COMP 102.2x
Introduction to Java Programming – Part 2
Lecture 3

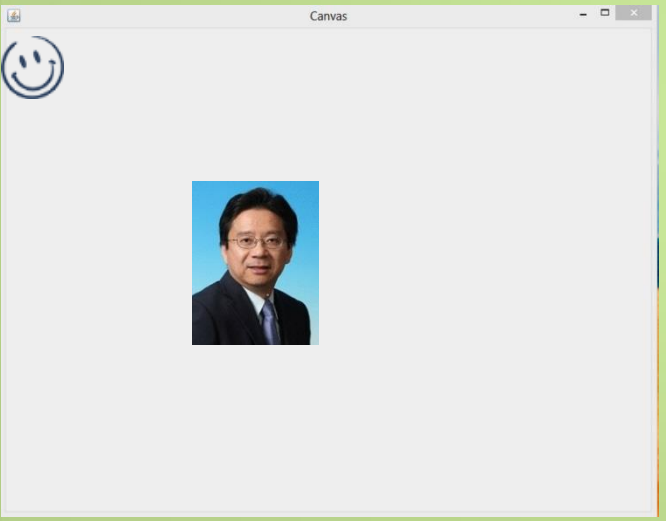
T.C. Pong
Department of Computer Science & Engineering
HKUST

Lecture 3

- Event driven programming
- Graphical User Interface (or GUI)

Procedural programming

- Code is executed in a step by step manner.

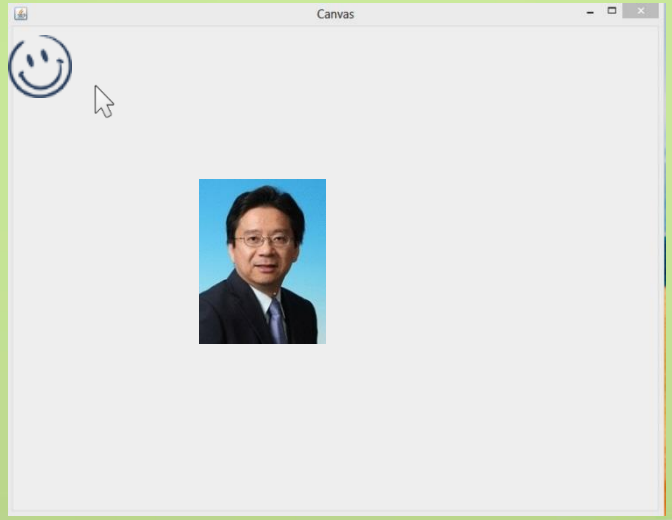



```
// A simple demo on Procedural vs Event Driven Programming
public class EventDrivenDemo
{
   private Canvas canvas = new Canvas();
   ColorImage image1 = new ColorImage("happyFace.png");
   ColorImage image2 = new ColorImage("tcpong.jpg");

   public EventDrivenDemo() {
      canvas.add(image2, 200, 200);
      canvas.add(image1);
   }
   public void moveHappyFace(int x, int y){
      image1.setX(x);
      image1.setY(y);
   }
}
```

Event driven programming

- In event driven programming, the flow of the program is determined by events.
- For example, drag an image using a mouse to overlay that with another image.



Events

In event-driven programming, code is executed upon activation of events.

- An event can be defined as a type of signal to the program that something has happened.
- The event can be generated by external user actions such as mouse movements, mouse clicks, and keystrokes, or by the operating system, such as a timer.

