Chapter 8: Getting Started with Graphics Programming

- AWT Class Hierarchy
- Frames
- The relationship between frame and other UI components
- Event-Driven Programming
  - Event Source, Listener, Listener Interface
- Layout Managers
  - FlowLayout, GridLayout, BorderLayout
- Panels
- Canvases
- repaint(), update() and paint()
- Drawing Geometric Figures
  - Color, Font, FontMetrics classes
  - Drawing methods

Frames

- Frame is a window that is not contained inside another window. Frame is the basis to contain other user interface components in Java graphical applications.
- The Frame class can be used to create windows.

```java
import java.awt.*;
public class MyFrame
{
    public static void main(String[] args)
    {
        Frame f = new Frame("Test Frame");
        f.setSize(400,300);
        f.setVisible(true);
    }
}
```
Event-Driven Programming

- *Procedural programming* is executed in procedural order.
- In *event-driven programming*, code is executed upon activation of events.

Events

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

- **id**: A number that identifies the event.
- **target**: The AWT component upon which the event occurred.
- **arg**: Additional information about the AWT components.
- **x, y coordinates**: The mouse pointer location when a mouse movement event occurred.
- **clickCount**: The number of consecutive clicks for the mouse events. For other events, it is zero.
- **when**: The time stamp of the event.
- **key**: The key that was pressed or released.

Event Classes

- AWTEvent
- EventObject
- AdjustmentEvent
- ComponentEvent
- TextEvent
- ItemEvent
- ActionEvent
- InputEvent
- WindowEvent
- MouseEvent
- KeyEvent
- ContainerEvent
- FocusEvent
- PaintEvent

Selected User Actions

<table>
<thead>
<tr>
<th>User Action</th>
<th>Source Object</th>
<th>Event Type Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clicked on a button</td>
<td>Button</td>
<td>ActionEvent</td>
</tr>
<tr>
<td>Changed text</td>
<td>TextComponent</td>
<td>TextEvent</td>
</tr>
<tr>
<td>Double-clicked on a list item</td>
<td>List</td>
<td>ActionEvent</td>
</tr>
<tr>
<td>Selected or deselected an item with a single click</td>
<td>List</td>
<td>ItemEvent</td>
</tr>
<tr>
<td>Selected or deselected an item</td>
<td>Choice</td>
<td>ItemEvent</td>
</tr>
</tbody>
</table>

The Delegation Model

1. **User action**
2. **Generating an event**
3. **Notify listener**
   - **Source Object**
   - **Register a listener object**
   - **Listener Object**
   - **Event Handler**
   - **EventObject**
   - **Targeting an event**
### Selected Event Handlers

<table>
<thead>
<tr>
<th>Event Class</th>
<th>Listener Interface</th>
<th>Listener Methods (Handlers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionEvent</td>
<td>ActionListener</td>
<td>actionPerformed(ActionEvent)</td>
</tr>
<tr>
<td>ItemEvent</td>
<td>ItemListener</td>
<td>itemStateChanged(ItemEvent)</td>
</tr>
<tr>
<td>WindowEvent</td>
<td>WindowListener</td>
<td>windowClosing(WindowEvent)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>windowOpened(WindowEvent)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>windowIconified(WindowEvent)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>windowDeiconified(WindowEvent)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>windowClosed(WindowEvent)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>windowActivated(WindowEvent)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>windowDeactivated(WindowEvent)</td>
</tr>
<tr>
<td>ContainerEvent</td>
<td>ContainerListener</td>
<td>componentAdded(ContainerEvent)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>componentRemoved(ContainerEvent)</td>
</tr>
</tbody>
</table>

### Example 8.1

**Closing Windows**

- Objective: Extend `java.awt.Frame` with window-closing capability.

```java
MyFrameWithExitHandling
```

### Example 8.2

**Handling Simple Mouse Events**

- Objective: Create a frame and display a solid square at the mouse pointer when the mouse is clicked.

```java
TestMouseEvent
```

### Example 8.3

**Handling Simple Action Events**

- Objective: Display a Close button in the window; terminate the program by clicking the Close button in the window or on the title bar.

```java
TestActionEvent
```
Layout Managers

- Java’s layout managers provide a level of abstraction to automatically map your user interface on all windowing systems.
- The AWT components are placed in containers. Each container has a layout manager to arrange the AWT components within the container.

Kinds of Layout Managers

- FlowLayout
- GridLayout
- GridBagLayout
- BorderLayout
- CardLayout

LayoutManager Hierarchy

Example 8.4 Testing the FlowLayout Manager

The components are arranged in the container from left to right in the order in which they were added. When one row becomes filled, a new row is started.
**FlowLayout Constructors**

- `public FlowLayout(int align, int hGap, int vGap)`: Constructs a new `FlowLayout` with a specified alignment, horizontal gap, and vertical gap. The *gaps* are the distances in pixels between components.

