

Classes

Preparation

- Scene so far has been background material and experience
 - Computing systems and problem solving
 - Variables
 - Types
 - Input and output
 - Expressions
 - Assignments
 - Objects
 - Standard classes and methods

Ready

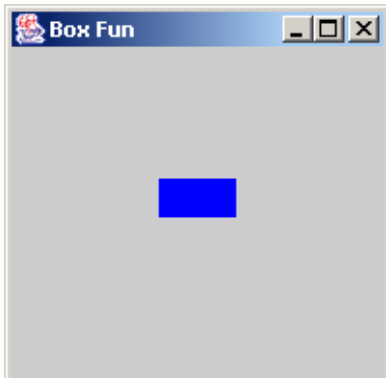
- Experience what Java is really about
 - Design and implement objects representing information and physical world objects

Object-oriented programming

- Basis
 - Create and manipulate objects with attributes and behaviors that the programmer can specify
- Mechanism
 - Classes
- Benefits
 - An information type is design and implemented once
 - Reused as needed
 - No need reanalysis and re-justification of the representation

First class – ColoredRectangle

- Purpose
 - Represent a colored rectangle in a window
 - Introduce the basics of object design and implementation



Background

- JFrame
 - Principal Java class for representing a titled, bordered graphical window.
 - Standard class
 - Part of the swing library

```
import javax.swing.*;
```

Example

- Consider

```
JFrame w1 = new JFrame("Bigger");
JFrame w2 = new JFrame("Smaller");
w1.setSize(200, 125);
w2.setSize(150, 100);
w1.setVisible(true);
w2.setVisible(true);
```

Example

- Consider

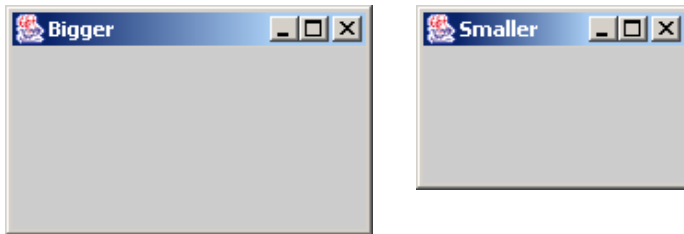
```
JFrame w1 = new JFrame("Bigger");
JFrame w2 = new JFrame("Smaller");
w1.setSize(200, 125);
w2.setSize(150, 100);
w1.setVisible(true);
w2.setVisible(true);
```



Example

- Consider

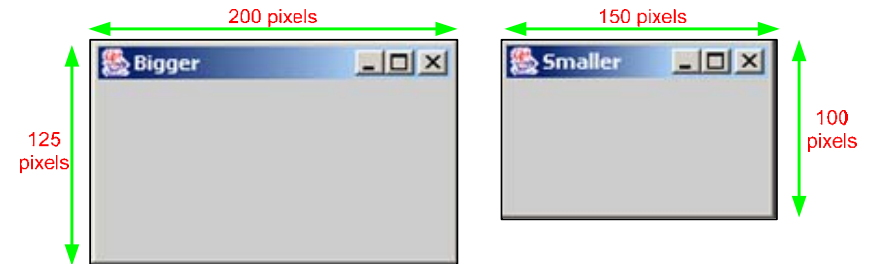
```
JFrame w1 = new JFrame("Bigger");  
JFrame w2 = new JFrame("Smaller");  
w1.setSize(200, 125);  
w2.setSize(150, 100);  
w1.setVisible(true);  
w2.setVisible(true);
```



Example

- Consider

```
JFrame w1 = new JFrame("Bigger");  
JFrame w2 = new JFrame("Smaller");  
w1.setSize(200, 125);  
w2.setSize(150, 100);  
w1.setVisible(true);  
w2.setVisible(true);
```