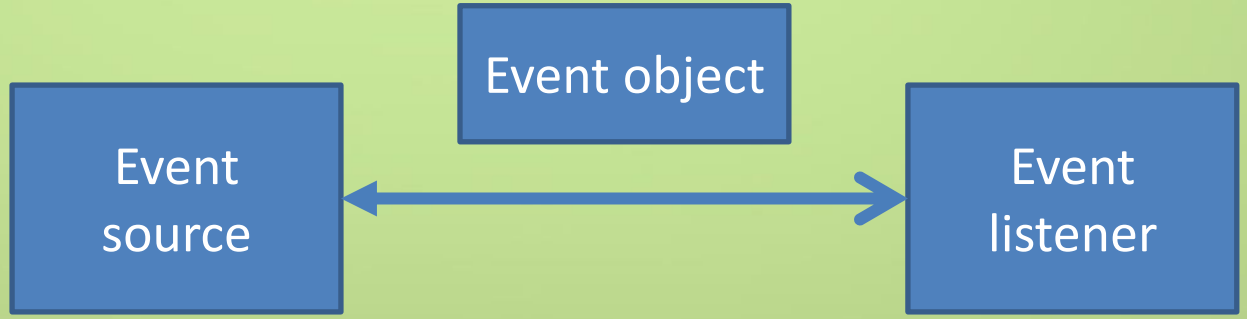
Delegation Event Model

1. An event is generated when a user interacts with a graphical component on the Graphical User Interface (GUI).
2. Once the event is generated, the event is passed (or delegated) to other objects which handle the event.
3. The objects which handle the events are called Event listeners/handlers.

Delegation Event Model

Three main components:

1. Event source
2. Event object
3. Event listener



Event source

- The event source is the origin of which the event occurs.
- For example
  - The Canvas as the source of mouse clicked events.
  - In same games, one can design a cannon object to be a source for generating cannon fired event.

Event object

- An event object contains the necessary information describing the event.
- For example
  - A mouse clicked event may include the x, y positions of the mouse on the Canvas.
  - A cannon fired event may include the tilt angle of the cannon when it is fired.

Event listener

- The event listener (or handler) is the logic of how the event should be handled.
- For example
  - The mouse clicked listener can show a color image at the x, y position specified by the mouse click event.
  - The cannon fired listener can move the cannon ball to the destination position according to the tilt angle specified by the cannon fired event.

Interface

- An interface is a group of related methods with empty bodies.
- **ALL** these methods **must be defined** by any class which implements that interface.
- An interface declaration is similar to a class declaration without method bodies, instance and static variables.
- *For example:*

```
public interface ActionListener {
        public void actionPerformed (ActionEvent e);
}
```

Example: interface

```
// The Shape interface describes the common shape features
public interface Shape {
    public double area();
    public double perimeter();
}
```

```
public class Rectangle implements Shape {
    private double width;
    private double height;

    public Rectangle(double w, double h) {
        width = w;
        height = h;
    }
    public double area() {
        return width*height;
    }
    public double perimeter() {
        return (width+height) * 2;
    }
}
```

```
public class Circle implements Shape {
    private double radius;
    private final double PI = 3.1416;

    public Circle (double r) {
        radius = r;
    }
    public double area() {
        return PI * radius * radius;
    }
    public double perimeter() {
        return 2 * PI * radius;
    }
}
```