- `public FlowLayout(int alignment)`: Constructs a new `FlowLayout` with a specified alignment and a default gap of five pixels for both horizontal and vertical.

- `public FlowLayout()`: Constructs a new `FlowLayout` with a default center alignment and a default gap of five pixels for both horizontal and vertical.

**Example 8.5**

Testing the GridLayout Manager

The `GridLayout` manager arranges components in a grid (matrix) formation with the number of rows and columns defined by the constructor. The components are placed in the grid from left to right starting with the first row, then the second, and so on.

**GridLayout Constructors**

- `public GridLayout(int rows, int columns)`: Constructs a new `GridLayout` with the specified number of rows and columns.

- `public GridLayout(int rows, int columns, int hGap, int vGap)`: Constructs a new `GridLayout` with the specified number of rows and columns, along with specified horizontal and vertical gaps between components.

**Example 8.6**

Testing the BorderLayout Manager

The `BorderLayout` manager divides the window into five areas: East, South, West, North, and Center. Components are added to a `BorderLayout` by using `add(String, Component)`, where `String` is "East", "South", "West", "North", or "Center".
Example 8.7
Testing Panel

- Panels act as smaller containers for grouping user interface components.
- It is recommended that you place the user interface components in panels and place the panels in a frame. You can also place panels in a panel.

Example 8.7
Using Canvases

Canvas is a UI component that can be used to draw graphics (including text) and enable user interaction.

repaint(), update(), and paint()

The Java system automatically creates a default graphics context, an object of the Graphics class, and passes it as a parameter to the update() and paint() methods. This object is local to those methods, and it cannot be used outside of those methods.

Colors

Color c = new Color(r, g, b);
r, g, and b specify a color by its red, green, and blue components.

Example:
Color c = new Color(128, 100, 100);
Setting Colors

You can use the following methods to set the component’s background and foreground colors:

- `setBackground(Color c)`
- `setForeground(Color c)`

Example:

```java
setBackground(Color.yellow);
setForeground(Color.red);
```

Graphics

- Graphics Coordinate System
- Font and FontMetrics
- Drawing Lines
- Drawing Rectangles
- Drawing Ovals
- Drawing Arcs
- Drawing Polygons

Graphics Coordinate System

```
(0,0) y
\arrow v \arrow (120,100)
```

Fonts

```
Font myFont = Font(name, style, size);

Example:

Font myFont = new Font("TimesRoman", Font.BOLD, 16);
Font myFont = new Font("Courier", Font.BOLD+Font.ITALIC, 12);
```
Font Example

```java
public void paint(Graphics g)
{
    Font myFont = new Font("Times", Font.BOLD, 16);
    g.setFont(myFont);
    g.drawString("Welcome to Java", 20, 40);

    // set a new font
    g.setFont(new Font("Courier", Font.BOLD+Font.ITALIC, 12));
    g.drawString("Welcome to Java", 20, 70);
}
```

The FontMetrics Class

```
public void paint(Graphics g)
{
    Font myFont = new Font("Times", Font.BOLD, 16);
    g.setFont(myFont);
    g.drawString("Welcome to Java", 20, 40);

    // set a new font
    g.setFont(new Font("Courier", Font.BOLD+Font.ITALIC, 12));
    g.drawString("Welcome to Java", 20, 70);
}
```

Get FontMetrics

- `g.getFontMetrics(Font f);`
- `g.getFontMetrics();`

- `public int getAscent();`
- `public int getDescent();`
- `public int getLeading();`
- `public int getHeight();`
- `public int stringWidth(String str);`

Example 8.9

Using FontMetrics

- Objective: Display “Welcome to Java” in Helvetica 20-point bold, centered in the frame.

Run TestFontMetrics
Drawing Lines

drawLine(x1, y1, x2, y2);

Drawing Rectangles

drawRect(x, y, w, h);
fillRect(x, y, w, h);

Drawing Rounded Rectangles

drawRoundRect(x, y, w, h, aw, ah);
fillRoundRect(x, y, w, h, aw, ah);

Drawing Ovals

drawOval(x, y, w, h);
fillOval(x, y, w, h);
Drawing Arcs

- `drawArc(x, y, w, h, angle1, angle2);`
- `fillArc(x, y, w, h, angle1, angle2);`

Example 8.10
Drawing a Clock

- Objective: Use drawing and trigonometric methods to draw a clock showing the current time in a frame.