Intermediate Java II
MISM/MSIT 95-713
Homework 2

There are 4 exercises in this homework, each 25 points.

The objectives of this homework include:

- Defining classes and creating objects
  - Providing encapsulation
  - Writing constructors
  - Implementing methods specifically the
    - Setter/getter methods
    - toString() method
    - Static functions
- Implementing composition and inheritance relationships among classes
- Implementing class hierarchies
- Implementing UML models
- Practicing static variables and functions
- Creating and using packages
- Practicing access modifiers
- Practicing overriding
- Providing documentation comments and generating Java documentation files.

For this homework assignment and for all the upcoming homework assignments, follow the commenting and coding convention discussed in the class, unless otherwise specified explicitly. Make sure you follow the minimal class description guidelines for all the classes you introduce, except for the test driver classes that contain the main function. Name your files as specified in each of the exercises below. Put all your java files into a zip file named *yourandrewid*_homework2.zip and submit it to the blackboard before the due date.

1. Define a class named ComplexNumber in order to abstract the complex numbers. Complex numbers have a real and an imaginary part, both of which to be represented by double type in your class definition. Define add and subtract methods which both take an object of type ComplexNumber and return the *this* reference after performing the addition or subtraction. Also, define static add and subtract functions that take two ComplexNumber objects and return a new ComplexNumber object. Additionally define the toString() method. Define at least three constructors; a default one, one taking two double parameters for the real and the imaginary parts, another one taking an object of ComplexNumber class as parameter. Have a Homework2_1 class to test each of the methods of the ComplexNumber class and print out the results.

2.Implement the inheritance hierarchy shown Figure 1 as described below:

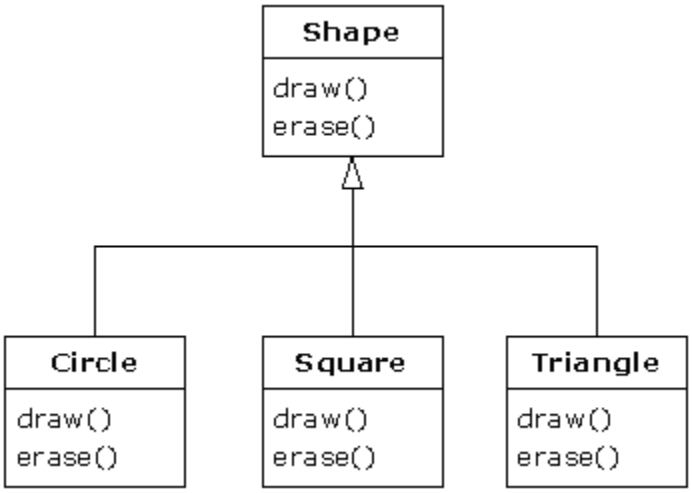

Figure 1 Inheritance Hierarchy for Shape Classes

Create a package called edu.heinz.cmu.oop95713.Shape and define the Shape, Circle, Square and Triangle classes in that package. Also define a Point class in this package as explained below. All the classes inside this package should be public.

Shape class is an abstract class with abstract draw and erase methods, both public and void. Circle, Square and Triangle classes are all final.

Introduce a Point class that has two integer instance members, x and y. You can use the Point class that you wrote for exercise 4, but make sure x and y are private. The toString() method of the Point class should return a String of the form "(x,y)" with the actual values of x and y.

The Circle class should have an instance of Point class representing its center and a member of double primitive data type representing its radius. The draw method of the Circle class simply should print out something like "Drawing circle at point (10,20) with radius 5.0". Its erase() method should print out "Erasing circle at point (10,20) with radius 5.0".

The Square class should have four instances of Point class. Its draw() method should print out something like "Drawing square at points (x1,y1), (x2,y1), (x1,y2), (x2,y2)" with actual values of x1, y1, x2, y2. Its erase method should simply print out a message indicating erasing at those points.

The Triangle class should have three instances of Point class. Its draw() and erase() methods should be similar to those of the Square class.

While printing out the messages in draw() and erase() methods, use the toString() method of the Point instances implicitly.

The constructors of Circle, Square and Triangle classes should be overloaded to take either x and y integer values to initialize the points, or, objects of Point class. For example, you should allow creation of Circle objects like:

```
Circle c = new Circle(10, 20, 5);
```

and also like:

```
Point p = new Point(10, 20);
Circle c = new Circle(p, 5);
```

Don't worry about verifying the inputs passed onto the constructors of Square and Triangle; just take them without error checking.

Define your main function in Homework2_2 class using the default package. In your main function, create objects of Circle, Square and Triangle classes, put them in an array of Shape references, invoke draw() on all of them and then invoke erase() on all of them.

