**LEBANESE AMERICAN UNIVERSITY**

**School of Arts and Science**

**Department of Computer Science and Mathematics**

**CSC 245: Objects and Data Abstraction**

# Lab I

05 Feb. 2015

**Problem 1**

There is a store owner that asked you to write a program for him to keep track of his items’ information. The items are separated into two categories which are taxable and non-taxable.

A non-taxable item has an ID, name, and price, while a taxable item has the same fields plus a tax rate. Since both entities share similar properties, it is logical to use inheritance. The classes’ design is as follows.

|  |
| --- |
| **Item** |
| - id : String  - name : String  - price : double |
| + Item()  + Item(id : String, name : String, price : double)  + setId(id : String) : void  + getId() : String  + setName(name : String) : void  + getName() : String  + setPrice(price : double) : void  + getPrice() : double |

|  |
| --- |
| **TaxableItem** |
| - taxRate : double |
| + TaxableItem()  + TaxableItem (id : String, name : String, price : double, taxRate : double)  + setTaxRate(taxRate : double)  + getTaxRate() : double  + getPrice() : double |

The Item class, which represents a non-taxable item, has three variables which are the ID, name, and price. It also has access modifier methods (setters and getters) for each of these variables, and two constructors.

The TaxableItem class will extend from the Item class and will have an extra double variable which is the tax rate, which should have access modifier methods. The class must also override the getPrice method which should return the price after applying the tax rate to it (price + price \* taxRate).

Implement the previous classes and a Main class for testing. The Main class should have a main method that will read input from the user about an item and prints its information.

The first line of input will be an integer representing the number of test cases. Each test case begins with a string having the value “taxable” or “non-taxable”. If “non-taxable”, the next three lines of input will be the id, name, and price of the item. If “taxable”, an extra line of input will be read which represents the tax rate (assume tax rate will always be between 0 and 1).

For each test case, print the item’s information using the following format:

**#id (name): $price**

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 2  non-taxable  12ABC  Green couch  100  taxable  AC767  Blue table  200  0.1 | #12ABC (Green couch): $100.0  #AC767 (Blue Table): $220.0 |

**Problem 2**

Your friend is majoring in Mathematics and is currently enrolled in a course about geometry. She is facing a problem with calculating information about different geometric shapes so she asked you to write her a program that will accomplish that.

You decided to start with circular shapes for now, which are a circle and a sphere. The circle will have one field which is the radius. Your friend would like to able to calculate the diameter, the circumference, and the area of the circle. As for the sphere, it will also have a radius and your friend would like to calculate its area and volume. Since radius is common for both shapes, you decided to use inheritance. The classes are designed as follows.

|  |
| --- |
| **Circle** |
| - radius : double |
| + Circle()  + Circle(radius : double)  + setRadius(radius: double) : void  + getRadius() : double  + calculateDiameter() : double  + calculateCircumference() : double  + calculateArea() : double |

|  |
| --- |
| **Sphere** |
|  |
| + Sphere()  + Sphere(radius : double)  + calculateArea() : double  + calculateVolume() : double |

The Circle class has one variable which is the radius. It also has access modifier methods (setters and getters) for the radius, and two constructors. The default constructor will initialize the radius to zero.

The calculateDiameter method returns a double value representing the diameter of the circle (2 \* radius).

The calculateCircumference method returns a double value representing the circumference of the circle (diameter \* pi).

The calculateArea method returns a double value representing the area of the circle (radius2 \* pi).

The Sphere class will extend the Circle class and override the calculateArea method. The area of a sphere is 4 \* radius2 \* pi. It will also have a method for calculating the volume of the sphere which is 4/3 \* pi \* radius3.

Implement the previous classes and a Main class for testing. The Main class should have a main method that will read input from the user about a shape and prints its information.

The first line of input will be an integer representing the number of test cases. Each test case begins with a string having the value “circle” or “sphere” and an integer representing the radius.

For each test case, print the diameter, circumference, and area if it is a circle. If the shape is a sphere, then print the area and the volume.

**Note:** Use Java’s Math.PI to get the value of pi.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 2  circle 5  sphere 4 | Diameter: 10  Circumference: 31.41  Area: 78.53  Area: 201.06  Volume: 268.08 |

**Problem 3**

A company has asked you to write them a program that calculates the bonus income of an employee based on which department he works in. There are three departments, the item sales department, the subscription sales department, and the assembly department.

An employee in the item sales department gets $100 per unit sold and an extra $20 per unit if they sell more than 30 items.

An employee in the subscription sales department gets 10% from the sales amount if his sales amount is more than $6000.

An employee in the assembly department will get $2 per item he assembles after assembling 500 items.

Implement three methods called getBonus each with their own parameters and implementation (method overloading) to handle the different types of employees.

The first line of input will be an integer representing the number of test cases. Each test case begins with a string determining which department the employee works in. If the department is item sales, the next input will be the number of units sold. If the department is subscription sales, the next input will be the sales amount. Finally, if the department is assembly, the next input will be the number of items assembled.

For each test case, print the bonus value that the employee will get.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 3  items 50  subscription 8000  assembly 510 | $2400  $800  $20 |