Writing Classes – Part 2

CSC 1051 – Data Structures and Algorithms I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/
Getting to know classes so far

- Predefined classes from the Java API.
- We have defined a few classes of our own:
  - Account
  - Die
  - Book
  - Person
  - BigA, Big B, etc
Getting to know classes so far

- Predefined classes from the Java API.
- We have defined a few classes of our own:
  - Account
  - Die
  - Book
  - Person
  - BigA, Big B, etc

**Driver classes:**

- Transactions
- RollingDice
- Bookshelf
- (Quiz 7 – we did not write a driver)
- BigAlphabet, BigLettersGUI
## More Examples of Classes?

<table>
<thead>
<tr>
<th>Class</th>
<th>Attributes</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>Name</td>
<td>Set address</td>
</tr>
<tr>
<td></td>
<td>Address</td>
<td>Set major</td>
</tr>
<tr>
<td></td>
<td>Major</td>
<td>Compute grade point average</td>
</tr>
<tr>
<td></td>
<td>Grade point average</td>
<td></td>
</tr>
<tr>
<td>Rectangle</td>
<td>Length</td>
<td>Set length</td>
</tr>
<tr>
<td></td>
<td>Width</td>
<td>Set width</td>
</tr>
<tr>
<td></td>
<td>Color</td>
<td>Set color</td>
</tr>
<tr>
<td>Aquarium</td>
<td>Material</td>
<td>Set material</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Width</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Height</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute volume</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute filled weight</td>
</tr>
<tr>
<td>Flight</td>
<td>Airline</td>
<td>Set airline</td>
</tr>
<tr>
<td></td>
<td>Flight number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Origin city</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Destination city</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Current status</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set flight number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Determine status</td>
</tr>
<tr>
<td>Employee</td>
<td>Name</td>
<td>Set department</td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Title</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Salary</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute wages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute bonus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute taxes</td>
</tr>
</tbody>
</table>
Today

• Review what we learned so far

• UML diagrams

• Encapsulation

• Extending classes

• Graphical Objects
public class Transactions1
{
  //-----------------------------------------------------------------
  //  Creates some bank accounts and requests various services.
  //-----------------------------------------------------------------
  public static void main (String[] args)
  {
    Account acct1 = new Account ("Ted Murphy", 72354, 102.56);
    Account acct2 = new Account ("Jane Smith", 69713, 40.00);

    System.out.println (acct1);
    System.out.println (acct2);

    acct1.deposit (25.85);
    System.out.println (" ");
    System.out.println (acct2);
  }
}
Review

- **class declaration**

```java
long acctNumber;
double balance;
String name;

constructor

deposit()
withdraw()
toString()
```

**Data declarations**

**Method declarations**
**Method definition**
- parameters
- return type
- return statement

```java
char calc (int num1, int num2, String message) {
    int sum = num1 + num2;
    char result = message.charAt(sum);
    return result;
}
```

```java
ch = obj.calc(25, count, "Hello");
```
Review

• *using methods*...
Account class: Using methods

acct1.deposit (25.85);
Review

Account class: Using methods

acct1.deposit (25.85);

// ----------------------------------------
// Deposits the specified amount into the account.
// ----------------------------------------
public void deposit (double amount)
{
    balance = balance + amount;
}

CSC 1051 M.A. Papalaskari, Villanova University
acct1.deposit (25.85);

```java
public void deposit (double amount) {
    balance = balance + amount;
}
```

Review
 acct1.deposit (25.85);

---

public void deposit (double amount) {
    balance = balance + amount;
}
 acct1.deposit (25.85);

 acct1

 acctNumber 72354
 balance 102.56
 name "Ted Murphy"

 // Deposits the specified amount into the account.
 public void deposit (double amount)
 {
     balance = balance + amount;
 }
Account class: Another Example

```java
acct1.withdraw (60,2);
```

```plaintext
 acct1
 | acctNumber | 72354 |
 | balance    | 128.41 |
 | name       | "Ted Murphy" |
```
acct1.withdraw (60, 2);

```
public void withdraw (double amount, double fee) {
    balance = balance - amount - fee;
}
```

Review
acct1.withdraw (60, 2);

```
public void withdraw (double amount, double fee) {
    balance = balance - amount - fee;
}
```

 acctNumber: 72354
 balance: 66.41
 name: "Ted Murphy"
Account acct1 = new Account ("Ted Murphy", 72354, 102.56);

public Account (String owner, long account, double initial) {
    acctNumber = account;
    balance = initial;
    name = owner;
}
Review **toString() method**

```java
public String toString ()
{
    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    return (acctNumber + "\t" + name + "\t" + fmt.format(balance));
}
```

System.out.println(acct1);
Review static methods

OtherClass.doSomething(acct1);

public static void doSomething()
{
    System.out.println("At your service.");
}
What’s next?