Class ColoredRectangle – initial version

- Purpose
 - Support the display of square window containing a blue filled-in rectangle
 - Window has side length of 200 pixels
 - Rectangle is 40 pixels wide and 20 pixels high
 - Upper left hand corner of rectangle is at (80, 90)
 - Limitations are temporary
 - Remember BMI.java preceded BMI Calculator.java
 - Lots of concepts to introduce

ColoredRectangle in action

- Consider

```
ColoredRectangle r1 = new ColoredRectangle();  
ColoredRectangle r2 = new ColoredRectangle();  
  
System.out.println("Enter when ready");  
System.in.read();  
  
r1.paint(); // draw the window associated with r1  
r2.paint(); // draw the window associated with r2
```

ColoredRectangle in action

- Consider

```
ColoredRectangle r1 = new ColoredRectangle();  
ColoredRectangle r2 = new ColoredRectangle();
```

```
System.out.println("Enter when ready");  
System.in.read();
```

```
r1.paint(); // draw the window associated with r1  
r2.paint(); // draw the window associated with r2
```

ColoredRectangle in action

- Consider

```
ColoredRectangle r1 = new ColoredRectangle();  
ColoredRectangle r2 = new ColoredRectangle();
```

```
System.out.println("Enter when ready");  
System.in.read();
```

```
r1.paint(); // draw the window associated with r1  
r2.paint(); // draw the window associated with r2
```

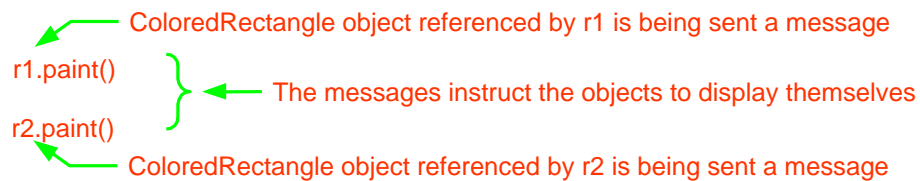
ColoredRectangle in action

- Consider

```
ColoredRectangle r1 = new ColoredRectangle();  
ColoredRectangle r2 = new ColoredRectangle();
```

```
System.out.println("Enter when ready");  
System.in.read();
```

```
r1.paint(); // draw the window associated with r1  
r2.paint(); // draw the window associated with r2
```



ColoredRectangle.java outline

```
import javax.swing.*;  
import java.awt.*;  
  
public class ColoredRectangle {  
    // instance variables for holding object attributes  
    private int width;  
    private int height;  
    private int x;  
    private int y;  
    private JFrame window;  
    private Color color;  
  
    // ColoredRectangle(): default constructor  
    public ColoredRectangle() { // ...  
    }  
  
    // paint(): display the rectangle in its window  
    public void paint() { // ...  
    }  
}
```

Instance variables and attributes

- Data field
 - Java term for an object attribute
- Instance variable
 - Symbolic name for a data field
 - Usually has private access
 - Assists in information hiding by encapsulating the object's attributes
 - Default initialization
 - Numeric instance variables initialized to 0
 - Logical instance variables initialized to false
 - Object instance variables initialized to null

```
public class ColoredRectangle {  
  
    // instance variables for holding object attributes  
    private int width;           private int x;  
    private int height;        private int y;  
    private JFrame window;     private Color color;  
  
    // ColoredRectangle(): default constructor  
    public ColoredRectangle() {  
        window = new JFrame("Box Fun");  
        window.setSize(200, 200);  
        width = 40;             x = 80;  
        height = 20;           y = 90;  
        color = Color.BLUE;  
        window.setVisible(true);  
    }  
  
    // paint(): display the rectangle in its window  
    public void paint() {  
        Graphics g = window.getGraphics();  
        g.setColor(color);  
        g.fillRect(x, y, width, height);  
    }  
}
```

```
public class ColoredRectangle {  
  
    // instance variables for holding object attributes  
    private int width;           private int x;  
    private int height;        private int y;  
    private JFrame window;     private Color color;  
  
    // ColoredRectangle(): default constructor  
    public ColoredRectangle() {  
        window = new JFrame("Box Fun");  
        window.setSize(200, 200);  
        width = 40;             x = 80;  
        height = 20;           y = 90;  
        color = Color.BLUE;  
        window.setVisible(true);  
    }  
  
    // paint(): display the rectangle in its window  
    public void paint() {  
        Graphics g = window.getGraphics();  
        g.setColor(color);  
        g.fillRect(x, y, width, height);  
    }  
}
```

