King's Own Institute
Success in Higher Education

ICT104

Program Design and Development

Lecture 7– A First Look at GUI applications

Adopted from: Gaddis & Gaddis (2019) Starting Out with Java: From Control Structures through Objects, 7th Edition.

KOI
King's Own Institute

Focus for this week

A First Look at GUI applications

- Introduction
- Creating Windows
- Equipping GUI Classes with a main() Method

Activity 1:
Revision Exercise

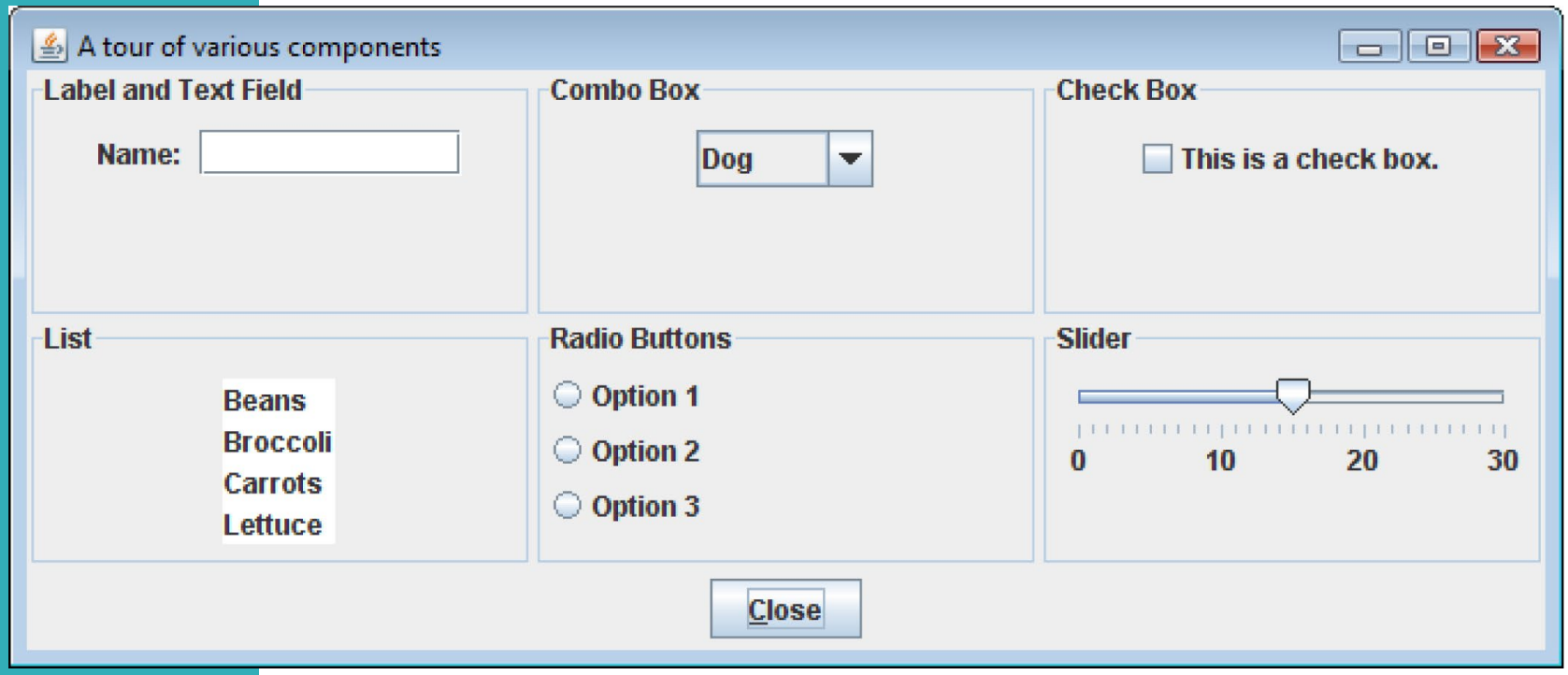
List any three concepts which you can remember from your previous week class

Introduction

- Many Java application use a *graphical user interface* or ***GUI*** (pronounced "gooey")
- A GUI is a graphical window or windows that provide interaction with the user
- GUI's accept input from
  - keyboard
  - 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, lists, and sliders



Activity 2: Discussion questions

Differentiate between:

A) Label and Text Box

B) Radio button and Checkbox

C) ComboBox and List

JFC, AWT, Swing

- Java programmers use the *Java Foundation Classes (**JFC**)* to create GUI applications
- The JFC consists of several sets of classes, many of which are beyond the scope of this class
- The two sets of JFC classes that we focus on are **AWT** and **Swing** classes
- Java is equipped with a set of classes for drawing graphics and creating graphical user interfaces
- These classes are part of the *Abstract Windowing Toolkit (AWT)*

JFC, AWT, Swing

- The AWT allows creation of applications and applets with GUI components
- The AWT does not actually draw user interface components on the screen
- The AWT communicates with a layer of software, *peer classes*
- Each version of Java for a particular operating system has its own set of peer classes

JFC, AWT, Swing

- 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
- The behavior of components across various operating systems can differ
- Programmers cannot easily extend the AWT components
- AWT components are commonly called *heavyweight components*

JFC, AWT, Swing

- *Swing* is a library of classes that provide an improved alternative for creating GUI applications and applets
- Very few Swing classes rely on peer classes, so they are referred to called *lightweight components*
- Swing draws most of its own components.
- Swing components have a consistent look and predictable behavior on any operating system
- Swing components can be easily extended

Activity 3: Poll

1. These components have a consistent look and predictable behaviour on any operating system

- ◯ AWT
- ◯ GUI
- ◯ Swing
- ◯ Peer classes

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

Activity 4: Poll

1. This is an action that takes place in an application, such as the clicking of a button.

- ○ instance
- ○ effect
- ○ case
- ○ event

`javax.swing` and `java.awt`

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

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

  - Note the letter `x` that appears after the word `java`
- Some of the AWT classes are used to determine when events, such as the clicking of a mouse, take place in applications
- In an application that uses an AWT class, it is necessary to use the following statement

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

  - Note that there is no `x` after `java` in this package name

Activity 5: Discussion questions

Differentiate between:

