CS1007: Object Oriented Design and Programming in Java

Lecture #9
Oct 6

Shlomo Hershkop
shlomo@cs.columbia.edu

Outline

- Polymorphism
- Basic graphics
- Layout managers
- Customization

- Reading: 4.5 - 4.8, 4.9

Annoucements

- Homework 2 due Oct 17
  - Hint: Start early
- Midterm October 20

Anonymous Classes

- No need to name objects that are used only once

```
Collections.sort(countries,
   new CountryComparatorByName());
```

- No need to name classes that are used only once

```
Comparator<Country> comp = new
   Comparator<Country>()
   {
      public int compare(Country country1, Country country2)
      {
         return country1.getName().compareTo(country2.getName());
      }
   };
```

- anonymous new expression:
  - defines anonymous class that implements Comparator
  - defines compare method of that class
  - constructs one object of that class
- Cryptic syntax for very useful feature

Anonymous Classes

- Commonly used in factory methods:

```
public static Comparator<Country> comparatorByName()
{
   return new Comparator<Country>()
   {
      public int compare(Country country1, Country
  country2)

      { . . . }
   };
}
```

- Collections.sort(a, Country.comparatorByName());
- Neat arrangement if multiple comparators make sense (by name, by area, ...)

Example

```
modeling.addActionListener(
  new ActionListener() {
     public void actionPerformed(ActionEvent evt) {
               try {
                   if (update_all_labels == true) {
```


`public interface ActionListener extends EventListener`

- The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked.

`void actionPerformed(ActionEvent e)`

Java Frame Class

- Frame window has decorations
  - title bar
  - close box
  - provided by windowing system

```
JFrame frame = new JFrame();
frame.pack();
frame.setDefaultCloseOperation(JFrame
  .EXIT_ON_CLOSE);
frame.setVisible(true);
```

9

Layout managers

- User interfaces made up of components
- Components placed in containers
- Container needs to arrange components
- Swing doesn't use hard-coded pixel coordinates
- Advantages:
  - Can switch "look and feel"
  - Can internationalize strings
- Layout manager controls arrangement
- Will return to this next class

10

Adding components

- Construct components

```
JButton helloButton = new JButton("Say Hello");
```

- Set frame layout

```
frame.setLayout(new FlowLayout());
```

- Add components to frame

```
frame.add(helloButton);
```

11

```
09    JFrame frame = new JFrame();
10:   JButton helloButton = new JButton("Say Hello");
11:   JButton goodbyeButton = new JButton("Say Goodbye");
12:
13:   final int FIELD_WIDTH = 20;
14:   JTextField textField = new JTextField(FIELD_WIDTH);
15:   textField.setText("Click a button!");
16:
17:   frame.setLayout(new FlowLayout());
18:
19:   frame.add(helloButton);
20:   frame.add(goodbyeButton);
21:   frame.add(textField);
22:
23:   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24:   frame.pack();
25:   frame.setVisible(true);
```

12

Reacting to user input

- Previous code buttons don't have any effect
- Add listener object(s) to button
- Belong to class implementing ActionListener interface type

```
public interface ActionListener
{
   int actionPerformed(ActionEvent event);
}
```

- Listeners are notified when button is clicked

13

Actions

- Add action code into actionPerformed method
- Gloss over routine code

```
helloButton.addActionListener(new
   ActionListener()
   {
      public void actionPerformed(ActionEvent event)
      {
          textField.setText("Hello, World");
     }
   });
```

- When button is clicked, text field is set

14

Back to scopes

- Remarkable: Inner class can access variables from enclosing scope
  e.g. textField
- Can access enclosing instance fields, local variables
- Local variables must be marked final

```
final JTextField textField = ...;
```

15

How to react

- Constructor attaches listener:

```
helloButton.addActionListener(listener);
```

- Button remembers all listeners
- When button clicked, button notifies listeners

```
listener.actionPerformed(event);
```

- Listener sets text of text field

```
textField.setText("Hello, World!");
```

16

Related work

- Write helper method that constructs objects
- Pass variable information as parameters
- Declare parameters final

```
public static ActionListener createGreetingButtonListener(
   final String message)
{
 return new
    ActionListener()
    {
       public void actionPerformed(ActionEvent event)
       {
          textField.setText(message);
       }
    };
}
```

17

Java Timers

- Supply delay, action listener

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

- Action listener called when delay elapsed

18

Code

```
22:       ActionListener listener = new
23:          ActionListener()
24:          {
25:             public void actionPerformed(ActionEvent
   event)
26:             {
27:                Date now = new Date();
28:                textField.setText(now.toString());
29:             }
30:          };
31:       final int DELAY = 1000;
32:          // Milliseconds between timer ticks
33:       Timer t = new Timer(DELAY, listener);
34:       t.start();
```

19

Be careful

- java.util.Timer
- java.swing.Timer

- How does java figure out which class Timer you are referring to??

