Programming with Java for Beginners
Bineet Sharma

Graphics Programming

Graphics Programming Series

Assumptions & Expectations

Graphics Programming

- **Assumptions**
  - Input Output & Collections
- **Expectations**
  - Understand GUI programming

Objectives:

Graphics Programming

- **Introducing**
- **Window Controls**
- **Event Handling**

Introducing :

Graphic Programming Series I

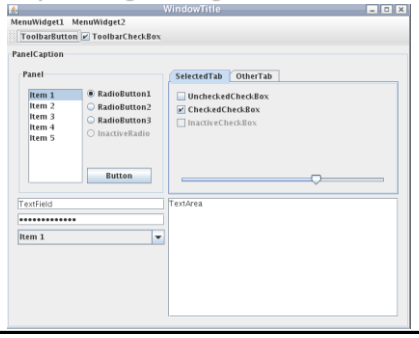
So far our program has been console program. Which runs in a command window. It did not know how to:

- Show or handle graphics
- Process mouse events
- Let user type in an window
- Provide a list box, or a drop down
- Give multiple choices with check boxes
- Give single choice with radio buttons

And many more using Graphical User Interface (GUI). It is about time.

Introducing:

Graphics Programming Series I



From Wiki

Introducing:

Graphic Programming Series I

So, how do we achieve that?

Java provides multiple different ways to build solid GUI application (desktop application and applets as well).

One of the oldest way is by AWT (Abstract Window Toolkit). API Another way is using Swing API.

Introducing:

Graphic Programming Series I

AWT (Advanced Window Toolkit): is a java package which can be used in any Java program.

import java.awt.*;

It provides many classes to be used for different OS, or as applet in browser. Adequate for many applications.

It uses controls defined by your OS: it uses the lowest common denominator for portability.

Is lighter and faster. Good way to start learning GUI

Introducing:

Graphic Programming Series I

Swing: Similar concept as AWT. Newer than AWT. More controls to make richer GUI (JSlider, JColorChooser, JSpinner)

import javax.swing.*;

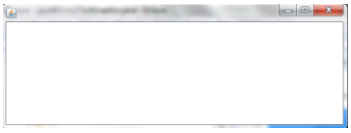
Many of the common controls are used in both:

AWT	**Swing**
Buttons	Jbuttons
Frames	Jframes
CheckBox	JCheckBox
Label	JLabel

Introducing:

Graphic Programming Series I

What a shortest GUI program looks like in Java?



```
import java.awt.*;

class TestGraphics {
   public static void main(String[] args) {
   Frame newFrame = new Frame();
   newFrame.setSize(550, 200);
   newFrame.setVisible(true);
   }
}
```

Introducing:

Graphic Programming Series I

So, what is involved in GUI Programming?

Your application usually first displays something. So, you need a **Frame** or **Dialog** to display (e.g. text, image, input/output widgets – called **components**)
You arrange these components using **layout.**
All these components interact with each other to solve your programming needs.
They listen to each other, so, you need to attach **listeners** to these components.
When an user interacts with these components an event is generated by OS and captured by listening components.

Introducing:

Graphic Programming Series I

Container: A container holds and displays **Components**. Examples of container include:
Panel, Window and Frame

Component: Components are widgets you use to get input/output from users. Examples of component include:
Checkbox, Label, Scrollbar, Menu, TextField

Remember that a **Container** is also a **Component**. This is a clever way to nest containers.

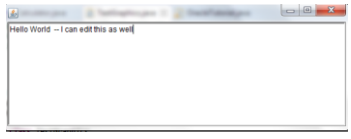
Introducing:

Graphic Programming Series I

Frame: A frame is a subclass of Container class. It has a title bar and a border around.

You can create multiple components and add to a container like Frame.

```
final Frame frame = new Frame();
final TextField textField = new TextField("Hello World");
frame.add(textField);
frame.setSize(550, 200);
frame.setVisible(true);
```



Introducing:

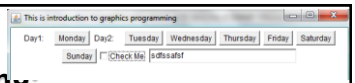
Graphic Programming Series I

All **container** has common functionality – as they are derived from Component.

Container provides event-handling methods.

All containers have these functionalities:

add(Component)
remove(Component)
getComponents()
setLayout(LayoutManager)
getLayout()

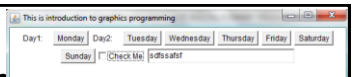


Introducing.

Graphic Programming Series I

Layout Manager: Use a layout manager to arrange components inside a Container.

```
Frame content = new Frame("This is introduction to graphics programming");
   content.setBackground(Color.CYAN);
   content.setLayout(new FlowLayout());
   content.add(new Label("Day1:"));
   content.add(new Button("Monday"));
   content.add(new Label("Day2:"));
   content.add(new Button("Tuesday"));
   content.add(new Button("Wednesday"));
…
   content.add(new Checkbox("Check Me"));
   content.add(new TextField("Write Here", 30));
   content.setSize(550, 200); // width, height
content.setVisible(true);
```

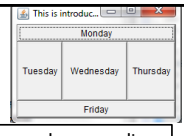


Introducing.