AWT and Swing class

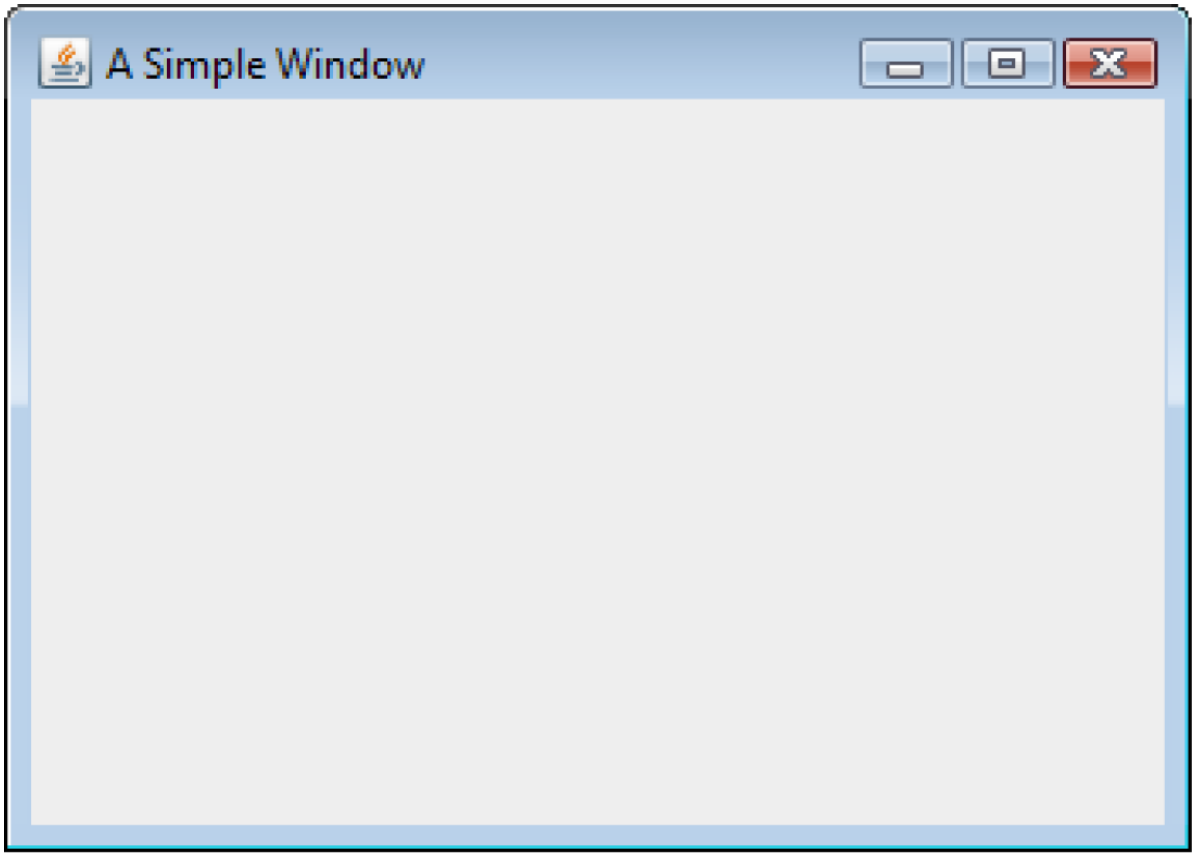
Creating Windows

- Often, applications need one or more windows with various components
- 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

Creating Windows

- A frame is a basic window that has:
  - a border around it,
  - a title bar, and
  - a set of buttons for:
    - minimizing,
    - maximizing, and
    - closing the window.
- These standard features are sometimes referred to as window *decorations*

Creating Windows



- See example: **`ShowWindow.java`**

Creating Windows

- The following `import` statement is needed to use the swing components:

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

- In the `main` method, two constants are declared:

```
final int WINDOW_WIDTH = 350;
final int WINDOW_HEIGHT = 250;
```

- We use these constants later in the program to set the size of the window
- The window's size is measured in pixels
- A *pixel (picture element)* is one of the small dots that make up a screen display

Creating Windows

- An instance of the `JFrame` class needs to be created

```
JFrame window = new JFrame();
```

- This statement
  - creates a `JFrame` object in memory and
  - assigns its address to the window variable

- The string that is passed to the `setTitle` method will appear in the window's title bar when it is displayed

```
window.setTitle("A Simple Window");
```

- A `JFrame` is initially invisible

Creating Windows

- To set the size of the window

```
window.setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
```

- To specify the action to take place when the user clicks on the close button

```
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

- The `setDefaultCloseOperation` method takes an `int` argument which specifies the action
  - `JFrame.HIDE_ON_CLOSE` - causes the window to be hidden from view, but the application does not end
  - The default action is `JFrame.HIDE_ON_CLOSE`

Creating Windows

- The following code displays the window:

```
window.setVisible(true);
```

- The `setVisible` method takes a `boolean` argument
  - `true` - display the window
  - `false` - hide the window

Extending `JFrame`

- We usually use inheritance to create a new class that extends the `JFrame` class
- When a new class extends an existing class, it inherits many of the existing class's members just as if they were part of the new class
- These members act just as if they were written into the new class declaration
- New fields and methods can be declared in the new class declaration
- This allows specialized methods and fields to be added to your window
- See examples: **`SimpleWindow.java`**, **`SimpleWindowDemo.java`**

Equipping GUI Classes with a `main()` Method

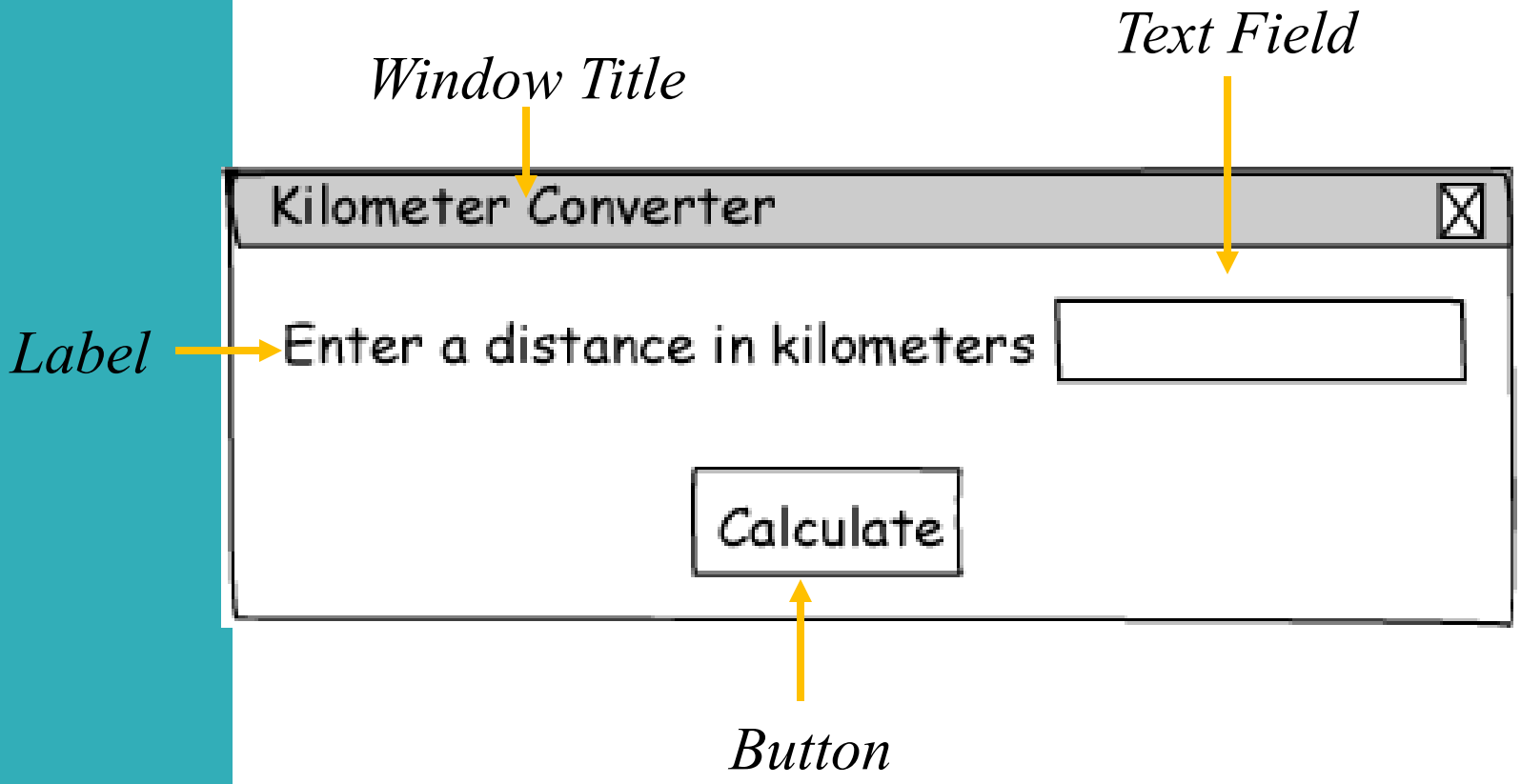
- Java applications always starts execution with a method named `main()`
- The previous example used two separate files
  - `SimpleWindow.java` -- the class that defines the GUI window
  - `SimpleWindowDemo.java` – containins the `main` method that creates an instance of the `SimpleWindow` class
- Applications can also be written with the `main` method directly written into the GUI class
- See example: **`EmbeddedMain.java`**

Adding Components

- Swing provides numerous components that can be added to a window
- 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

Sketch of Kilometer Converter Graphical User Interface



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");
```

- This code declares and instantiates three Swing components

Adding Components

- A *content pane* is a container that is part of every `JFrame` object
- Every component added to a `JFrame` must be added to its content pane. You do this with the `JFrame` class's `add` method
- The content pane is not visible and it does not have a border
- A *panel* is also a container that can hold GUI components

Adding 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

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

Adding Components

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

```
add(panel);
```

- See example: **`KiloConverter.java`**

Activity 6:
Poll

1. In Swing, labels are created with this class:

- ○ JFCLabel
- ○ AWTLabel
- ○ JLabel
- ○ SwingLabel

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*
- It is possible that the source component is connected to one or more event listeners

Handling Action Events

- An *event listener* is an object that responds to events
- The source component *fires* an event which is passed to a method in the event listener
- Event listener classes are specific to each application
- Event listener classes are commonly written as private inner classes in an application

Writing Event Listener Classes as Private Inner Classes

A class that is defined inside of another class is known as an inner class

```
public class Outer
{
   Fields and methods of the Outer class appear here.
   private class Inner
   {
      Fields and methods of the Inner class appear here
   }
}
```