ColoredRectangle default constructor

```
public class ColoredRectangle {  
    // instance variables to describe object attributes  
    ...  
  
    // ColoredRectangle(): default constructor  
    public ColoredRectangle() {  
        ...  
    }  
    ...  
}
```

The name of a constructor always matches the name of its class

A constructor does not list its return type. A constructor always returns a reference to a new object of its class

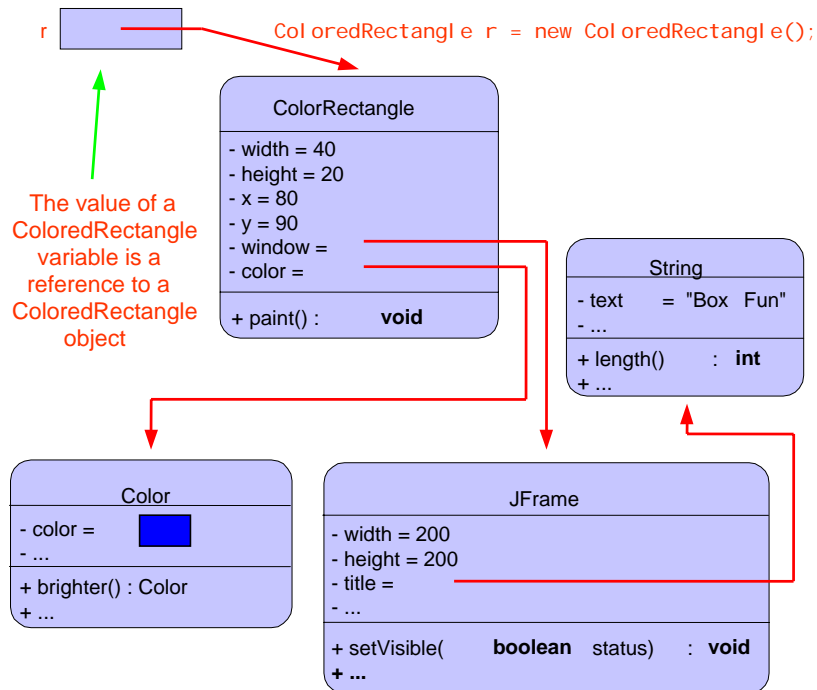
```

public class ColoredRectangle {
    // instance variables for holding object attributes
    private int width;           private int x;
    private int height;         private int y;
    private JFrame window;      private Color color;
    // ColoredRectangle(): default constructor
    public ColoredRectangle() {
        window = new JFrame("Box Fun");
        window.setSize(200, 200);
        width = 40;             x = 80;
        height = 20;           y = 90;
        color = Color.BLUE;
        window.setVisible(true);
    }
    // paint(): display the rectangle in its window
    public void paint() {
        Graphics g = window.getGraphics();
        g.setColor(color);
        g.fillRect(x, y, width, height);
    }
}

