GUI GRAPHICS

Tanjina Helaly

Graphics

- All graphics are drawn relative to a window.
- The origin of each window is at the top-left corner and is 0,0.
- Coordinates are specified in pixels.
- A graphics context is encapsulated by the **Graphics class.**
- **Two ways to obtain a** graphics.
  - It is passed to a method, such as **paint( ) or update( ), as an argument.**
  - It is returned by the **getGraphics( ) method of Component.**
- **Graphics class defines a number of methods that draw various** types of objects, such as **lines**, **rectangles**, and **arcs**.
- Objects can be drawn **edge-only** or **filled**.
- Objects are drawn in the **currently selected color**, which is **black** by default.

Drawing Lines

- Lines are drawn by means of the **drawLine( )** method, shown here:
  - void drawLine(int *startX*, int *startY*, int *endX*, int *endY*)

Drawing Rectangles

- The **drawRect( )** and **fillRect( )** methods display an outlined and filled rectangle,
  - void drawRect(int *left*, int *top*, int *width*, int *height*)
  - void fillRect(int *left*, int *top*, int *width*, int *height*)
  - void clearRect(int *left*, int *top*, int *width*, int *height*)

Drawing Ellipses and Circles

- To draw an ellipse, use **drawOval( )**. To fill an ellipse, use **fillOval( )**.
  - void drawOval(int *left*, int *top*, int *width*, int *height*)
  - void fillOval(int *left*, int *top*, int *width*, int *height*)
- The ellipse is drawn within a bounding rectangle whose upper-left corner is specified by *left, top* and whose width and height are specified by *width* and *height*.
- To draw a circle, specify a square as the bounding rectangle.

Working with Color

- We need different color to
  - set the background/foreground of a component.
    - Example:
      - f.setBackground(Color.***CYAN);***
      - f.setForeground(Color.***BLACK);***
  - paint different shapes.
    - Example: the line and oval will be drawn in red color

```
g.setColor(Color.RED);
g.drawLine(20, 20, 100, 100);
g.fillOval(10, 10, 30, 40);
```

- To specify a color we can
  - create your own color in the following form
    - Color(int *red*, int *green*, int *blue*)
      - These values must be between 0 and 255
      - Example: new Color(255, 100, 100); // light red
  - Or, the constant defines in the **Color class.**

Example code – using getGraphics() – Unreliable – need better understanding of Graphics

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

public class Test {
	JFrame f;
	public Test(){
		f = new JFrame("Graphics");
		f.setSize(400, 400);
		f.setBackground(Color.WHITE);
		f.setVisible(true);
		f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	public void show(){
		Graphics g = f.getGraphics();
		g.setColor(Color.RED);
		g.fillRect(20, 20, 100, 100);
		f.setVisible(true);
	}

	public static void main(String[] args) {
	      Test t = new Test();
	       t.show();
	}
}
```

Frame’s paint method

- a program should place the component's rendering code inside a particular overridden method, and the toolkit will invoke this method when it's time to paint.
- The method to be overridden is in java.awt.Component:

```
        public void paint(Graphics g)
```

- The paint() method is called when window needs to be drawn. Do not call paint() directly. Becareful!
- When you need to call/invoke paint(), call the repaint method instead.

Example code –using paint() method

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

public class Test extends JFrame{
    public Test(){
        super("Graphics");
        setSize(300, 200);
        setBackground(Color.WHITE);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void paint(Graphics g){
        g.setColor(Color.RED);
        g.fillRect(100, 80, 100, 50);
    }

    public static void main(String[] args) {
       new Test();
    }
}
```

Output