3. Consider the UML conceptual model shown in Figure 2. You will be writing class definitions for each of the concepts and additionally for the test driver program following the instructions below.

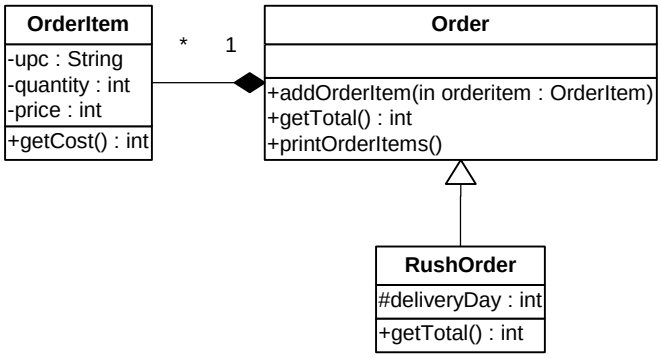

Figure 2 Conceptual Model for Order

Add the class definitions of OrderItem, Order and RushOrder classes in separate .java files and place them within the edu.heinz.cmu.java package. Note that all of these classes are to be public.
OrderItem class has a String upc, an integer quantity and an integer price. The getCost() method returns the multiplication of its quantity and price.
Order class has a list of OrderItem objects. The addOrderItem() method takes an object of OrderItem as the parameter and stores it in the list. The getTotal() method returns the total cost of all order items in the order. The printOrderItems() method prints information about each order item. While printing information about each order item, the toString() method of the OrderItem instances is invoked.
RushOrder class extends the Order class. It has an integer instance variable deliveryDay, a protected member, which represents in how many days the order should be delivered. The getTotal() method overrides the definition of the super class in the following way: It first invokes the getTotal() of the super class to find the total for all items in the order. It then adds the delivery charge. The delivery charge for one day delivery is $25, for two day delivery is $15, for three day delivery is $10. It is free for four or more days. Note that, the delivery charge should be added only if there are items in the order.
Introduce a Homework2_3 class as a test driver in the default package. It should execute as follows:

i. Create an array of four Order objects, named “orders”. Create an instance of Order class and assign it to the first element of the array. Create an instance of RushOrder class with one day delivery and assign it to the second element of the array. Create an instance of RushOrder class with

two day delivery and assign it to the third element of the array. Create an instance of RushOrder class with three day delivery and assign it to the forth element of the array.

ii. Prompt the user to enter the UPC code for an item or enter "done" to quit. Read the user input.
   - While the user enters a UPC (anything other than "done"), prompt for the quantity and read it. You may assume the user enters a valid input, that is, an integer.
   - Generate a random number from 50 to 100 for the price.
   - Create an OrderItem object using the values of UPC, quantity and price.
   - Generate a random number from 1 to 7 for the delivery day.
   - Add the order item to the element of the orders array corresponding to the delivery day via the addOrderItem(). If delivery day is four or bigger, add the order item into the first element, which is the instance of Order class. If the delivery day is one, add it to the second element of the orders array, which is the instance of the RushOrder class with one day delivery, and so on.
   - Continue the loop until the user enters "done".

iii. For each order in the orders array:
   - Print information about each order including its type (make sure the information is printed by the toString() method).
   - Print out information about all the order items in the order.
   - Print out the subtotal for this order.

iv. Print out the total cost in all of the orders.

4.

a. Implement the inheritance hierarchy shown in Figure 3. Have Point class, Circle class and Cylinder class in a single file named Homework2_4a.java. Have a main function to create point, circle and cylinder objects, print information about them via their toString() methods. Invoke the area() method on circle object, invoke area() and volume() methods on cylinder object, and print all the results.

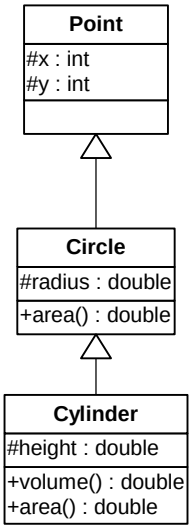

Figure 3 "BAD" Inheritance Hierarchy

b. Inheritance hierarchy in Figure 3 doesn't represent the is-a relationship. Rather than achieving code reuse by inheritance, implement the has-a relationship among the point, circle and cylinder classes where circle has a point and cylinder has a circle. Implement them following the composition relationship in a single file named Homework2_4b.java. Name the classes as CPoint, CCircle and CCylinder to differentiate them from the ones in the previous exercise. Have a main function to create objects of these new classes and call their methods just like you did in the previous exercise.