```

Color constants

- Color.BLACK
- Color.BLUE
- Color.CYAN
- Color.DARK_GRAY
- Color.GRAY
- Color.GREEN
- Color.LIGHT_GRAY
- Color.MAGENTA
- Color.ORANGE
- Color.PINK
- Color.RED
- Color.WHITE
- Color.YELLOW



```

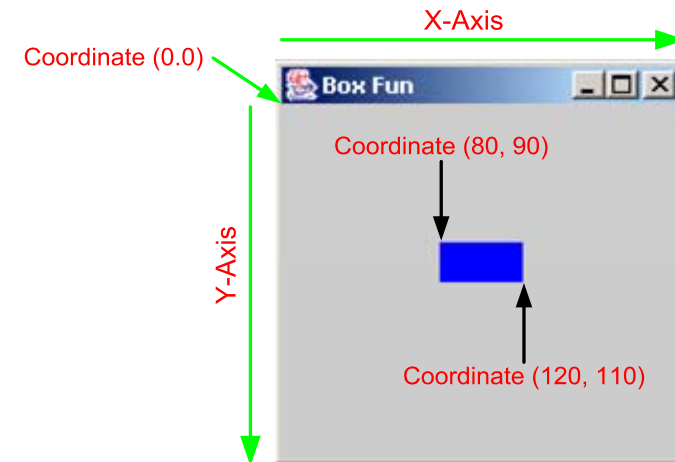
public class ColoredRectangle {
    // instance variables for holding object attributes
    private int width;           private int x;
    private int height;         private int y;
    private JFrame window;      private Color color;
    // ColoredRectangle(): default constructor
    public ColoredRectangle() {
        window = new JFrame("Box Fun");
        window.setSize(200, 200);
        width = 40;             x = 80;
        height = 20;           y = 90;
        color = Color.BLUE;
        window.setVisible(true);
    }
    // paint(): display the rectangle in its window
    public void paint() {
        Graphics g = window.getGraphics();
        g.setColor(color);
        g.fillRect(x, y, width, height);
    }
}

```

Graphical context

- Graphics
 - Defined in java.awt.Graphics
 - Represents the information for a rendering request
 - Color
 - Component
 - Font
 - ...
 - Provides methods
 - Text drawing
 - Line drawing
 - Shape drawing
 - Rectangles
 - Ovals
 - Polygons

Java coordinate system



```
public class ColoredRectangle {  
    // instance variables for holding object attributes  
    private int width;           private int x;  
    private int height;         private int y;  
    private JFrame window;       private Color color;  
    // ColoredRectangle(): default constructor  
    public ColoredRectangle() {  
        window = new JFrame("Box Fun");  
        window.setSize(200, 200);  
        width = 40;             x = 80;  
        height = 20;            y = 90;  
        color = Color.BLUE;  
        window.setVisible(true);  
    }  
    // paint(): display the rectangle in its window  
    public void paint() {  
        Graphics g = window.getGraphics();  
        g.setColor(color);  
        g.fillRect(x, y, width, height);  
    }  
}
```

Method invocation

- Consider
 - `r1.paint();` // display window associated with r1
 - `r2.paint();` // display window associated with r2
- Observe
 - When an instance method is being executed, the attributes of the object associated with the invocation are accessed and manipulated
 - Important that you understand what object is being manipulated

Method invocation

```
public class ColoredRectangle {
    // instance variables to describe object attributes
    ...
    // paint(): display the rectangle in its window
    public void paint() {
        window.setVisible( true );
        Graphics g = window.getGraphics();
        g.setColor(color);
        g.fillRect(x, y, width, height);
    }
    ...
}
```

The values of these instance variables are also from the ColoredRectangle object that invoked method paint().

Instance variable window references the JFrame attribute of the object that caused the invocation. That is, the invocation r1.paint() causes the window attribute of the ColoredRectangle referenced by r1 to be accessed. Similarly, the invocation r2.paint() causes the window attribute of the ColoredRectangle referenced by r2 to be accessed.