- UML diagrams
- encapsulation and Java modifiers
- graphical objects
A UML class diagram for the `RollingDice` program:

- `RollingDice`:
  - `main (args : String[]) : void`

- `Die`:
  - `faceValue : int`
  - `roll() : int`
  - `setFaceValue (int value) : void`
  - `getFaceValue() : int`
  - `toString() : String`
Encapsulation

- An encapsulated object can be thought of as a *black box* -- its inner workings are hidden from the client.
- The client invokes the interface methods and they manage the instance data.
Violating Encapsulation

• It is possible for a class to access the instance data of another class directly
Violating Encapsulation - Example

- It is possible for a class to access the instance data of another class directly - but don’t do this!
- See Account.java (modified)
- See ImInUrClassMessingUrInstanceData.java

Account.java

Transactions.java

acct1.name = “Joe”;

deposit()
withdraw()
addInterest()

name
acctNumber
balance
Visibility Modifiers

• In Java, we enforce encapsulation through the appropriate use of *visibility modifiers*:
  
  – **public** – can be referenced from other classes
  
  – **private** – can be referenced only within that class:
  
  – **protected** – involves inheritance (discussed later)

• Data declared without a visibility modifier have *default visibility* and can be referenced by any class in the same package

• An overview of all Java modifiers is presented in Appendix E
public constants are ok - Example

Account acct1 = new Account ("Sartre", 72354, 102.56);

System.out.println (acct1);

System.out.println ("Interest rate = " + acct1.RATE);
## Visibility Modifiers – the **RULES**

<table>
<thead>
<tr>
<th></th>
<th>public</th>
<th>private</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variables</strong></td>
<td>NO (but OK for public constants)</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Methods</strong></td>
<td>Yes</td>
<td>Yes, for support methods only</td>
</tr>
</tbody>
</table>
Encapsulation – Accessing the data

- Indirect access through methods
- accessors and mutators ("getters" and "setters")
- Usually named getX() and setX()

**Example**

```java
acct1.getBalance()
```

**Transactions.java**

**Account.java**
Encapsulation – Accessing the data

- Mutators (setters) can restrict access to the data, as appropriate

- Example:

```java
public void setQuantity(int num) {
    if (num<0) {
        System.out.println("*Error in setQuantity()");
        System.out.println("negative quantity.");
        System.out.println("quantity not changed.");
    } else
        quantity = num;
}
```
Graphical Objects

- Some objects contain information that determines how the object should be represented visually.
- Most GUI components are graphical objects.
- We can have some effect on how components get drawn.
- We did this in Chapter 2 when we defined the `paint` method of an applet.
- Let's look at some other examples of graphical objects.
Smiling Face Example

• The SmilingFace program draws a face by defining the paintComponent method of a panel

• See SmilingFace.java

• See SmilingFacePanel.java

• The SmilingFacePanel class is derived from the JPanel class using inheritance
import javax.swing.JFrame;

public class SmilingFace
{
    //---
    // Creates the main frame of the program.
    //---
    public static void main (String[] args)
    {
        JFrame frame = new JFrame("Smiling Face");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        SmilingFacePanel panel = new SmilingFacePanel();
        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JFrame;

public class SmilingFace {

    //-------------------------------
    // Creates the main frame of the program.
    //-------------------------------
    public static void main (String[] args) {

        JFrame frame = new JFrame("Smiling Face");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        SmilingFacePanel panel = new SmilingFacePanel();

        frame.getContentPane().add(panel);

        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JPanel;
import java.awt.*;

public class SmilingFacePanel extends JPanel {
    private final int BASEX = 120, BASEY = 60; // base point for head

    public SmilingFacePanel () {
        setBackground (Color.blue);
        setPreferredSize (new Dimension(320, 200));
        setFont (new Font("Arial", Font.BOLD, 16));
    }
}

continue
public void paintComponent (Graphics page) {
    super.paintComponent (page);

    page.setColor (Color.yellow);
    page.fillOval (BASEX, BASEY, 80, 80); // head
    page.fillOval (BASEX-5, BASEY+20, 90, 40); // ears

    page.setColor (Color.black);
    page.drawOval (BASEX+20, BASEY+30, 15, 7); // eyes
    page.drawOval (BASEX+45, BASEY+30, 15, 7);

