More Java GUI and graphics

Horstmann Chapter 4.8-10

CS 4354
Summer II 2016

Jill Seaman

Goal: Animated Car Icon

- We will use some GUI and graphics classes to animate a car icon in this lecture.
  - ✦Using classes we already know along with some new ones.

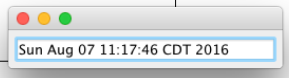
4.8 Timers

- The javax.swing.Timer class generates a sequence of action events, spaced apart at equal time intervals, and notifies a designated action listener.

```
ActionListener listener = ...;
final int DELAY = 1000; // 1000 millisec = 1 sec
Timer t = new Timer(DELAY, listener);
t.start();
```

- For example, use a Timer to display a digital clock:

```
ActionListener listener = new
   ActionListener() {
      public void actionPerformed(ActionEvent event) {
         Date now = new Date();
         textFieldsetText(now.toString());
      }
};
Timer t = new Timer(DELAY, listener);
```



4.9 Drawing Shapes

- Recall the Icon Interface:
  - ✦the `paintIcon` method receives a graphics context of type `Graphics`
- Actual object passed is a Graphics2D object in modern Java versions, so we go ahead and cast it:

```
public void paintIcon(Component c, Graphics g, int x, int y) {
 Graphics2D g2 = (Graphics2D)g;
 . . .
}
```

- The Graphics object is a graphics context.
- It can draw any object that implements the Shape interface

```
Shape s = . . .;
g2.draw(s);
```

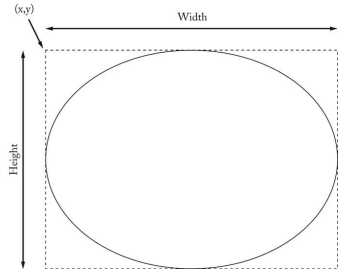
Drawing Rectangles and Ellipses

- The Java library supplies a number of classes that implement the Shape interface type.
- To construct and draw a Rectangle2D.Double object, specify:
  - ✦the top left corner, width, and height

```
Shape rectangle = new Rectangle2D.Double(x, y, width, height);
g2.draw(rectangle);
```

- For Ellipse2D.Double, width and height specify the bounding box:

```
Shape ellipse =
   new Ellipse2D.Double
   (x, y, width, height);
g2.draw(ellipse);
```



Drawing Line Segments, and filling shapes

- Point2D.Double is a point in the plane
- Line2D.Double joins two points

```
Point2D.Double start = new Point2D.Double(x1, y1);
Point2D.Double end = new Point2D.Double(x2, y2);
Shape segment = new Line2D.Double(start, end);
g2.draw(segment);
```

- You can also fill a shape instead of drawing the outline:

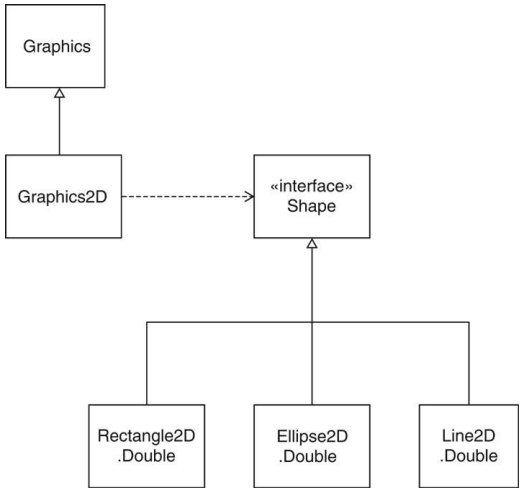
```
g2.fill(ellipse);
```

  fills the inside of the ellipse with the *current* color.

- To change the color, make a call such as this first:

```
g2.setColor(Color.RED);
```


Relationships between Shapes classes



Drawing the car icon


```
public class CarIcon implements Icon
{
  private int width;

  public CarIcon(int aWidth)
   {
      width = aWidth;
   }

   public int getIconWidth()
   {
      return width;
   }

   public int getIconHeight()
   {
      return width / 2;
   }
```