Improving ColoredRectangle

- Analysis
 - A ColoredRectangle object should
 - Be able to have any color
 - Be positionable anywhere within its window
 - Have no restrictions on its width and height
 - Accessible attributes
 - Updateable attributes

Improving ColoredRectangle

- Additional constructions and behaviors
 - Specific construction
 - Construct a rectangle representation using supplied values for its attributes
 - Accessors
 - Supply the values of the attributes
 - Individual methods for providing the width, height, x-coordinate position, y-coordinate position, color, or window of the associated rectangle
 - Mutators
 - Manage requests for changing attributes
 - Ensure objects always have sensible values
 - Individual methods for setting the width, height, x-coordinate position, y-coordinate position, color, or window of the associated rectangle to a given value

A mutator method

- Definition

```
// setWidth(): width mutator
public void setWidth(int w) {
    width = w;
}
```
- Usage

```
ColoredRectangle s = new ColoredRectangle();
s.setWidth(80);
```

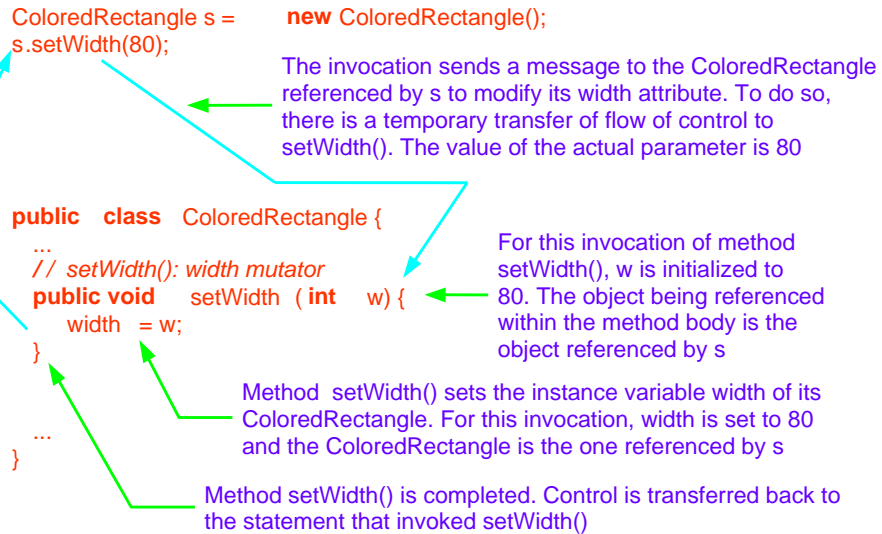
Object to be manipulated is the one referenced by s

```
public void setWidth (int w) {
    ...
}
```

Initial value of the formal parameter comes from the actual parameter

Changes to the formal parameter do not affect the actual parameter

Mutator setWidth() evaluation



Subtleties

- Consider


```

ColoredRectangle r = new ColoredRectangle();
r.paint();
r.setWidth(80);
r.paint();
            
```
- What is the width of the rectangle on the screen after the mutator executes?

Other mutators

```

public void setHeight(int h) {
height = h;
}

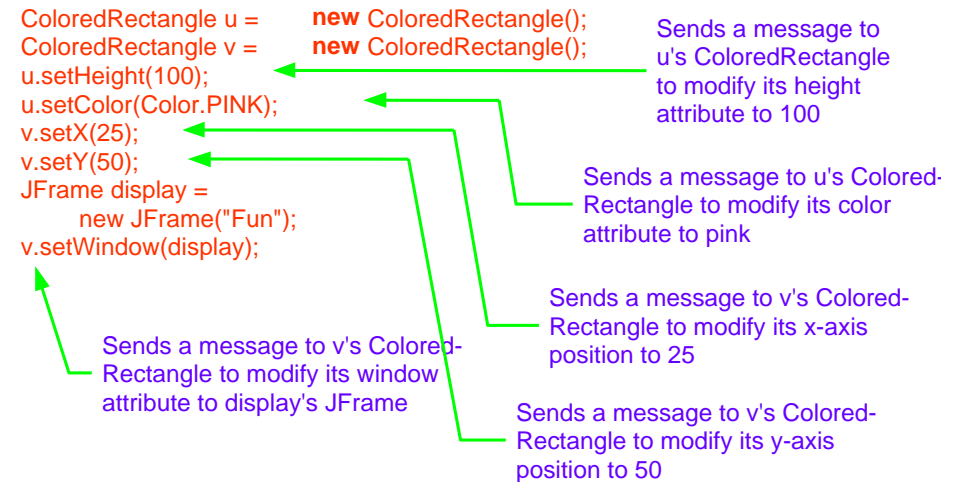
public void setX(int ux) {
x = ux;
}

public void setY(int uy) {
y = uy;
}

public void setWindow(JFrame f) {
window = f;
}

public void setColor(Color c) {
color = c;
}
    
```

