(size));
   }
}
```

93

RADIO BUTTONS (Alt.2)

```
   private class MyRadioButtonListener implements ActionListener {
      public void actionPerformed(ActionEvent event) {
         Object source = event.getSource();
         /* (JRadioButton)source).getActionCommand() ile
          * komut alınıyor. Gerisi String->int yapmak ve
          * font boyutunu atamaktır. */
         label.setFont(new Font("Serif", Font.PLAIN, Integer.valueOf
                (((JRadioButton)source).getActionCommand())));
      }
   }
}
```

94

RADIO BUTTONS (Alt.3)

```
package oop11;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.Border;

/**
 * Burada tüm düğmelere tek bir ActionListener örneği atayacağız.
 * Hangi düğmenin basıldığını düğmenin grubundan öğrenip
 * seçilen değeri o düğmenin eylem komutundan öğreneceğiz. */
public class UIElements04_RadioButton_Alt3 extends JFrame {
    private static final long serialVersionUID = 1L;
    public static final int DEFAULT_WIDTH = 800;
    public static final int DEFAULT_HEIGHT = 200;
    private JPanel buttonPanel;
    private ButtonGroup group;
    private JLabel label;
    private static final int DEFAULT_SIZE = 12;

    @SuppressWarnings("unused")
    private MyRadioButton RBsmall, RBmed, RBlarge, RBxlarge;
    private MyRadioButtonListener listener;
```

95

RADIO BUTTONS (Alt.3)

```
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                UIElements04_RadioButton_Alt3 frame = new
                        UIElements04_RadioButton_Alt3();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
            }
        });
    }
```

96

RADIO BUTTONS (Alt.3)

```
public UIElements04_RadioButton_Alt3() {
    setTitle("RadioButtonTestV3");
    setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    Dimension screenSize =
            Toolkit.getDefaultToolkit().getScreenSize();
    setLocation(screenSize.width/10,screenSize.height/10);
    label = new Jlabel
            ("The quick brown fox jumps over the lazy dog.");
    label.setFont(new Font("Serif", Font.PLAIN, DEFAULT_SIZE));
    add(label, BorderLayout.CENTER);
    // add the radio buttons
    buttonPanel = new JPanel(); group = new ButtonGroup();
    listener = new MyRadioButtonListener();
    RBsmall = new MyRadioButton("Small", 8);
    RBmed = new MyRadioButton("Medium", 12);
    RBlarge = new MyRadioButton("Large", 18);
    RBxlarge = new MyRadioButton("Extra large", 36);
    // setup a border using the decorator pattern.
    Border etched = BorderFactory.createEtchedBorder();
    Border titled = BorderFactory.createTitledBorder(
            etched, "Size options:");
    buttonPanel.setBorder(titled);
    add(buttonPanel, BorderLayout.SOUTH);
}
```

97

RADIO BUTTONS (Alt.3)

```
/* İç sınıf yapmasaydık size ve DEFAULT_SIZE bilgilerini
 * kurucuya parametre vermeliydik. */
private class MyRadioButton extends JRadioButton {
    private static final long serialVersionUID = 1L;
    public MyRadioButton( String name, int size ) {
       super(name, size == DEFAULT_SIZE);
       group.add(this);
       buttonPanel.add(this);
       addActionListener(listener);