20

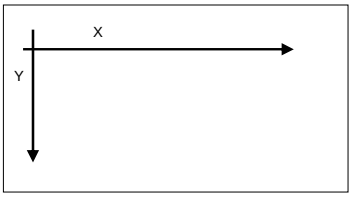
Java Drawing

- All drawing done on graphic context objects
- Area to draw on
- It is instantiated to the paint() or update() method

21

Coordinate System

- `(0,0)` on top left corner
- X increase to right
- Y increase downwards
- `(0,object.getwidth() )` is upper right corner



22

Drawing Shapes

- paintIcon method receives graphics context of type Graphics
- Actually a Graphics2D object in modern Java versions

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

- Can draw any object that implements Shape interface

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

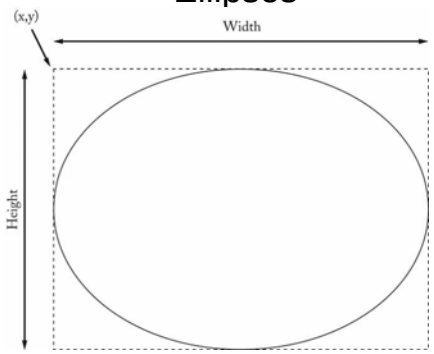
23

Rectangles & Ellipses

- Rectangle2D.Double constructed with
  - top left corner
  - width
  - height
- g2.draw(new Rectangle2D.Double(x, y, width, height));
- For Ellipse2D.Double, specify bounding box

24

Ellipses



25

Shape rectangle = new
rectangle2D.Double(x, y, width, height);

g2.draw(rectangle);

26

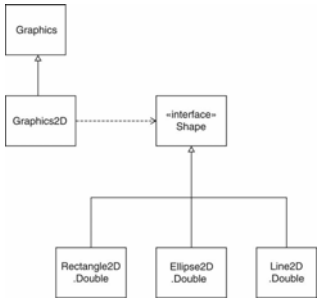
Line Segments

- `Point2D.Double` is a point in the plane
- `Line2D.Double` joins to 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);
```

27

Relationships between shape classes



28

Drawing Text

- `g2.drawString(text, x, y);`
- `x, y` are base point coordinates

29

Filling Shapes

- Fill interior of shape

`g2.fill(shape);`

- Set color for fills or strokes:

`g2.setColor(Color.red);`

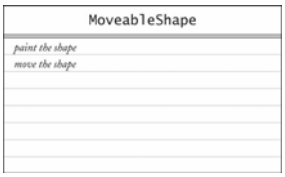
30

Defining a New Interface Type

- Use timer to move car shapes
- Draw car with CarShape
- Two responsibilities:
  - Draw shape
  - Move shape
- Define new interface type MoveableShape

31

CRC



32

- Name the methods to conform to standard library

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

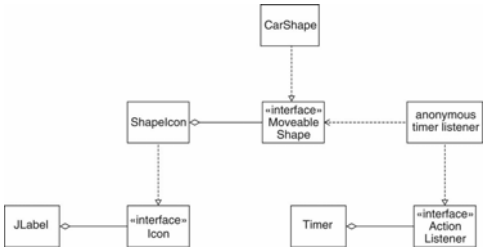
- CarShape class implements MoveableShape

```
public class CarShape implements MoveableShape
{
   public void translate(int dx, int dy)
   { x += dx; y += dy; }
   . . .
}
```


Implimenting an Animation

- Label contains icon that draws shape
- Timer action moves shape, calls repaint on label
- Label needs `Icon`, we have `MoveableShape`
- Supply ShapeIcon adapter class
- `ShapeIcon.paintIcon` calls `MoveableShape.draw`

- Ch4/animation/MoveableShape.java
- Ch4/animation/ShapeIcon.java
- Ch4/animation/AnimationTester.java
- Ch4/animation/CarShape.java



Next

- Do reading till chapter 5
- Start homework
- Will review for midterm, prepare questions

37