Graphic Programming Series I

LayoutManager: Is a abstract class, you must subclass it or use these derived classes (More Layout Managers):

BorderLayout: Arrange in North, East, Center, West, South direction (maximum five comonents)
CardLayout: Like tabbed dialogs
GridLayout: Arrange in grid of fixed size. Components are placed in left to right and top to down
FlowLayout: Default frame layout. Arrange in a row and aligned, place on new line when needed



Introducing:

Graphic Programming Series I

FlowLayout: It is convenient, but, not that appealing. Components are added left to right and down if there is no room in the right side.
BorderLayout: Add five components in one panel. If need more add more panel.

```
content.setLayout(new BorderLayout());
content.add(new Button("Monday"),
BorderLayout.NORTH);
content.add(new Button("Tuesday"),
BorderLayout.WEST);
content.add(new Button("Wednesday"),
BorderLayout.CENTER);
content.add(new Button("Thursday"),
BorderLayout.EAST);
content.add(new Button("Friday"), BorderLayout.SOUTH
```

Introducing:
Graphics Programming Series I

- **Demo**

Introducing:

Event Handling:
Graphic Programming Series II

Java GUI and Windowing systems use the **event-driven** model to interact with the user.

OS is constantly polling for these events. When it finds one, it tells you the (program), what just happened.

You can chose not to care about that event, in that case, OS handles with default behavior, or you (your program) do something with that event – you can take it and ignore it as well.

Event Handling:
Graphic Programming Series II

What are these events?

A window is closed, maximized, minimized
A mouse is moved, clicked or dragged
A button is clicked
A value is selected from list box
A radio button is pressed
A text is typed in a text box

All of these generates events and OS first grabs it and then sends to those components which generated it.

Event Handling:
Graphic Programming Series II

How do you handle them in Java GUI?

Most component (button, textbox), already handle some events, e.g. button click, check mark display.

Components send the event and listeners listen for such event. Different components send different event

You can attach a *listener* to a component to associate an action with a component.

Event Handling:
Graphic Programming Series II

There are multiple ways to handle events.

Window: Handle a window close event.

```
Frame content = new Frame("This is a frame");
content.add(new Button("Monday"), BorderLayout.NORTH);
content.addWindowListener(new WindowAdapter() {
//inner anonymous class
        public void windowClosing(WindowEvent e) {
        System.exit(0);
         }
 }
);
```

Event Handling:
Graphic Programming Series II

There are multiple ways to handle events.

Window: Implement WindowListener and define the method windowClosing

```
class TestGraphics extends Frame implements WindowListener {
public TestGraphics(String s) {
addWindowListener(this); // listen for events on this Window
}
public void paint(Graphics g){
g.drawString("Hello World", 100, 100);
}
// define methods in WindowListener interface
public void windowClosing(WindowEvent event) {
System.exit(0);
}
```

Event Handling:
Graphic Programming Series II

There are multiple ways to handle events.

Button: You need a ActionListener, which must have a matching actionPerformed(ActionEvent) method

```
public TestGraphics(String s) {
      add(pushButton);
      pushButton.addActionListener(this);
}

// define action for Button press
public void actionPerformed(ActionEvent event) {
   if (event.getActionCommand().equals("Click Me")) {
     System.out.println("ouch!");}}
}
```

Event Handling:
Graphic Programming Series II

There are multiple ways to handle events.

TextField: You need a ActionListener – waiting for enter key pressed, which must have a matching actionPerformed(ActionEvent) method.

A TextListner listens for any and all keys. This requires a textValueChanged method implemented

Simple Graphics:
Graphic Programming Series II

```
final TextField textfield = new TextField("Type something");
textfield.setEnabled(true);
textfield.setEditable(true);
textfield.addTextListener(new TextListener() {
    public void textValueChanged(TextEvent e) {
        System.out.println(textfield.getText());
    }
});
frame.add(textfield);
frame.setSize(550, 200);
frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
}
});
```

Event Handling:
Graphics Programming Series II

- **Demo**

EventHandling:

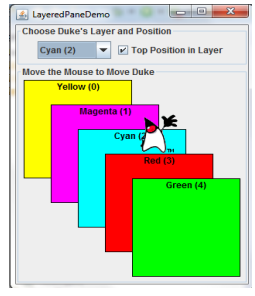
Simple Graphics:
Graphic Programming Series III

Graphics: Use this abstract class for drawing graphics.

drawString(str, x, y)
drawRect(x1, y1, width, height)
drawRoundRect(x1, y1, width, height, arcWidth, arcHeight)
drawRect3D(x1, y1, width, height, raised)

Image img = getImage(getDocumentBase(), "hello.gif");
g.drawImage(img, 0, 0, this);

Simple Graphics:
Graphic Programming Series III



Oracle Tutorial: http://docs.oracle.com/javase/tutorial/uiswing/components/layeredpane.html

Summary:

Graphics Programming Using Java

- **Introducing**
- **Window Controls**
- **Event Handling**