Outline

• Graphics
• Strings, Lines, Rectangles, Ovals
• Arcs, Polygons, Polylines
• Program
Graphics Overview

- Each GUI component has a graphics context
- The coordinate system starts at (0, 0) in the top left corner and increases positive to the right in the x direction and down in the y direction

- The upper left corner of a component is relative to its parent component
The `Graphics` class provides methods for drawing strings, lines, rectangles, ovals, arcs, polygons and polylines.

The `Graphics` object displays figures and images on the screen on different platforms.

`Graphics` is an abstract class that is automatically created by the JVM when a component is displayed.

- This object is passed into the `paintComponent(Graphics)` method of the component to display the drawing.
- The `paintComponent(Graphics)` method is defined in the `JComponent` class and is invoked whenever a component is first displayed or redisplayed.
- If you override the `paintComponent(Graphics)` method, you can utilize all of the methods available in the `Graphics` class.
### Graphics API

#### `java.awt.Graphics`

- `setColor(color: Color): void`
- `setFont(font: Font): void`
- `drawString(s: String, x: int, y: int): void`
- `drawLine(x1: int, y1: int, x2: int, y2: int): void`
- `drawRect(x: int, y: int, w: int, h: int): void`
- `fillRect(x: int, y: int, w: int, h: int): void`
- `drawRoundRect(x: int, y: int, w: int, h: int, ar: int, ah: int): void`
- `fillRoundRect(x: int, y: int, w: int, h: int, ar: int, ah: int): void`
- `draw3DRect(x: int, y: int, w: int, h: int, raised: boolean): void`
- `fill3DRect(x: int, y: int, w: int, h: int, raised: boolean): void`
- `drawOval(x: int, y: int, w: int, h: int): void`
- `fillOval(x: int, y: int, w: int, h: int): void`
- `drawArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void`
- `fillArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void`
- `drawPolygon(xPoints: int[], yPoints: int[], nPoints: int): void`
- `fillPolygon(xPoints: int[], yPoints: int[], nPoints: int): void`
- `drawPolygon(g: Polygon): void`
- `fillPolygon(g: Polygon): void`
- `drawPolyline(xPoints: int[], yPoints: int[], nPoints: int): void`

Sets a new color for subsequent drawings.
Sets a new font for subsequent drawings.
Draws a string starting at point `(x, y)`.
Draws a line from `(x1, y1)` to `(x2, y2)`.
Draws a rectangle with specified upper-left corner point at `(x, y)` and width `w` and height `h`.
Draws a filled rectangle with specified upper-left corner point at `(x, y)` and width `w` and height `h`.
Draws a round-cornered rectangle with specified arc width `aw` and arc height `ah`.
Draws a filled round-cornered rectangle with specified arc width `aw` and arc height `ah`.
Draws a 3-D rectangle raised above the surface or sunk into the surface.
Draws a filled 3-D rectangle raised above the surface or sunk into the surface.
Draws an oval bounded by the rectangle specified by the parameters `x`, `y`, `w`, and `h`.
Draws a filled oval bounded by the rectangle specified by the parameters `x`, `y`, `w`, and `h`.
Draws an arc conceived as part of an oval bounded by the rectangle specified by the parameters `x`, `y`, `w`, and `h`.
Draws a filled arc conceived as part of an oval bounded by the rectangle specified by the parameters `x`, `y`, `w`, and `h`.
Draws a closed polygon defined by arrays of `x`- and `y`-coordinates. Each pair of `(x[i], y[i])`-coordinates is a point.
Draws a filled polygon defined by arrays of `x`- and `y`-coordinates. Each pair of `(x[i], y[i])`-coordinates is a point.
Draws a closed polygon defined by a `Polygon` object.
Draws a filled polygon defined by a `Polygon` object.
Draws a polyline defined by arrays of `x`- and `y`-coordinates. Each pair of `(x[i], y[i])`-coordinates is a point.
Outline

• Graphics
• Strings, Lines, Rectangles, Ovals
• Arcs, Polygons, Polylines
• Program
Drawing Strings

- To draw items on a component, create a class that extends the `JPanel` class and overrides the `paintComponent(Graphics)` method.