       /* setActionCommand mutlaka çalıştırılmalı
        * çünkü radyo düğmesinin modelinde eylem
        * komutu null'dır. Halbuki JButton'da eylem
        * komutu düğmenin etiketi ile aynıydı. */
      setActionCommand(String.valueOf(size));
   }
}
private class MyRadioButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
      label.setFont(new Font("Serif", Font.PLAIN,
      Integer.valueOf(group.getSelection().getActionCommand())));
    }
}
}
```

98

GUI PROGRAMMING WITH JAVA

COMBO BOXES

- **javax.swing.JComboBox** instances are used for making an exclusive selection by taking up less space than radio button groups.
- There are two ways to add elements into a combo box:
    - The direct but slower way: Using the addItem method
        - Example: oop11.UIElements05_ComboBox_Alt1
    - The indirect but faster way: Using the **model** of a combo box
        - Example: oop11.UIElements05_ComboBox_Alt2

AN INTERLUDE: INTRODUCTION OF THE MVC DESIGN PATTERN

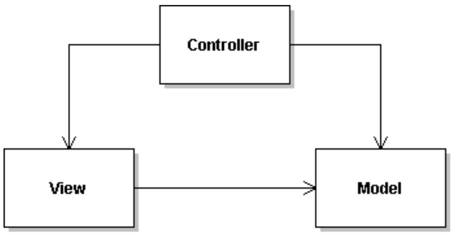
DESIGN PATTERNS:

- A **design pattern** is a solution to a general design problem.
- The benefits of studying design patterns:
    - You don't need to reinvent the wheel.
    - Leads you to a better design, so that you can come up with a more flexible and maintainable solution.
    - Gives a good way of transferring experience of veterans to newbies
    - Constructs a common dictionary among professionals.
- There is a formal way of documenting design pattern:
    - A pattern must have a descriptive and preferably short name
    - A pattern must first give a description of the problem it solves.
    - Then alternative solutions must be described in a way that presents their pros and cons relative to each other and according to different sub-cases.
    - Source code and UML schemas are needed for a good description.
    - Please refer to Gang of Four's classical book and other books which have additional, detailed examples.
        - Design Patterns – Elements of Reusable OO Software, Erich Gamma et.al (Gang of Four), Addison-Wesley, 1994

AN INTERLUDE: INTRODUCTION OF THE MVC DESIGN PATTERN

THE MODEL – VIEW – CONTROLLER DESIGN PATTERN:

- The problem:
  - We need to present the same data in different ways.
  - Meanwhile, we need to give the users the ability to choose which part of this data is to be presented and how.
- Solution: Model the data, how it is represented and how the representation is controlled in three kinds of components:
  - Model class: Represents the raw data.
  - View class: Represents how data is displayed.
  - Controller class: Handles commands of the user.



101

COMBO BOXES

- Example of adding items to combo boxes directly:

```
package oop11;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class UIElements05_ComboBox_Alt1 extends JFrame {
   private static final long serialVersionUID = 1L;
   public static final int DEFAULT_WIDTH = 400;
   public static final int DEFAULT_HEIGHT = 200;
   private JComboBox<String> faceCombo;
   private JLabel label;
   private static final int DEFAULT_SIZE = 12;

   public static void main(String[] args) {
      EventQueue.invokeLater(new Runnable() {
          public void run() {
             UIElements05_ComboBox_Alt1 frame =
                new UIElements05_ComboBox_Alt1();
             frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             frame.setVisible(true);
          }
       });
}
```

102

COMBO BOXES

```
public UIElements05_ComboBox_Alt1() {
   setTitle("ComboBoxTest");
   setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
   Dimension screenSize =
       Toolkit.getDefaultToolkit().getScreenSize();
   setLocation(screenSize.width/10,screenSize.height/10);
   label = new JLabel(
       "The quick brown fox jumps over the lazy dog.");
   label.setFont(new Font("Serif", Font.PLAIN, DEFAULT_SIZE));
   add(label, BorderLayout.CENTER);

   // make a combo box and add face names
   faceCombo = new JComboBox<String>();
   /* Alternative constructors:
    * JComboBox(E[] items)
    * JComboBox(Vector<E> items) */

   /* Let this JComboBox to be editable. */
   faceCombo.setEditable(true);
```

103

COMBO BOXES

```
      /* Add an item to the end of the list.
       * Use insertItemAt(<E> item, int index) for insertion
       * index=0 means adding to the top.
       * You can add/remove items even at runtime.
       * Use JComboBox.removeItem(<E> item) or
       * JComboBox.removeItemAt(int index) for removal. */
      faceCombo.addItem("Serif");
      faceCombo.addItem("SansSerif");
      faceCombo.addItem("Monospaced");
      faceCombo.addItem("Dialog");
      /* the combo box listener changes the label font
       * to the selected face name. */
      faceCombo.addActionListener(newActionListener() {
         public void actionPerformed(ActionEvent event) {
           label.setFont(new Font(
                faceCombo.getItemAt(faceCombo.getSelectedIndex()),
                        Font.PLAIN, DEFAULT_SIZE));
        }
      });
      JPanel comboPanel = new JPanel();
      comboPanel.add(faceCombo);
      add(comboPanel, BorderLayout.SOUTH);
   }
}
```

104

COMBO BOXES

- Example of adding items to a combo box indirectly via its model:
  - Changing the constructor as shown below is enough:

```
/* Adım 1: Boş JComboBox oluştur. */
faceCombo = new JComboBox<String>();
faceCombo.setEditable(true);
/* Adım 2: DefaultComboBoxModel oluştur. */
DefaultComboBoxModel<String> model = new
    DefaultComboBoxModel<String>();
/* Adım 3: Model'e elemanları ekle */
model.addElement("Serif");
model.addElement("SansSerif");
model.addElement("Monospaced");
model.addElement("Dialog");
/* Adım 4: ComboBox'a modeli ata. */
faceCombo.setModel(model);
```

105

GUI PROGRAMMING WITH JAVA

MENU OPERATIONS

- Menus are created by using three classes in Java:
  - javax.swing.JMenuBar instance represents the main menu bar
  - javax.swing.JMenu instances represent top level menu choices
  - javax.swing.JMenuItem sınıfı: instances represent menu items
- Moreover, menu items must be associated with some program code by using the addActionListener method.
  - You can create a menu item directly from an action object.
- Example:
  - oop11. UIElements06_Menu

```
package oop11;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class UIElements06_Menu extends JFrame {
   private static final long serialVersionUID = 1L;
   public static final int DEFAULT_WIDTH = 400;
   public static final int DEFAULT_HEIGHT = 200;
```

106

GUI PROGRAMMING WITH JAVA

MENU OPERATIONS

```
public static void main(String[] args) {
   EventQueue.invokeLater(new Runnable() {
      public void run() {
      UIElements06_Menu frame = new UIElements06_Menu();
          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          frame.setVisible(true);
        }
     });
}
public UIElements06_Menu() {
   setTitle("MenuTest");
   setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
   Dimension screenSize =
     Toolkit.getDefaultToolkit().getScreenSize();
   setLocation(screenSize.width/10,screenSize.height/10);
```

107

GUI PROGRAMMING WITH JAVA

MENU OPERATIONS

```
// Adım 1: Bir javax.swing.JMenuBar oluşturup Frame'e ata.
JMenuBar menuBar = new JMenuBar();
this.setJMenuBar(menuBar);

/* Adım 2: Her menü için bir javax.swing.JMenu
 * nesnesi oluştur. */
JMenu fileMenu = new JMenu("File");
JMenu editMenu = new JMenu("Edit");

/* Adım 3: Üst düzey menüleri JMenuBar'a ekle. */
menuBar.add(fileMenu);
menuBar.add(editMenu);

/* Adım 4: Üst düzey menülere menü seçenekleri
 * ekle: javax.swing.JMenuItem. */
JMenuItem newItem = new JMenuItem("New");
JMenuItem openItem = new JMenuItem("Open");
fileMenu.add(newItem);
fileMenu.add(openItem);
```

108

GUI PROGRAMMING WITH JAVA

MENU OPERATIONS

```
/* Adım 5: Ayraç da ekleyebilirsin. */
fileMenu.addSeparator();

/* Adım 6: Alt menüler de ekleyebilirsin. */
JMenu optionsMenu = new JMenu("Options");
fileMenu.add(optionsMenu);

JMenuItem readOnlyItem = new JMenuItem("Read Only");
optionsMenu.add(readOnlyItem);
fileMenu.addSeparator();