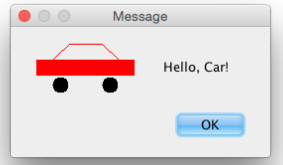
Drawing the car icon


```
public void paintIcon(Component c, Graphics g, int x, int y) {

   Graphics2D g2 = (Graphics2D) g;

   //These are the three shapes:
   Rectangle2D.Double body
     = new Rectangle2D.Double(x, y + width/6, width - 1, width/6);
   Ellipse2D.Double frontTire
     = new Ellipse2D.Double(x + width/6, y + width/3, width/6, width/6);
   Ellipse2D.Double rearTire
     = new Ellipse2D.Double(x + width * 2/3, y + width/3, width/6, width/6);

   // These are the four points needed to draw the three lines:
   // The bottom of the front windshield
   Point2D.Double r1  = new Point2D.Double(x + width/6, y + width/6);
   // The front of the roof
   Point2D.Double r2 = new Point2D.Double(x + width/3, y);
   // The rear of the roof
   Point2D.Double r3 = new Point2D.Double(x + width*2/3, y);
   // The bottom of the rear windshield
   Point2D.Double r4 = new Point2D.Double(x + width*5/6, y + width/6);
```


Drawing the car icon


```
   //These are the three lines:
   Line2D.Double frontWindshield = new Line2D.Double(r1, r2);
   Line2D.Double roofTop = new Line2D.Double(r2, r3);
   Line2D.Double rearWindshield = new Line2D.Double(r3, r4);

   // Now to fill, color, and draw using the graphics context:
   g2.fill(frontTire);
   g2.fill(rearTire);
   g2.setColor(Color.red);
   g2.fill(body);
   g2.draw(frontWindshield);
   g2.draw(roofTop);
   g2.draw(rearWindshield);
}
```

```
public static void main(String[] args) {
   JOptionPane.showMessageDialog(
       null,
       "Hello, Car!",
       "Message",
       JOptionPane.INFORMATION_MESSAGE,
       new CarIcon(100));
 System.exit(0);
}
```



4.10 Designing an Interface Type

- Now we'll use a timer to move car shapes
- Ten times per second, the car shape will move and the window will be repainted so that the new position is displayed.
- There are two responsibilities:
  - ✦ Draw shape
  - ✦ Move shape
- Define new interface type MoveableShape (so we can animate any shape that provides these two operations):

```
public interface MoveableShape
{
   void draw(Graphics2D g2);
   void translate(int dx, int dy);
}
```

Methods are named to conform to standard library names

The CarShape class

```
public class CarShape implements MoveableShape {
   private int x;
   private int y;
   private int width;

   /** Constructs a car item.
      @param x the left of the bounding rectangle
      @param y the top of the bounding rectangle
      @param width the width of the bounding rectangle
   */
   public CarShape(int x, int y, int width) {
      this.x = x;
      this.y = y;
      this.width = width;
   }
   public void translate(int dx, int dy) {
      x += dx;
      y += dy;
   }
   public void draw(Graphics2D g2) {
      //insert code from CarIcon.paintIcon here
   }
```

Implementing the Animation

- The Moveable shape draws and moves a shape
- We want to put it into a JFrame, which needs a JComponent
- So we'll make a ShapeAdapter class (like the IconAdapter)
- ShapeAdapter.paintComponent calls MoveableShape.draw

- Then the Timer action moves shape, calls repaint on ShapeAdapter

The ShapeAdapter class

```
public class ShapeAdapter extends JComponent {
    private int width;
    private int height;
    private MoveableShape shape;
   /** Constructs a JComponent that displays a given MoveableShape.
       @param mShape the shape to display
       @param width
       @param height
   */
   public ShapeAdapter(MoveableShape mShape, int width, int height) {
        this.shape = mShape;
        this.width = width;
        this.height = height;
   }
   @Override
   public void paintComponent(Graphics g) {
       Graphics2D g2 = (Graphics2D) g;
       shape.draw(g2);
   }
   @Override
   public Dimension getPreferredSize() {
      return new Dimension(width, height);
   }
}
```


The AnimationTester class

```
public class AnimationTester {
   public static void main(String[] args) {
     JFrame frame = new JFrame();
     final MoveableShape shape = new CarShape(0, 0, CAR_WIDTH);
     JComponent component = new ShapeAdapter(shape, ICON_WIDTH, ICON_HEIGHT);
     frame.setLayout(new FlowLayout());
     frame.add(component);
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.pack();
     frame.setVisible(true);

     final int DELAY = 100;     // Milliseconds between timer ticks
     Timer t = new Timer(DELAY, new
         ActionListener() {
            public void actionPerformed(ActionEvent event) {
               shape.translate(1, 0);  // increment x by 1
               component.repaint();    // repaint the Shape
            }
         });
     t.start();
   }
   private static final int ICON_WIDTH = 400;
   private static final int ICON_HEIGHT = 100;
   private static final int CAR_WIDTH = 100;
}
```


Classes Involved in the Car Animation

- Note the CarShape can easily be replaced by any MoveableShape