- The `Graphics` class defines a method `drawString(String, int x, int y)`.
  - The string passed into the method is drawn at location (x, y) of the parent component.
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Test extends JFrame {
    public Test() {
        super("Graphics Example");
        add(new NewPanel());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(250, 225);
        setLocationRelativeTo(null);
        setVisible(true);
    }

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

class NewPanel extends JPanel {
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawString("CSCI 201", 0, 40);
    }
}
The Graphics class defines a method `drawLine(int x1, int y1, int x2, int y2)`

A straight line is drawn from location \((x1, y1)\) of the parent component to \((x2, y2)\) of the parent component.
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Test extends JFrame {
    public Test() {
        super("Graphics Example");
        add(new NewPanel());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(250, 225);
        setLocationRelativeTo(null);
        setVisible(true);
    }

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

class NewPanel extends JPanel {
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawLine(0, 0, 100, 50);
    }
}
The `Graphics` class defines two methods for drawing regular rectangles:

- `drawRect(int x, int y, int w, int h)`
- `fillRect(int x, int y, int w, int h)`

![Example of drawing rectangles](image1)

![Example of filling rectangles](image2)
The **Graphics** class defines two methods for drawing rounded rectangles:

- `drawRoundRect(int x, int y, int w, int h, int aw, int ah)`
  - `aw` is the horizontal diameter of the arcs at the corner
  - `ah` is the vertical diameter of the arcs at the corner

- `fillRoundRect(int x, int y, int w, int h, int aw, int ah)`
The `Graphics` class defines two methods for drawing 3D rectangles:

- `draw3DRect(int x, int y, int w, int h, boolean raised)`
- `fill3DRect(int x, int y, int w, int h, boolean raised)`

The third dimension is not very visible though.

Raised = true

Raised = false
Drawing Ovals

- The **Graphics** class defines two methods for drawing ovals
  
  - `drawOval(int x, int y, int w, int h)`
  
  - `fillOval(int x, int y, int w, int h)`
Outline

• Graphics
• Strings, Lines, Rectangles, Ovals
• Arcs, Polygons, Polylines
• Program
Drawing Arcs

- An arc is part of an oval bounded by a rectangle
- The **Graphics** class defines two methods for drawing arcs
  - `drawArc(int x, int y, int w, int h, int startAngle, int arcAngle)`
  - `fillArc(int x, int y, int w, int h, int startAngle, int arcAngle)`
Drawing Polygons

- A polygon is a closed two-dimensional region bounded by an arbitrary number of edges.
- A polygon consists of a number of \((x, y)\) coordinates that define the vertices of the polygon.
  - Two successive points are the endpoints of a line that is a side of the polygon.
  - The first and last points are joined by a line segment.
- There are two ways to create polygons:
  - Create a \texttt{Polygon} object.
  - Draw the polygon with an array of \(x\) and an array of \(y\) values.
import java.awt.Graphics;
import java.awt.Polygon;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Test extends JFrame {
    public Test() {
        super("Graphics Example");
        add(new NewPanel());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(250, 225);
        setLocationRelativeTo(null);
        setVisible(true);
    }

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

    class NewPanel extends JPanel {
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Polygon poly = new Polygon();
            poly.addPoint(110, 10);
            poly.addPoint(160, 10);
            poly.addPoint(210, 60);
            poly.addPoint(210, 110);
            poly.addPoint(160, 160);
            poly.addPoint(110, 160);
            poly.addPoint(60, 110);
            poly.addPoint(60, 60);
            g.drawPolygon(poly);
        }
    }
}
import java.awt.Graphics;
import java.awt.Polygon;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Test extends JFrame {
    public Test() {
        super("Graphics Example");
        add(new NewPanel());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(250, 225);
        setLocationRelativeTo(null);
        setVisible(true);
    }

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

class NewPanel extends JPanel {
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        int x[] = {110, 160, 210, 210, 160, 110, 60, 60};
        int y[] = {10, 10, 60, 110, 160, 160, 110, 60};
        g.drawPolygon(x, y, x.length);
    }
}
Drawing Polylines

- A polyline is an open two-dimensional region bounded by an arbitrary number of edges.
- A polyline does not join the first and last points as a polygon does.
- There is only one way to create a polyline:
  - Draw the polyline with an array of x and an array of y values.
Drawing Polylines Example

```java
import java.awt.Graphics;
import java.awt.Polygon;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Test extends JFrame {
    public Test() {
        super("Graphics Example");
        add(new NewPanel());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(250, 225);
        setLocationRelativeTo(null);
        setVisible(true);
    }

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

class NewPanel extends JPanel {
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        int x[] = {110, 160, 210, 210, 160, 110, 60, 60};
        int y[] = {10, 10, 60, 110, 160, 160, 110, 60};
        g.drawPolyline(x, y, x.length);
    }
```
Outline

• Graphics
• Strings, Lines, Rectangles, Ovals
• Arcs, Polygons, Polylines
• Program
Create the following GUI. Make the clock functional to update the arms and the time every second. Look up the `Thread.sleep(1000)` method to make your program more efficient than using busy waiting.