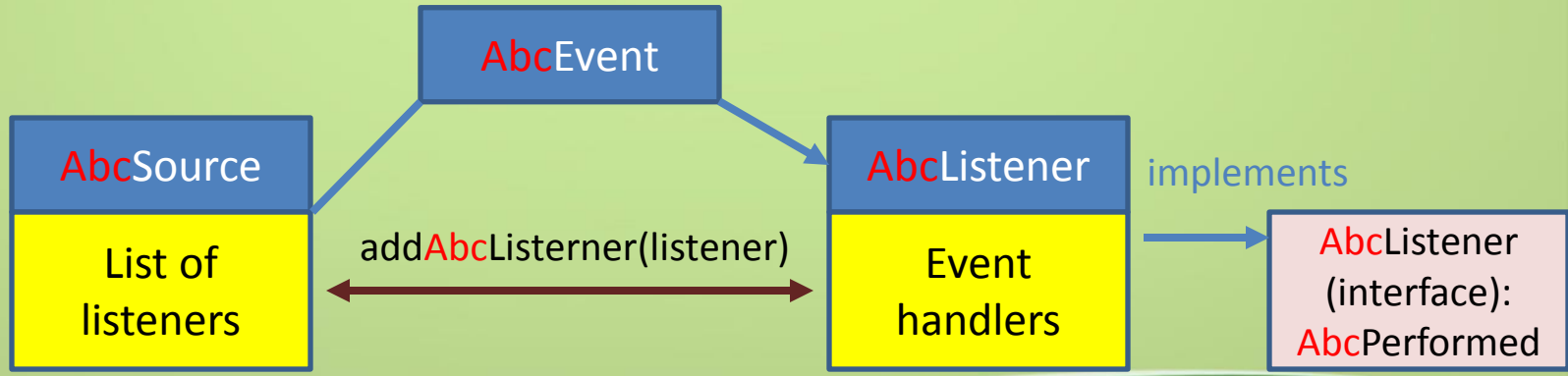
The Mechanism

- Suppose we have an event called *Abc*.
- Then the *Abc*Source class will have a method named:
  - add*Abc*Listener(*Abc*Listener listener)

  for the source object to register its listeners.
- Calling this method allows the source class to know which listener it should notify when the event occurs.

The Mechanism

- The *Abc*Listener class will have a method named:
  - public void *Abc*Performed (*Abc*Event e) {
    // handling the logic
    }
- When the event *Abc* occurs, the method will be called by the *Abc*Source class, passing along the *Abc*Event object which describes the event.



Event Listener

- All listeners interested in an event must implement the event listener interface.
- The Event Listener class is a Java interface which contains a set of methods to be implemented.

```
class MyListener implements AbcListener {
    public void AbcPerformed(AbcEvent e) {
            // my handling logic
    }
}
```

Example: Event Driven Programming

```
import comp1022p.Canvas;
import comp1022p.ColorImage;
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;

public class MyListener implements MouseListener {
   private Canvas canvas;

   public MyListener ( ) {
      canvas = new Canvas();
      canvas.addMouseListener(this);
   }

   public void mouseClicked(MouseEvent e) {
      ColorImage image = new ColorImage("happyFace.png");
      int x = e.getX() - image.getWidth()/2;
      int y = e.getY() - image.getHeight()/2;
      canvas.add(image, x, y);
   }

   public void mousePressed(MouseEvent e) { }
   public void mouseReleased(MouseEvent e) { }
   public void mouseEntered(MouseEvent e) { }
   public void mouseExited(MouseEvent e) { }
}
```