    page.fillOval (BASEX+25, BASEY+31, 5, 5); // pupils
    page.fillOval (BASEX+50, BASEY+31, 5, 5);

    page.drawArc (BASEX+20, BASEY+25, 15, 7, 0, 180); // eyebrows
    page.drawArc (BASEX+45, BASEY+25, 15, 7, 0, 180);

    page.drawArc (BASEX+35, BASEY+40, 15, 10, 180, 180); // nose
    page.drawArc (BASEX+20, BASEY+50, 40, 15, 180, 180); // mouth

    continue
continue

    page.setColor (Color.white);
    page.drawString ("Always remember that you are unique!",
                     BASEX-105, BASEY-15);
    page.drawString ("Just like everyone else.", BASEX-45, BASEY+105);
}
Jpanel Class – let’s look at Java API

public class JPanel
extends JComponent
implements Accessible

JPanel is a generic lightweight container. For examples and task-oriented documentation for JPanel, see...
Jpanel Class – let’s look at Java API
Smiling Face Example

- Every Swing component has a `paintComponent` method

- The `paintComponent` method accepts a `Graphics` object that represents the graphics context for the panel

- We define the `paintComponent` method to draw the face with appropriate calls to the `Graphics` methods

- Note the difference between drawing on a panel and adding other GUI components to a panel
Splat Example

- The Splat example is structured a bit differently
- It draws a set of colored circles on a panel, but each circle is represented as a separate object that maintains its own graphical information
- The `paintComponent` method of the panel "asks" each circle to draw itself

- See [Splat.java](#)
- See [SplatPanel.java](#)
- See [Circle.java](#)
//********************************************************************
//  Splat.java       Author: Lewis/Loftus
//
//  Demonstrates the use of graphical objects.
//********************************************************************

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

public class Splat
{
    //-----------------------------------------------------------------
    //  Presents a collection of circles.
    //-----------------------------------------------------------------
    public static void main (String[] args)
    {
        JFrame frame = new JFrame ("Splat");
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

        frame.getContentPane().add(new SplatPanel());

        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.*;  
import java.awt.*;  

public class Splat {  
  //-----------------  
  // Presents a collection of circles.  
  //-----------------  
  public static void main (String[] args) 
  {  
    JFrame frame = new JFrame("Splat");  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    frame.getContentPane().add(new SplatPanel());  
    frame.pack();  
    frame.setVisible(true);  
  }  
}
import javax.swing.*;
import java.awt.*;

public class SplatPanel extends JPanel {

  private Circle circle1, circle2, circle3, circle4, circle5;

  public SplatPanel() {
    circle1 = new Circle (30, Color.red, 70, 35);
    circle2 = new Circle (50, Color.green, 30, 20);
    circle3 = new Circle (100, Color.cyan, 60, 85);
    circle4 = new Circle (45, Color.yellow, 170, 30);
    circle5 = new Circle (60, Color.blue, 200, 60);

    setPreferredSize (new Dimension(300, 200));
    setBackground (Color.black);
  }
}

continue
continue

//-----------------------------------------------------------------------------
// Draws this panel by requesting that each circle draw itself.
//-----------------------------------------------------------------------------
public void paintComponent (Graphics page)
{
    super.paintComponent(page);

    circle1.draw(page);
    circle2.draw(page);
    circle3.draw(page);
    circle4.draw(page);
    circle5.draw(page);
}
}
import java.awt.*;

public class Circle {
    private int diameter, x, y;
    private Color color;

    public Circle (int size, Color shade, int upperX, int upperY) {
        diameter = size;
        color = shade;
        x = upperX;
        y = upperY;
    }
}

continue
continue

//Draws this circle in the specified graphics context.
public void draw (Graphics page)
{
    page.setColor (color);
    page.fillOval (x, y, diameter, diameter);
}

//Diameter mutator.
public void setDiameter (int size)
{
    diameter = size;
}

//Color mutator.
public void setColor (Color shade)
{
    color = shade;
}
continue
// X mutator.
public void setX (int upperX) {
    x = upperX;
}

// Y mutator.
public void setY (int upperY) {
    y = upperY;
}

// Diameter accessor.
public int getDiameter () {
    return diameter;
}
// Color accessor.
public Color getColor ()
{
    return color;
}

// X accessor.
public int getX ()
{
    return x;
}

// Y accessor.
public int getY ()
{
    return y;
}
Homework

• Review Sections 4.1- 4.6
  – Always do all self-review exercises when you review material

• Do Exercises EX 4.9 - 4.11

• Run all the programs to prepare for the lab Monday

• Read Sections 4.7- 4.9 to prepare for class Wednesday