**COEN 275 OO Analysis, Design and Programming Fall 2016**

**Assignment 1 (100 pts) Due: 11th Oct(10 pm)**

In this first assignment, you will learn to use a **Java IDE** (**Eclipse, NetBeans** etc)

* To create a Java program, compile and run the program.
* To define classes in packages.
* Use composition and inheritance.
* To test your classes.

**Question 1 (30 pts)**

**Define the class required in this question, in a package with name, your *lastname*.assign1.money**

Create a class called **USMoney** with two instance data members (integers), **dollars** and **cents**. Define the following methods:

* A **constructor** that takes two parameters (*dollars* and *cents*) and initializes the corresponding data members. The constructor should check that the cents value is between 0 and 99 and if not, transfer some of the cents to the dollars to make it between and 99.
* Define a default constructor.
* Define two **setter** methods, one to set dollars (**setDollars())** and one to set cents (**setCents**()).
* Define two **getter** methods, one to return the dollars and one to return the cents.
* Define a method called **addTo**(int *dollars*, int *cents*) that add the parameter values to the data members.
* Define a method called **add** (USMoney *money*) that creates and returns a new **USMoney** object representing the sum of the object whose add() is invoked and the object passed as parameter. Make sure it does not modify the value of the two existing objects.
* Define a method called **toString()** that returns a string representation of the object.

Given below is an example of using some of the methods of **USMoney** object.

Usage:

USMoney m1 = new USMoney (15,80);

System.out.println (m1); // Should print $15.80

m1.addTo(25,100);

System.out.println (m1); // Should print $41.80

USMoney m2 = m1.add( new USMoney (2.90));

System.out.println (m2); // Should print $44.70

System.out.println (m1); // Should print $41.80

Create a class called **USMoneyTester with the main, in a package with name, your *lastname*.assign1.q1**

Inside the main, include statements to create instances of USMoney (you must create instances using each of the constructors) and call the methods on each of the instances.

**Include the following test case. Make sure that you are calling the methods with values that test the constraints and rules specified for the class.**

**// Creating instances**

**USMoney amt1 = new USMoney ();**

System.out.println (amt1);

**amt1.setCents (250);**

System.out.println (amt1);

**amt1.setDollars (10);**

System.out.println (amt1);

System.out.println (amt1.getCents());

**USMoney amt2 = amt1.add( new USMoney (2,90));**

System.out.println (amt1);

System.out.println (amt2);

**amt2.addTo(amt1.getDollars(), amt1.getCents());**

System.out.println (amt2);

**USMoney amt3 = new USMoney (99,120);**

**amt3.addTo(99,120);**

System.out.println (amt3);

**Question 2 (70 pts)**

In this question, you will use the class **USMoney** from exercise 1, as the type for a data member in another class.

**Define the classes and interfaces required in this question, in a package with name, your *lastname*.assign1.salebin**

Define the classes and interfaces as described below:

**interface ItemType**

isFragile(): Boolean

getPrice(): USMoney

getWeight(): double

getDetails() : String

**class SaleItem implements ItemType**

itemName: String

price: USMoney

weight: double

fragile: boolean

getDetails(): String

This method should return a concatenated string of itemName, price

**SaleItem** (itemName:String, price: USMoney, weight:double,fragile:boolean)

**interface BinType**

addItem (ItemType item)

calculatePrice(): USMoney

getWeight(): double

getNoOfItems() : integer

showDetails() : String

**class Bin implments BinType**

* binNumber: String
* ItemType [] items;
* double maxWeight (a constant and is the same for all Bins): initialized to a value of your choice
* **Bin()-** The numbers for bins should start with at least one letter and followed by a unique number. Use the generateBinNumber() method to generate a unique binNumber.

**generateBinNumber():** integer

A unique number (among all the bins created) should be generated and returned. This can be done using a class-level (static) ***counter*** that can be incremented after using the counter for the bin number. Therefore this method should be a class-level method.

* **addItem (ItemType item):**checks the weight of the item and if adding it to the bin does not exceed Bin’s ***maxWeight***, and if the item is **not fragile**, adds it to the array, ***items***.
* **calculatePrice():** USMoney

calculates the total price of all items in items array and adds $100.00 as the bin cost and returns the total**.**

* **showDetails()**: String

returns a string by concatenating the binNumber+currentWeight+totalCost of Items.

**class SmartBin extends Bin**

* **label**: String
* **SmartBin()-** Bin numbers for Smartbins should start with a “SM” and followed by an integer. Use the generateBinNumber() method to generate a unique binNumber.
* **setLabel(String label)**

sets the label with the value in the parameter.

* **addItem (ItemType item):**checks the weight of the item and if adding it to the bin does not exceed Bin’s maxWeight, adds it to the array, items. If the added item is fragile, sets the bin’s label to “*Fragile - Handle with Care*”.
* **calculatePrice():** USMoney

calculates the total price of all items in items array and adds $100.00 as the bin cost. If there are fragile items in the bin, adds $10 extra for each of the fragile items. Returns the total price.

* **showDetails()**: String

returns a string by concatenating the binNumber+label+currentWeight+totalCost of Items.

**class Q2TestCase: This class will have the main().**

**Define the class required in this question, in a package with name, your *lastname*.assign1.q2**

**Include the following test cases and capture the output.**

1. Create at least 5 instances of SaleItems with data of your choice. Make some of the items “fragile”.
2. Create an instance of a Bin.
3. Add the instances of SaleItems from a) to the Bin instance in b). Make sure that you exceed the maximum weight of the bin.
4. Show details on the Bin instance in b)
5. Create at least 5 more instances of SaleItems with data of your choice. Make some of the items “fragile”.
6. Create an instance of a SmartBin.
7. Add the instances of SaleItems from d) to the Bin instance in b). Make sure that you exceed the maximum weight of the bin.
8. Show details on the Bin instance in f)
9. Create an instance of a Bin as follows:

Bin bin3 = new SmartBin();

1. Can you add fragile SaleItems to bin3?

**Note:**

1. Make the data members private or protected (not public)
2. Write all the necessary constructors. Make sure that you have the required constructors to initialize the superclass members, in case of inheritance.
3. You are free to add any extra data members or methods as you need.
4. Remember the principle, “**Program to the interface and not to the implementation**”, wherever it is possible. For example, declare the most generic type (an interface?) for a parameter , so that you can actually pass an instance of a class that implements that interface.

**General Requirements:**

1. Include comments in the program (where applicable) to make your logic clear.
2. Use Java conventions for naming classes, methods and variables.

**Point distribution:**

Correctness: 80%

Program comments: 5%

Naming Conventions: 5%

Source code + executable jar file: 10%

What to submit:

1. **A zipped file (assign1\_*yourfirstInitialLastName*) with the following:**

* A readme document (include your name, course name, assignment number) identifying the IDE and Java version used.
* Source code
* An executable jar file. (Please check the documentation of your IDE, Eclipse for example, to create a jar file in that environment)

References:

[Java packages](https://docs.oracle.com/javase/tutorial/java/package/packages.html)

[Java Naming Conventions](http://java.about.com/od/javasyntax/a/nameconventions.htm)

Creating [Executable](https://docs.oracle.com/javase/tutorial/deployment/jar/build.html) jarfiles