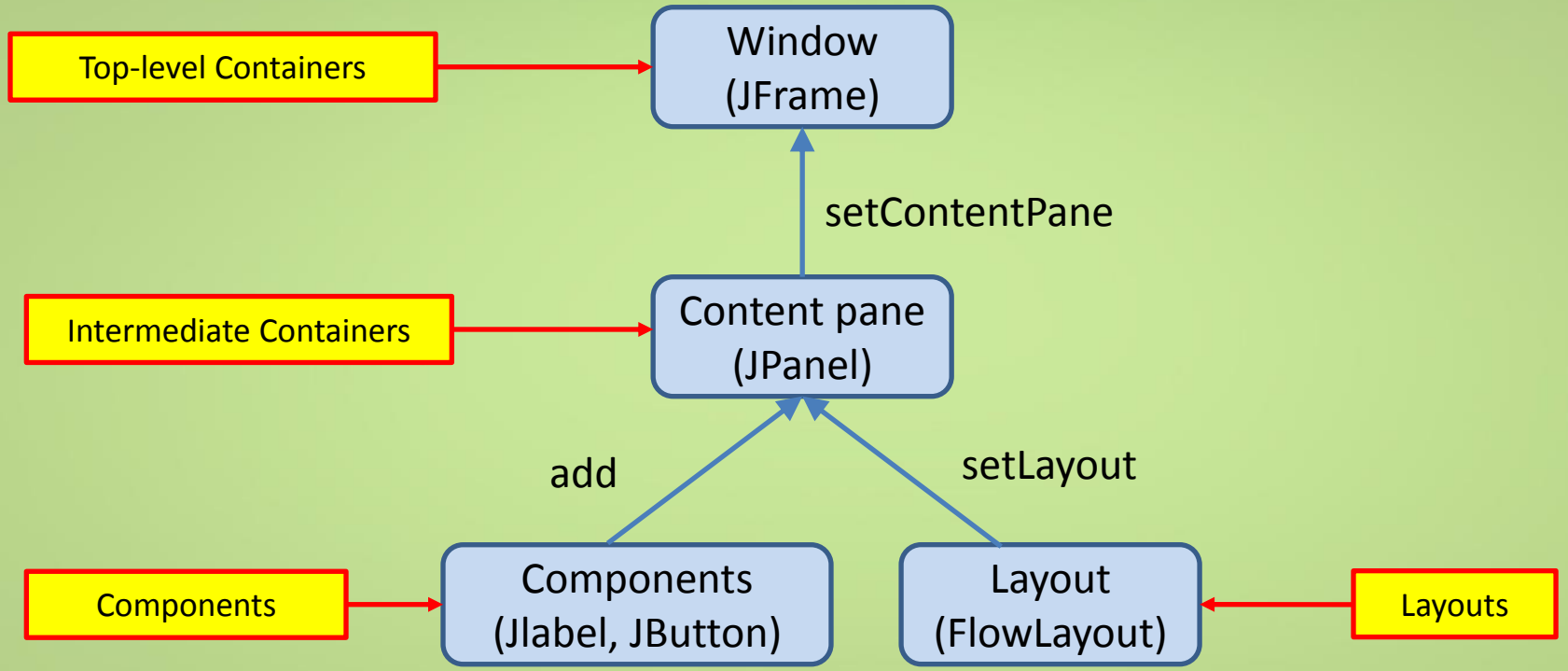
Lecture 3

Graphical User Interface (GUI)

GUI

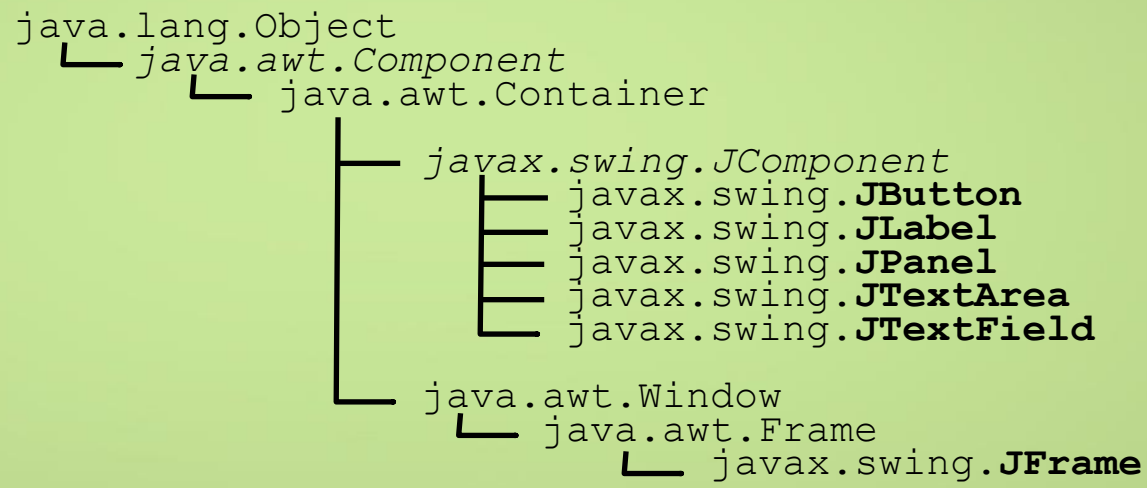
- Modern GUIs are event-driven
- Events occur when the user interact with the graphic components:
  - *A mouse click on a button*
  - *A mouse drag on an image*
  - *Some text is input into a textbox*
  - *An item is selected from a pull-down menu*
  - *A window is to be resized or closed*
  - *…*

Overall Structure



Swing component hierarchy

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



Subclass and Inheritance

- A subclass is a class that is derived from another class (superclass).
  - public class SubclassName extends SuperClassName
- The class **Object** is the root of the Java class hierarchy.
- A subclass inherits all the fields and methods from its superclass.
- The keyword super can be used for a subclass to invoke the constructors or methods of its superclass.