Event Listeners Must Implement an Interface

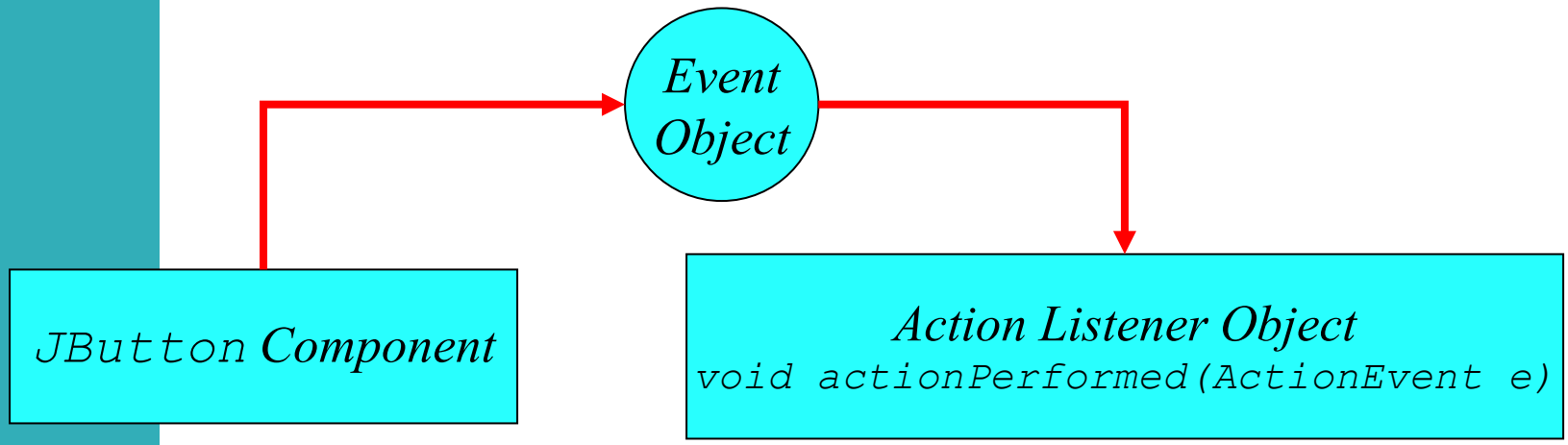
- All event listener classes must *implement an interface*
- An interface is something like a class containing one or more method headers
- When you write a class that implements an interface, you are agreeing that the class will have all of the methods that are specified in the interface

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)
{
   Code to be executed when button is pressed goes here.
}
```

Handling Action Events



When the button is pressed ...

The `JButton` *component generates an event object and passes it to the action listener object's* `actionPerformed` *method*

- See example: **`KiloConverter.java`**

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());
```

- When the user clicks on the source button, the action listener object's `actionPerformed` method will be executed

Activity 7: Poll

1. To use the ActionListener interface, as well as other event listener interfaces, you must have the following import statement in your code:

- ○ import java.swing;
- ○ import java.atw;
- ○ import java.atw.*;
- ○ import java.awt.event.*;

Background and Foreground Colors

- Many of the Swing component classes have methods named `setBackground` and `setForeground`
- `setBackground` is used to change the color of the component itself
- `setForeground` is used to change the color of the text displayed on the component
- Each method takes a color constant as an argument

Color Constants

- There are predefined constants that you can use for colors

```
Color.BLACK                Color.BLUE
Color.CYAN                 Color.DARK_GRAY
Color.GRAY                 Color.GREEN
Color.LIGHT_GRAY           Color.MAGENTA
Color.ORANGE               Color.PINK
Color.RED                  Color.WHITE
Color.YELLOW
```

- See example: **`ColorWindow.java`**

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`

Activity 8:
Poll

1. In a Swing application, you create a frame object from the:

- ◯ Jlabel class
- ◯ JFrame class
- ◯ Jpanel class
- ◯ AbstractButton class

Summary of today's lesson

A First Look at GUI applications

- Introduction
- Creating Windows
- Equipping GUI Classes with a main() Method

Activity 9:
Reflection Exercise

List any four concepts you have learnt in today's lesson

Activity 10: Homework Exercise

Write a program using Eclipse or NetBeans to implement any one concept you have learnt in today's lesson