Mutator usage



Accessors

- Properties
 - Do not require parameters
 - Each accessor execution produces a return value
 - Return value is the value of the invocation

```
public int getWidth() {
    return width;
}
```

The method return type precedes the name of the method in the method definition

For method getWidth(), the return value is the value of the width attribute for the ColoredRectangle associated with the invocation. In invocation t.getWidth(), the return value is the value of the instance variable width for the ColoredRectangle referenced by t

Accessor usage

```
ColoredRectangle t = new ColoredRectangle();
int w = t.getWidth();

public class ColoredRectangle {
    ...
    // getWidth(): accessor
    public int getWidth () {
        return width;
    }
    ...
}
```

Invocation sends a message to the ColoredRectangle referenced by t to return the value of its width. To do so, there is a temporary transfer of flow of control to getWidth()

Method getWidth() starts executing. For this invocation, the object being referenced is the object referenced by t

The return expression evaluates to 40 (the width attribute of the ColoredRectangle object referenced by t)

Method completes by supplying its return value (40) to the invoking statement. Also, invoking statement regains the flow of control. From there variable w is initialized with the return value of the invocation

Specific construction

```
public ColoredRectangle(int w, int h, int ulx, int uly,
                        JFrame f, Color c) {
    setWidth(w);
    setHeight(h);
    setX(ulx);
    setY(uly);
    setWindow(f);
    setColor(c);
}
```

- Requires values for each of the attributes


```
JFrame display = new JFrame("Even more fun");
display.setSize(400, 400);
ColoredRectangle w = new ColoredRectangle(60, 80,
20, 20, display, Color.YELLOW);
```

Specific construction

```
public ColoredRectangle(int w, int h, int ulx, int uly,
                        JFrame f, Color c) {
    setWidth(w);
    setHeight(h);
    setX(ulx);
    setY(uly);
    setWindow(f);
    setColor(c);
}
```

- Advantages to using mutators
 - Readability
 - Less error prone
 - Facilitates enhancements through localization

Seeing double

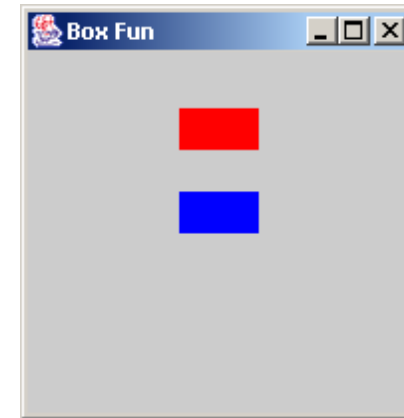
```
import java.io.*;
import java.awt.*;

public class SeeingDouble {
    public static void main(String[] args)
        throws IOException {
        ColoredRectangle r = new ColoredRectangle();
        System.out.println("Enter when ready");
        System.in.read();

        r.paint();

        r.setY(50);
        r.setColor(Color.RED);
        r.paint();
    }
}
```

Seeing double



Assignment (Due Feb 13, Tuesday, before noon)

- Explain the following terms:
Instance variable, instance method, default constructor, accessor method, mutator method, specific constructor.
- 4.24
- 4.25 (after you define this class Date, write another problem, which will use the class Date and do the following operations:
 1. Configure a new date "13 February 2007", output it.
 2. Change the date to "27 February 2007" using the setDay method, and output the following message:
"The first midterm will be on 27 February 2007", where the value of "27", "February" and "2007" should be obtained by getDay, getMonth and getYear methods.