Simple GUI Example

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

class MyWIndowDemo extends JFrame {
   public MyWindowDemo() {
      setTitle(“Window created using JFrame");
      setSize(400, 200);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setVisible(true);
}
```

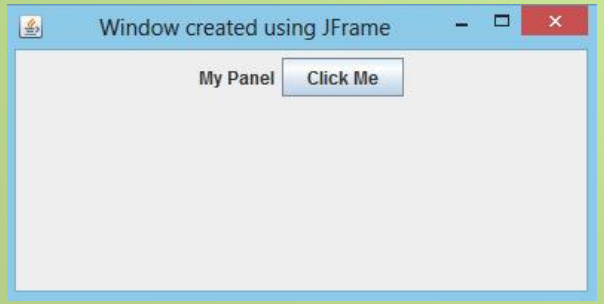
Simple GUI Example

```
import java.awt.*;
import javax.swing.*;
class MyWindowDemo2 extends JFrame {
   public MyWindowDemo2() {
      setTitle(“Window created using JFrame");
      setSize(400, 200);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setVisible(true);

      JPanel content = new JPanel();
      content.setLayout(new FlowLayout());

      JLabel label = new JLabel(“My Panel");
      content.add(label);
      JButton button = new JButton("Click Me");
      content.add(button);

      setContentPane(content);
}
```



Simple GUI Example

```
import java.awt.*;
import javax.swing.*;
Import java.awt.event.*;
class MyWindowDemo3 extends JFrame implements MouseListener {
   public MyWindowDemo2() {
      setTitle("Window created using JFrame");
      setSize(400, 200);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setVisible(true);

      JPanel content = new JPanel();
      content.setLayout(new FlowLayout());

      JLabel label = new JLabel("My Panel");
      content.add(label);
      JButton button = new JButton("Click Me");
      content.add(button);

      button.addMouseListener(this);
      setContentPane(content);
   }
```

```
  public void mouseClicked(MouseEvent e) {
     Toolkit.getDefaultToolkit().beep( );
  }

  public void mousePressed(MouseEvent e) { }
  public void mouseReleased(MouseEvent e) { }
  public void mouseEntered(MouseEvent e) { }
  public void mouseExited(MouseEvent e) { }
}
```

Simple GUI Example

```
public void mouseClicked(MouseEvent e) {
    Toolkit.getDefaultToolkit().beep( );
}

public void mouseEntered(MouseEvent e) {
    label.setText("Entered" );
}
public void mouseExited(MouseEvent e) {
   label.setText("Exited");
}
public void mouseClicked(MouseEvent e) {
   Toolkit.getDefaultToolkit().beep( );
}

public void mousePressed(MouseEvent e) { }
public void mouseReleased(MouseEvent e) { }
public void mouseEntered(MouseEvent e) { }
public void mouseExited(MouseEvent e) { }
}
```

```
import java.awt.*;
import javax.swing.*;
Import java.awt.event.*;
class MyWindowDemo4 extends JFrame  implements MouseListener {
   public MyWindowDemo2() {
      setTitle("Window created using JFrame");
      setSize(400, 200);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setVisible(true);
      JPanel content = new JPanel();
      content.setLayout(new FlowLayout());
      content.addMouseListener(this);
      JLabel label = new JLabel("My Panel");
      content.add(label);
      JButton button = new JButton("Click Me");
      content.add(button);
      button.addMouseListener(this);
      setContentPane(content);
   }
```