User Interface Programming

OOP/Java Primer
Step 1
Object-Oriented Programming

- Focus on OOP rather than Java to start with
- What is an Object?
Object-Oriented Programming

- Focus on OOP rather than Java to start with
- What is an Object?

"I Exist! Therefore I am!"
Object

- An entity, a collection of data*
- A blueprint for the creation of INSTANCES

* And some other stuff too
Objects (instances)

- Knows "all" about themselves
  - Data
  - Available "messages"
  - What they need
Inheritance

- Objects can inherit from each other
  - Object hierarchies
  - X inherits from Y  ==> ”X is a type of Y”

- Inheritance is a strong programming concept
  - But can be overused
public class <Name>
{
    // class variables

    // instance variables

    // constructors

    // accessor methods

    // class methods
}
Concepts

- Class variables = common to all instances
- Instance variables = unique in every instance
- Constructor = Initializing method
- Accessor = returns contents of variables
  - Provides for encapsulation
- Methods = performs the ”real work”
class Ship {
    private int speed = 0;
    private int course = 0;

    public void setCourse(degree) { course = degree; }
    public int getCourse () { return course; }

    public void move() { … }
    public report() {
        System.out.println( "Course is " + course);
        …
    }
}
class ShipMain1 {
    public static void main(String[] args) {
        Ship argo = new Ship();
        argo.setCourse(90);
        argo.move();
        argo.report();
    }
}

// "setCourse" is an accessor method.
OOP vs. Procedures

- An object encapsulates a certain "aspect" of a system
- It defines what we may do with this "aspect"
- It has a limited scope
  - only works with its own responsibility
- It communicates with other objects to make things happen
Good programming practice

- Make small objects
  - Clear definition of what the object is.
- Make small methods
  - Clear definition of what the method does
- Give the object a simple structure
  - Don’t mix too much in a single object
- Use inheritance (with some constraints)
Private data

- Modifiers: ensure class integrity
- Private, Protected, Public – controls access
- Static – declares class variables and methods
- Final – declares constant values
Good Programming Practice

- Do not let other classes modify variable values directly!
- Use Accessor methods!

  - boat.speed = 10; // Bad!
  - boat.setSpeed(10); // Good!

- The accessor method can make integrity checks.
Packages

- A collection of classes that belong together
- A package defines a special part of the application
- Files within a package are stronger related to each other
Relations

- Knows about

- Has
  - contains other objects

- Is
  - Inheritance relation
Geometry

- Simple package to calculate geometric shapes
  - Cylinders, balls, pyramids, cones and boxes

- Simple arithmetic calculations

- In- and output
<table>
<thead>
<tr>
<th>Sphere</th>
<th>Cylinder</th>
</tr>
</thead>
<tbody>
<tr>
<td>radius : double;</td>
<td>radius : double;</td>
</tr>
<tr>
<td>volume : double;</td>
<td>height : double;</td>
</tr>
<tr>
<td>surface : double;</td>
<td>volume : double;</td>
</tr>
<tr>
<td></td>
<td>surface : double;</td>
</tr>
</tbody>
</table>

Etc.
public class Sphere {
    private double radius;
    private static final double PI = Math.PI;

    public Sphere () { radius = 1; }
    public Sphere (double r) { radius = r; }

    public void setRadius (double r) {  // Accessor
        radius = r;
    }
    public double getRadius () {
        return radius;
    }

    public double volume () {
        return 4 * PI * radius * radius * radius / 3;
    }
    public static double volume(double r) {  // Class method that can be used
        return 4 * PI * r * r *r / 3;  // directly without an instance
    }
}
Object-Oriented Design
Good programming practice

- Classes should be:
  - Correct – good overview
  - Efficient – small routines, little overhead
  - Reusable
  - Changeable