CSCI 1101 – Winter 2017
Laboratory No. 4

This lab is a continuation of the concepts of object-oriented programming, and focuses on object aggregation and inheritance.

Note:

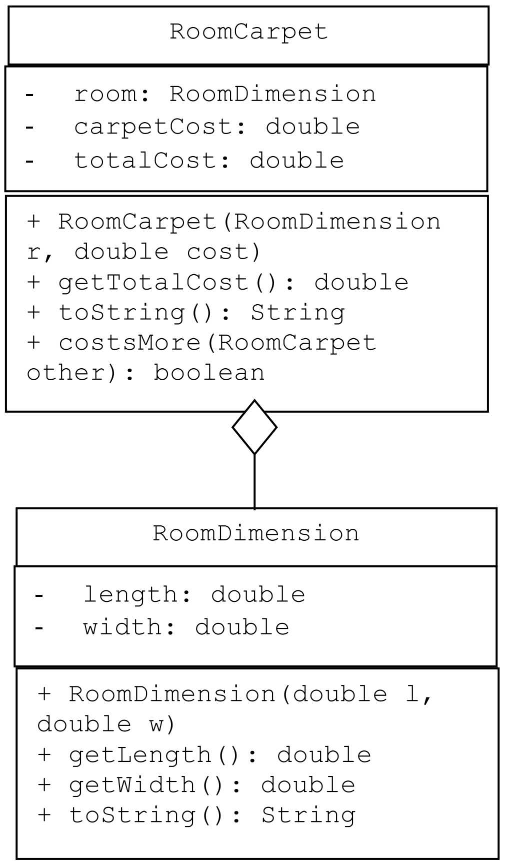
1. *All submissions must be made on Brightspace (dal.ca/brightspace).*
2. ***Submission deadline is 11.55 p.m. (5 minutes to midnight) on Saturday, February 11th, 2017.***
3. *Put the java source code files and the text outputs for each exercise in a folder. Zip the folder into one file and submit the zip file.*

4. Marking Scheme:
Each exercise carries 10 points. Your final score will be scaled down to a value out of 10.
Working code, Outputs included, Efficient, Comments included: 10/10
No comments: subtract one point
Unnecessarily long code and inefficient program, improper use of variables: subtract one point
No outputs: subtract two points
Code not working: subtract up to six points depending upon the extent to which the program is incorrect.

***5. Error checking**: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.*

***6. Testing your code and generating the outputs:** If the test data is included in the question, use that to test your classes. In addition, test it with two more input sets. Otherwise, create your own test data and run your program for at least 3 input sets such that they cover the range of results expected.*

Exercise 1: The following figure shows a UML diagram that **aggregates** the RoomCarpet class from the RoomDimension class.



length and width are in feet.
carpetCost is the cost of the carpet per square foot.
totalCost is the total cost to carpet the room.
Constructor RoomCarpet(RoomDimension r, double cost) sets the attribute room to r, carpetCost to cost, and finds the total cost to carpet the room (that is, length*width*carpetCost) and sets totalCost to this value.
getTotalCost(); returns the value of the totalCost attribute.
toString() method in RoomDimension returns the length and width
toString() method in RoomCarpet returns the size (length and width), cost per square foot and total cost. (Note: You must use the toString() method of the RoomDimension class here).
costsMore(RoomCarpet other): returns true if the total cost of carpeting "this" room is more than the total cost of carpeting another room (other), and false otherwise.