/* Adım 7: Gerekli eylemleri tanımla. Burada anonim iç
 * sınıf olarak tanımlandı. Eylemin adaptör sınıfı olan
 * javax.swing.AbstractAction'ı hatırla. Bu eylem sonra
 * menü seçeneği olacağı için kurucusuna etiket verdik */
Action exitAction = new AbstractAction("Exit") {
   private static final long serialVersionUID = 1L;
      public void actionPerformed(ActionEvent event) {
         System.exit(0);
      }
};
```

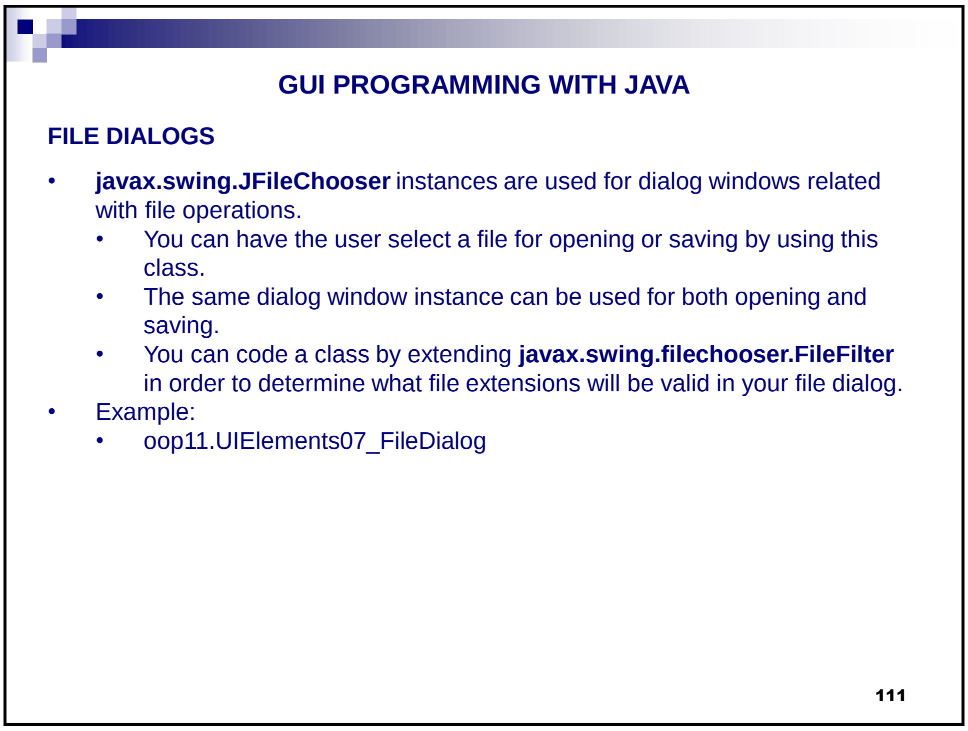
109

GUI PROGRAMMING WITH JAVA

MENU OPERATIONS

```
        /* Adım 8: Eylemleri ata.
         * Alternatif 8a: Eylemden hemen menü seçeneği yapmak */
        fileMenu.add(exitAction);
        /* Adım 8: Eylemleri ata.
         * Alternatif 8b: Mevcut menü seçeneğine eylem atamak */
        JMenuItem testItem = new JMenuItem("Test");
        testItem.addActionListener(new TestAction01("Test"));
        editMenu.add(testItem);

        /* File menüsünün eylemlerini de atayalım */
        newItem.addActionListener( new TestAction01("New") );
        openItem.addActionListener( new TestAction01("Open") );
        readOnlyItem.addActionListener(new TestAction01("Read Only"));
    }
}
class TestAction01 extends AbstractAction {
   private static final long serialVersionUID = 1L;
   public TestAction01(String name) { super(name); }
   public void actionPerformed(ActionEvent event) {
      System.out.println(getValue(Action.NAME) + " selected.");
   }
}
```

110



GUI PROGRAMMING WITH JAVA

FILE DIALOGS

- **javax.swing.JFileChooser** instances are used for dialog windows related with file operations.
  - You can have the user select a file for opening or saving by using this class.
  - The same dialog window instance can be used for both opening and saving.
  - You can code a class by extending **javax.swing.filechooser.FileFilter** in order to determine what file extensions will be valid in your file dialog.
- Example:
  - oop11.UIElements07_FileDialog

111