Object Oriented Development with Java

(CT038-3-2 and Version VC1)

A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Graphical User Interface (GUI)

GUI-based Development

Topic & Structure of the lesson

- AWT vs Swing
- Swing components
- Creating a window
- Adding components
- Handling action events

Learning outcomes

- At the end of this lecture you should be able to:
  - Differentiate between AWT and Swing
  - Create a Window
  - Understand GUI Classes in a main method
  - Radio Buttons and Check Boxes
  - Focus on Problem Solving: Extending Classes from Jpanel

Key terms you must be able to use

If you have mastered this topic, you should be able to use the following terms correctly in your assessments:

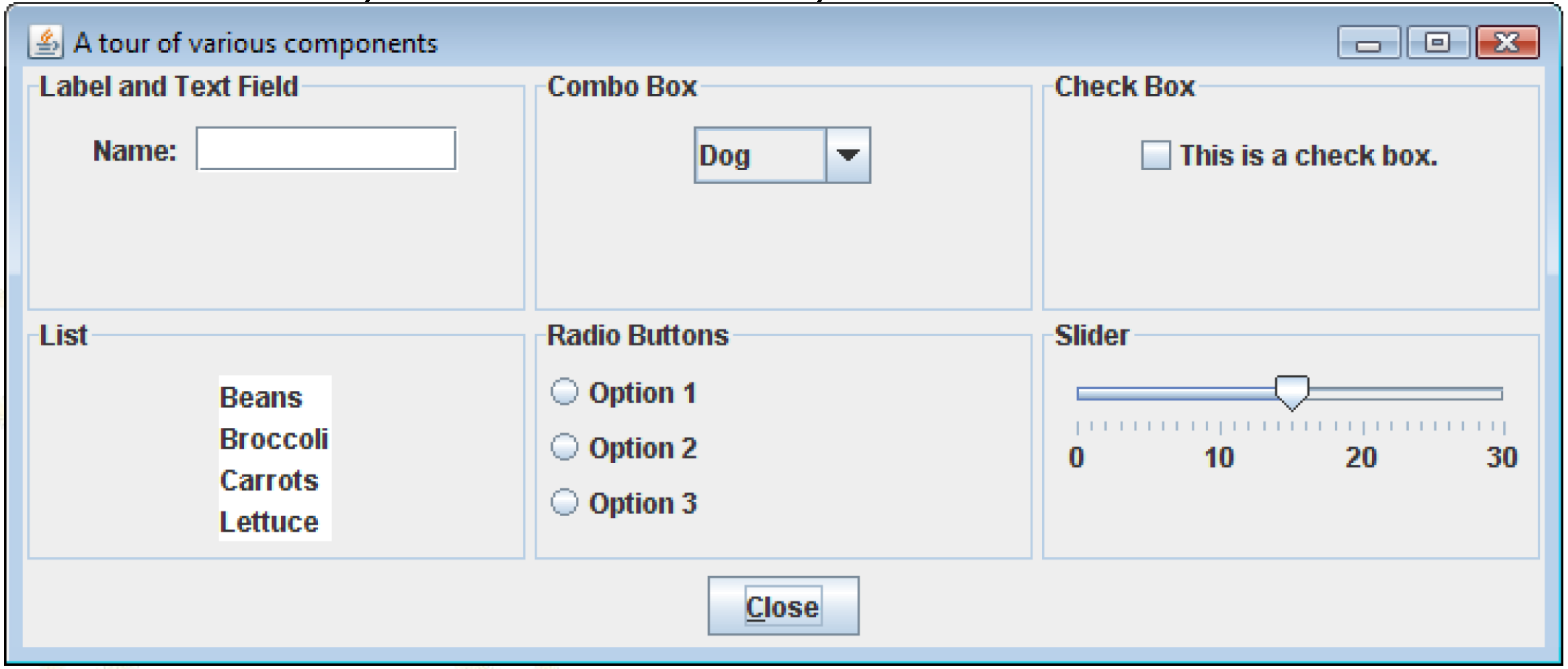
- AWT
- Swing

Introduction

- A GUI is a graphical window or windows that provide interaction with the user
- GUIs accept input from:
  - the keyboard
  - a mouse
- A window in a GUI consists of components that:
  - present data to the user
  - allow interaction with the application

Introduction

- Some common GUI components are:
  - buttons, labels, text fields, check boxes, radio buttons, combo boxes, and sliders.