Implement the two classes. Test this in a demo program RoomCarpetDemo.java that prompts a user to enter values of two rooms of different sizes and carpet costs, and displays if the first room costs more to carpet than the second room using the costsMore method. Test it for at least three different input sets. Here's a sample screen dialog:

```
Enter the dimensions of the first room: 10.5 12.5
Enter the cost per square foot of carpet: 10.00
Enter the dimensions of the second room: 20.0 15.0
Enter the cost per square foot of carpet: 7.00
Room with Length: 20.0 and Width: 15.0  Carpet Cost ($per sq.ft): 7.0   Total cost ($): 
2100.0
costs more than
Room with Length: 10.5 and Width: 12.5  Carpet Cost ($per sq.ft): 10.0  Total cost ($): 
1312.5
```

Here's another one:

```
Enter the dimensions of the first room: 10.0 10.0
Enter the cost per square foot of carpet: 5.00
Enter the dimensions of the second room: 10.0 5.0
Enter the cost per square foot of carpet: 10.00
Room with Length: 10.0 and Width: 10.0  Carpet Cost ($per sq.ft): 5.0   Total cost ($): 
500.0
and
Room with Length: 10.0 and Width: 5.0   Carpet Cost ($per sq.ft): 10.0  Total cost ($): 
500.0
cost the same
```

Files for submission: RoomDimension.java, RoomCarpet.java and RoomCarpetDemo.java.

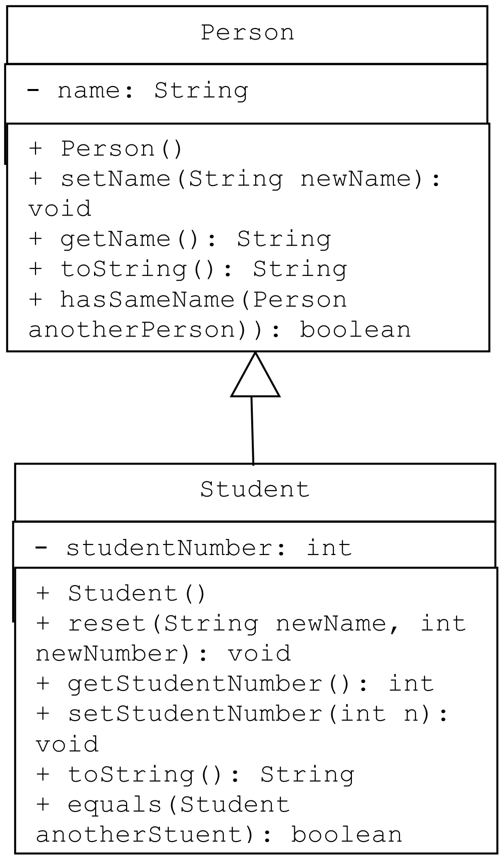
Exercise 2: Create a class called Coin as follows:

Attributes: name and value of a coin (valid names and values are Toonie(200), Loonie(100), Quarter(25), Dime(10) and Nickel(5)).
Methods : Constructor that sets the name and value;
Get methods
toString method

Create another class called Wallet that is aggregated from the Coin class as follows:
Attributes: An array of Coin objects
capacity (how many coin objects it can hold – an integer value)
num (current number of coin objects in the wallet – an integer value)
total value of all Coin objects (in cents) stored in the wallet (an integer value)
Methods: Constructor (that sets the total value to zero, num to zero, the capacity to the given value, and creates an array of Coin objects of size equal to the capacity)
addCoin (that adds a given Coin object to the Coin object array and updates the num and total value)
removeCoin (that removes the last Coin object from the array and returns it; it should also update the total value and num)
toString method
Note: addCoin and removeCoin methods should have the appropriate error checks

Test the classes by creating a Wallet, and adding and removing coins from it.

Exercise 3: The following figure shows a UML diagram in which the class `Student` is inherited from the class `Person`


Person
- name: String
+ Person()
+ setName(String newName): void
+ getName(): String
+ toString(): String
+ hasSameName(Person anotherPerson)): boolean
Student
- studentNumber: int
+ Student()
+ reset(String newName, int newNumber): void
+ getStudentNumber(): int
+ setStudentNumber(int n): void
+ toString(): String
+ equals(Student anotherStuent): boolean

Implement the two classes. Write a demo program that tests all the methods in the `Student` class as well as the inherited methods. Note: In the `toString` method in the `Student` class, use the method `getName()` to get the name.