CSCI 1101 – Winter 2013
Laboratory No. 3

This lab is a continuation of the concepts of object-oriented programming, static variables and static methods, and aggregation, in addition to the previous concepts that we learned.
Note: Submission by Moodle only.
Deadline: Saturday, February 2, NOON

Exercise 1 (a) : This is an example on static variables that we discussed in class. Implement and test the following program. Trace the output so that you understand the operation of the program. No submission required for this part.
Warning: Cutting and pasting code may cause errors!

```
public class TurnTaker
{
	private static int turn = 0;
	private int myTurn;
	private String name;
	public TurnTaker(String n, int t)
	{
		name = n;
		myTurn = t;
	}
	public String getName()
	{
		return name;
	}
	public static int getTurn()
	{
		turn++;
		return turn;
	}
	public boolean isMyTurn()
	{
		if (turn==myTurn)
			return true;
		else
			return false;
	}
}


public class TurnTakerDemo
{
	public static void main(String[] args)
	{
		TurnTaker person1 = new TurnTaker("Romeo", 1);
		TurnTaker person2 = new TurnTaker("Juliet", 3);

		for(int i = 1; i<= 5; i++)
		{
			System.out.println("Turn = " + TurnTaker.getTurn());
			if (person1.isMyTurn())
				System.out.println("Love from "+ person1.getName());
			if (person2.isMyTurn())
				System.out.println("Love from " + person2.getName());
		}
	}
}
```

Exercise 1(b): Modify the program so that you get the following output:

```
Turn = 1
Love from Romeo
Love from Juliet
Turn = 2
Love from Romeo
Love from Juliet
Turn = 3
Love from Romeo
Love from Juliet
Turn = 4
Love from Romeo
Love from Juliet
Turn = 5
Love from Romeo
Love from Juliet
```

Exercise 2: Chuck owns several pizza stands distributed throughout town. You are the java expert who will help Chuck keep track of the number of pizzas sold by the stand as well as the total number of pizzas sold in all the stands. For this, define a class named PizzaStand that has an instance variable for the Pizza Stand's ID number and an instance variable for how many pizzas sold in that stand that day. Add a static variable that tracks the total number of pizzas sold by all the stands. Add another static variable that specifies the cost per pizza.

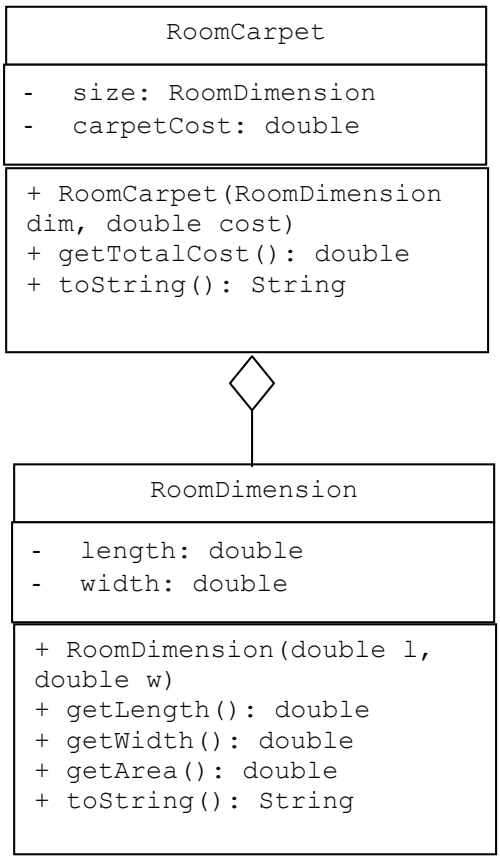
Add the following methods:
A constructor that sets the ID number to some value and the number of pizzas sold by that stand to 0.
A method named justSold that increments the number of pizzas sold by that stand by 1.
Another method to return the number of pizzas sold by that stand.
A static method to set the cost per pizza.
A static method that returns the total number of pizzas sold by all stands.
A static method to return the total sales.
Test the class with at least five pizza stands that each sell a number of pizzas during the day.

A sample screen dialog is given below. The cost of the pizza is set to $5.00.

```
Pizza Sales:
1    2      (means that Pizza Stand 1 sold 2 pizzas)

2    1

3    1

4    1

5    1

Total pizzas sold: 6
Total sales: 30.0

Process completed.
```

…over/

Exercise 3: The following figure shows a UML diagram that **aggregates** the RoomCarpet class from the RoomDimension class. Create the two classes. Note: carpetCost is the cost of the carpet per square foot.



Exercise 4: Create a class called Coin as follows:

Attributes: name and value of a coin (for e.g., quarter has a value of 25, dime has a value 10 and so on);
Methods : Constructor that sets the name and value;
Get methods
toString method

Create another class called Wallet as follows:
Attributes: total value of all coins (in cents) stored in the wallet
Methods: Constructor (that sets the total value to zero)
addCoin (that adds a given Coin object's value to the total value)
removeCoin (that removes a given Coin object's value from the total value)
toString method

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