Introduction

- Java is equipped with a set of classes for drawing graphics and creating graphical user interfaces
- These classes are part of the *Abstract Window Toolkit (AWT)*

Abstract Window Toolkit (AWT)

- The AWT allows creation of applications and applets with GUI components
- Java programs using the AWT:
  - look consistent with other applications on the same system
  - can offer only components that are common to all the operating systems that support Java

Swing

- *Swing* is a library of classes that provide an improved alternative for creating GUI applications and applets
- Part of Oracle's Java Foundation Classes (JCF)
- Developed to provide a more sophisticated set of GUI components than AWT
- Swing components have a consistent look and predictable behavior on any operating system
- Swing components can be easily extended

Event Driven programming

- Programs that operate in a GUI environment must be *event-driven*
- An *event* is an action that takes place within a program, such as the clicking of a button
- Part of writing a GUI application is creating event listeners
- An *event listener* is an object that automatically executes one of its methods when a specific event occurs

`javax.swing` and `java.awt`

- In an application that uses Swing classes, it is necessary to use the following statement:

```
import javax.swing.*;
```

- In an application that uses an AWT class, it is necessary to use the following statement.

```
import java.awt.*;
```

Swing Components

- Two classes: *Container* and *Component*
- Container holds components and/or containers

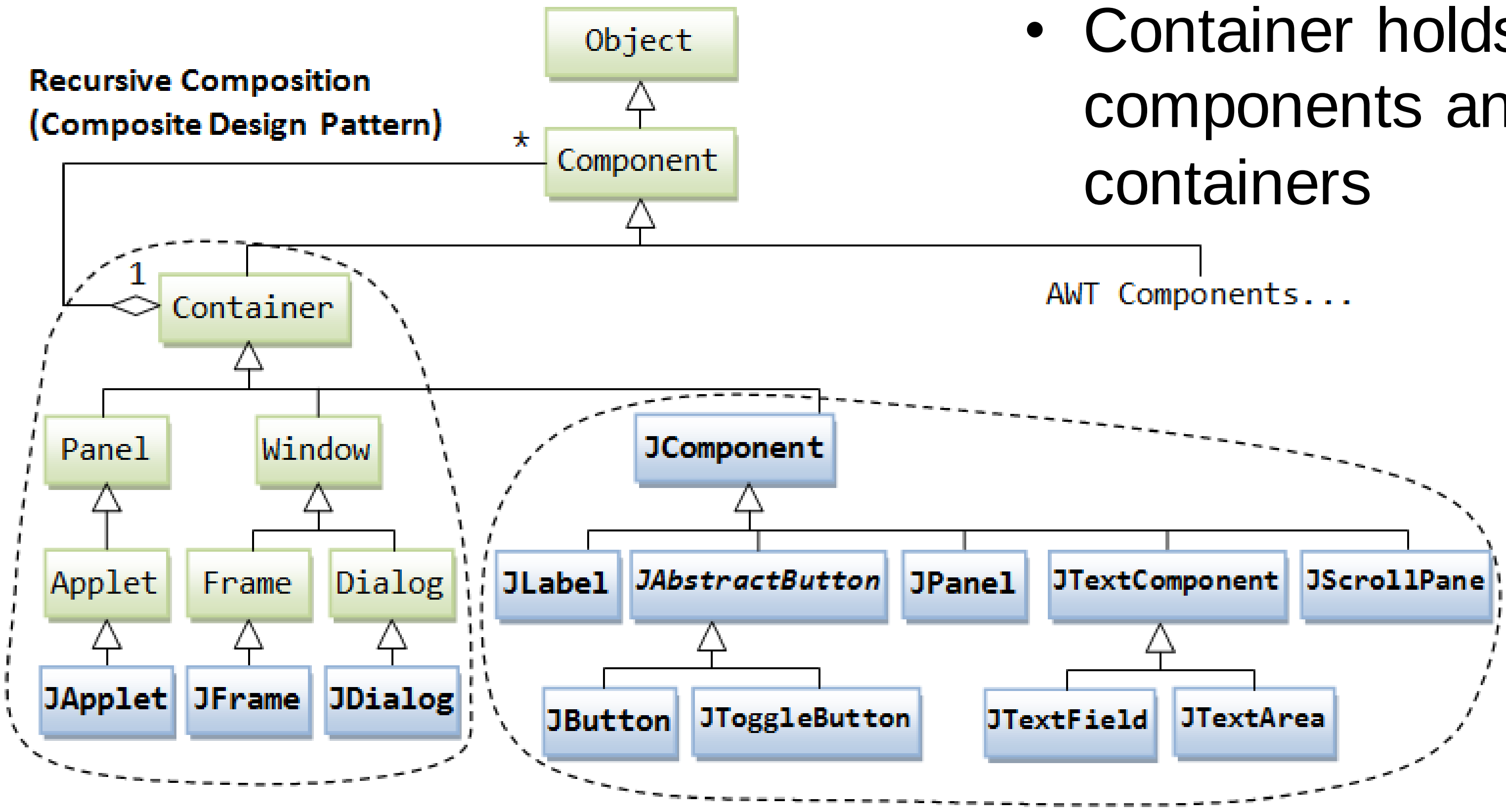
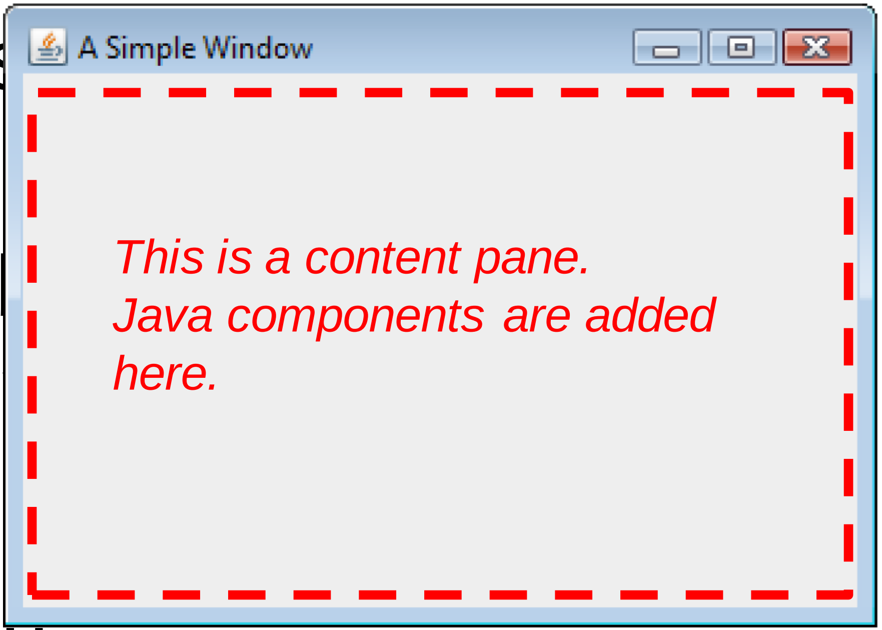


Image Source: https://www.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html#zz-10.

Creating a window in Swing

- A window is a *container*, which is simply a component that holds other components
- A container that can be displayed as a window is a *frame*
- In a Swing application, you create a frame from the `JFrame` class (top-level container)

javax.swing.JFrame



This is a content pane.
Java components are added here.

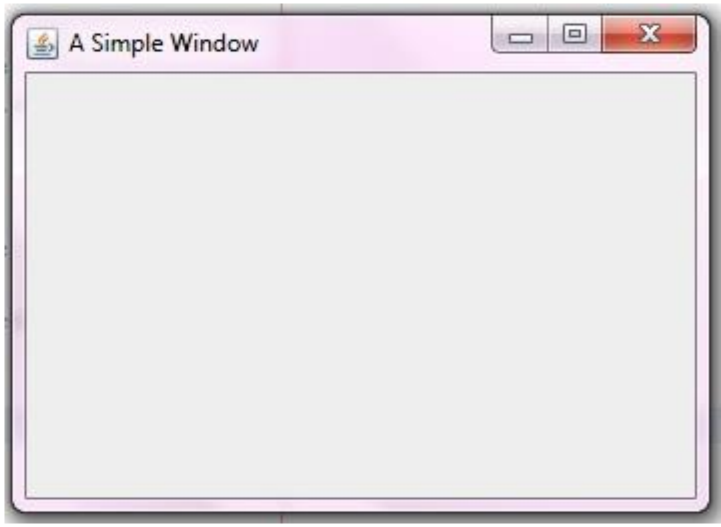
Creating a window

```
import javax.swing.*;   // Needed for Swing classes

public class ShowWindow
{
   public static void main(String[] args)
   {
      // Window width and height in pixels
      final int WINDOW_WIDTH = 350;
      final int WINDOW_HEIGHT = 250;

      JFrame window = new JFrame(); // Create a window
      window.setTitle("A Simple Window"); // Set the title
      window.setSize(WINDOW_WIDTH, WINDOW_HEIGHT); // Set the size of
                                                   //the window
      // Specify what happens when the close button is clicked
      window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      window.setVisible(true); // Display the window
   }
```



Creating a window Extending JFrame

```
import javax.swing.*;   // Needed for Swing classes

public class SimpleWindow extends Jframe {

   /* Constructor */
   public SimpleWindow() {

      final int WINDOW_WIDTH = 350;   // Window width in pixels
      final int WINDOW_HEIGHT = 250;  // Window height in pixels

      setTitle("A Simple Window"); // Set this window's title
      setSize(WINDOW_WIDTH, WINDOW_HEIGHT); // Set the size of
                                            //this window
      // Specify what happens when the close button is clicked.
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setVisible(true); // Display the window
   }
}
```

Note: You will need to create a main program to run this code

Adding Components

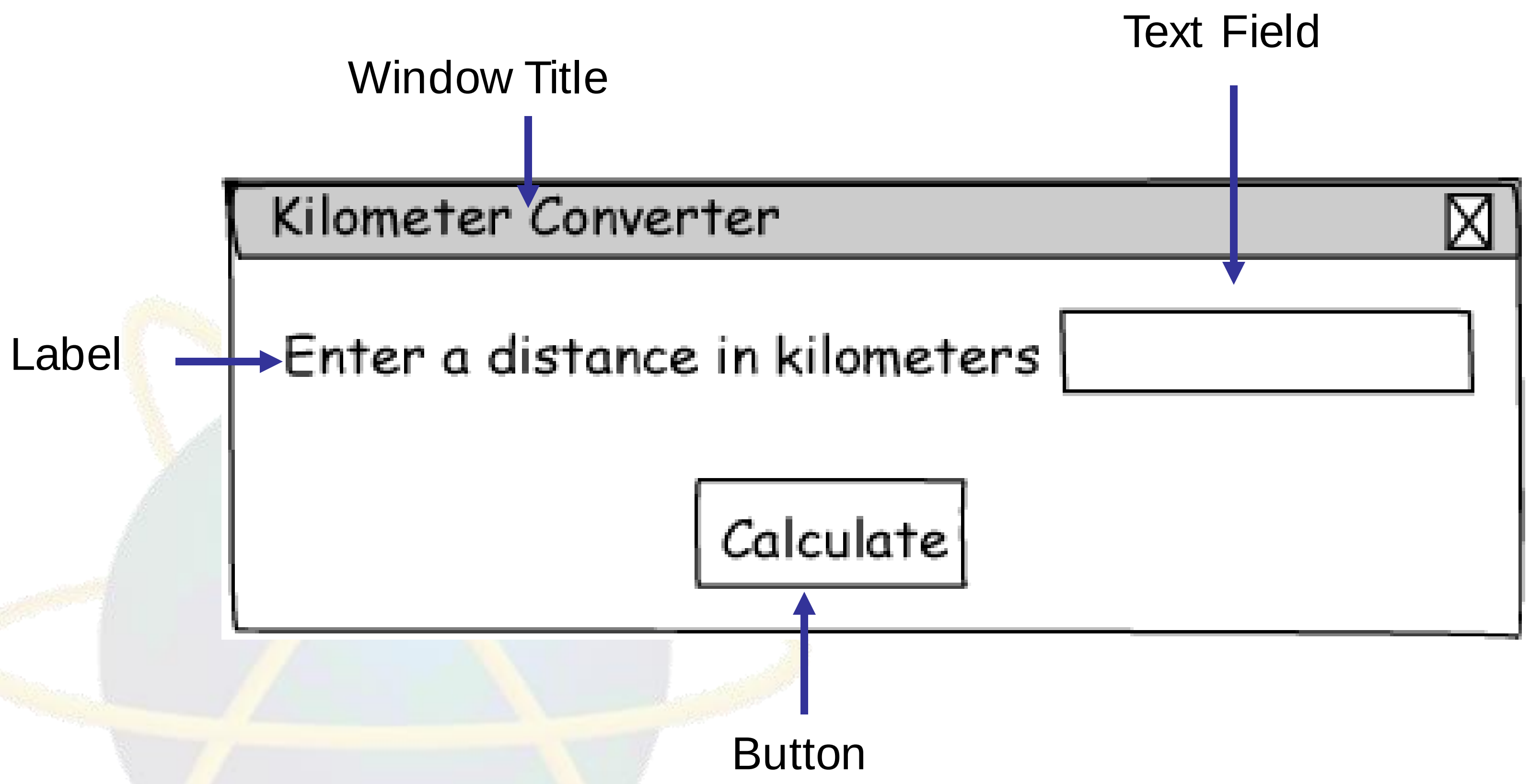
- Swing provides numerous components that can be added to a window.
- Swing components begin with a `J`
- Three fundamental components are:

`JLabel`: An area that can display text

`JTextField` : An area in which the user may type a single line of input from the keyboard

`JButton`: A button that can cause an action to occur when it is clicked

Adding Components



Adding Components

```
private JLabel message;
private JTextField kilometers;
private JButton calcButton;
…
message = new JLabel("Enter a distance in kilometers");
kilometers = new JTextField(10);
calcButton = new JButton("Calculate");
```

Adding Components

- A *<u>content pane</u>* is a container that is part of every `JFrame` object
- A *panel* is also a container that can hold GUI components
  - Panels cannot be displayed by themselves
  - Panels are commonly used to hold and organize collections of related components
  - Create panels with the `JPanel` class

Adding Components

```
private JPanel panel;
…
panel = new JPanel();
panel.add(message);
panel.add(kilometers);
panel.add(calcButton);
```

- Components are typically placed on a panel and then the panel is added to the `JFrame`'s content pane

```
add(panel);
```

Handling action events

- An *event* is an action that takes place within a program, such as the clicking of a button
- When an event takes place, the component that is responsible for the event creates an *event object* in memory
- The event object contains information about the event
- The component that generated the event object is know as the *event source*

Handling action events

- An *event listener* is an object that responds to events
- All event listener classes must *implement an interface*

Handling action events

- `JButton` components generate *action events*, which require an *action listener* class
- Action listener classes must meet the following requirements:
  - It must implement the `ActionListener` interface
  - It must have a method named `actionPerformed`
- The `actionPerformed` method takes an argument of the `ActionEvent` type

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

Registering a listener

- The process of connecting an event listener object to a component is called *registering* the event listener
- `JButton` components have a method named `addActionListener`

```
calcButton.addActionListener(
                  new CalcButtonListener());
```

The `ActionEvent` object

- Event objects contain certain information about the event
- This information can be obtained by calling one of the event object's methods
- Two of these methods are:
  - `getSource` - returns a reference to the object that generated this event
  - `getActionCommand` - returns the action command for this event as a `String`

Radio button

- `JRadioButton` constructors:
  - `JRadioButton(String` *`text)`*
  - `JRadioButton(String` *`text, boolean selected)`*
- Example:

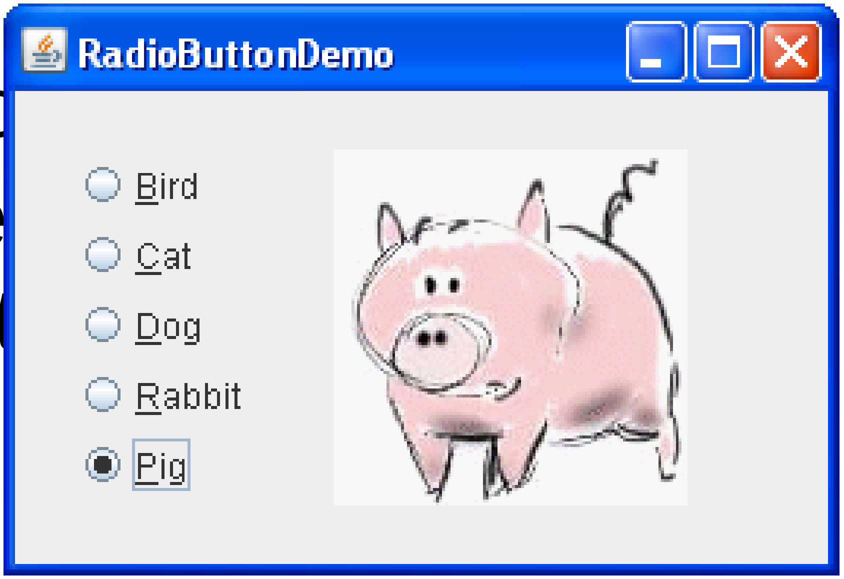
```
JRadioButton radio1 = new JRadioButton("Choice 1");
```

or

```
JRadioButton radio1 = new JRadioButton(
                             "Choice 1", true);
```

Button group

- Radio buttons normally are grouped together
- In a radio button group only one of the radio buttons in the group may be selected at any time
- The `ButtonGroup` c *mutually exclusive* re the radio buttons that



Button group

```
JRadioButton radio1 = new JRadioButton("Choice 1",
                                       true);
JRadioButton radio2 = new JRadioButton("Choice 2");
JRadioButton radio3 = new JRadioButton("Choice 3");
ButtonGroup group = new ButtonGroup();
group.add(radio1);
group.add(radio2);
group.add(radio3);
```

Button group

- `ButtonGroup` objects are not containers like `JPanel` objects, or content frames
- If you wish to add the radio buttons to a panel or a content frame, you must add them individually

```
panel.add(radio1);
panel.add(radio2);
panel.add(radio3);
```

Check boxes

Check appears in box if true

- Two `JCheckBox` constructors:

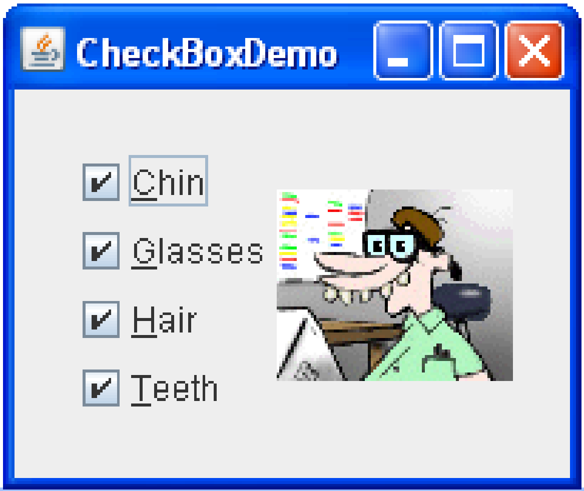
```
JCheckBox(String text)
JCheckBox(String text, boolean selected)
```

- Example:

```
JCheckBox check1 = new JCheckBox("Macaroni");
```

or

```
JCheckBox check1 = new JCheckBox("Macaroni", true);
```



Check boxes

- When a `JCheckBox` object is selected or deselected, it generates an *item event*
- Handling item events is similar to handling action events
- When writin an *item listener* class, the following requirements must be met:
  - It must implement the `ItemListener` interface
  - It must have a method named `itemStateChanged`
    - This method must take an argument of the

- Many more components available to be used …

Quick Review Questions

- What is GUI based programming?
- What is the difference between Swing and AWT?
- What is event handling?
- What is event listener?
- How to handle the event?

Summary of Main Teaching Points

- AWT vs Swing
- Swing components
- Creating a window
- Adding components
- Handling action events

Q & A

Next Session

- File Class
- I/O Stream