**CSCI 1101 – 2017**

**Laboratory Report 3**

Your task is to complete the assigned work using JGrasp or an IDE of your choosing. You may use your own computer, or one of the lab computers provided.

Your submission should be a **ZIP** file containing your source code files. You should submit your **ZIP file** on Brightspace:

[http://dal.brightspace.com](http://dal.brightspace.com/)

|  |
| --- |
| **Submission Deadlines (firm):**  Monday Labs: due Wednesday by 12:00pm (noon)  Friday Labs: due Sunday by 12:00pm (noon) |

NB:

* Try to submit this report *during* the lab so that your TA can check it for you before you submit!
* Attendance is mandatory in all labs, and will form part of your overall lab grade
* Acknowledge any help that you obtained from friends, TAs, the Learning Centre, etc., by completing the Declaration on the first page of this document. Obtaining help is fine, *so long as you acknowledge it!*
* Any students who cannot log on to the lab computers should speak to the Help Desk to set up their account.
* Textbooks, class handouts, and any other materials are welcomed and encouraged in all labs!
* Food and drink are not permitted in the computer labs
* Late labs are ***not*** accepted! It is known that computer errors, power outages, and network lag are 105% more likely to occur between 11:55-11:59am, in the moment they can do the most damage. Account for this, and give yourself the chance to make a timely submission!

**Header Comments**

Your code should now include header comments for **all** of your class (.java) files. The comment should include the lab/assignment number, the course (CSCI 1101), the name of your program and a short description of the entire class, the date, your name and Banner ID, and a declaration that matches the first page of this document (e.g., whether you received help). See the example below for what a header comment should include:  
  
**/\*Lab1, Question 1 CSCI 1101**

**Student.java holds information about a student at Dalhousie in CSCI1101 and**

**their grades**

**June 29, 2015**

**John Smith B00112345**

**This is entirely my own work. \*/**

**public class Student {**

**//rest of Code**

If applicable, your demo class should then also have a similar header:

**/\*Lab1, Question 1 - demo class CSCI 1101**

**StudentDemo.java is a demo program for the Student class. It creates student**

**objects, and compares different students.**

**June 29, 2015**

**John Smith B00112345**

**I received help with creating Student objects from my TA but the rest is my**

**own work. \*/**

**public class StudentDemo {**

**//rest of Code**

**Exercise 1**

Sarah owns several ice-cream trucks in Toronto. You will help Sarah track the number of ice-creams sold by each of the trucks and the total number of ice-creams sold by all the trucks in the city. For this, define a class named IceCreamTruck. Each IceCreamTruck has a truckID (an instance variable) and an instance variable for ice-creams sold that day by that truck. IceCreamTruck also has two static variables. One static variable tracks the total number of ice-creams sold by all the trucks. The other static variable specifies the price of ice-cream, which Sarah will only update once per year.

Add the following methods:

* A constructor that sets the truckID number to some value and the number of ice-creams sold by that cart to 0.
* An instance method named sale that increments the number of ice-creams sold by that truck by 1.
* A toString method that returns the number of ice-creams sold by the truck, that truck's total sales, and labels this using the truck’s ID number.
* A static method to set the cost per ice-cream.
* A static method that returns the total number of ice-creams sold by all the trucks.
* A static method that returns the average number of ice-creams sold by all the trucks.
* A static method to return the revenue (total value of all sales) from all the trucks.

Test the class with at least five ice-cream trucks that each sells a different number of ice-creams during the day.

A sample screen dialog output is given below - your output does not have to match exactly. The cost of one ice-cream is set to $2.00 in this example.

Ice-cream Sales by Truck:

Ice-creams sold by truck1: 2 Total Sales for truck1: $4.00

Ice-creams sold by truck2: 1 Total Sales for truck2: $2.00

Ice-creams sold by truck3: 1 Total Sales for truck3: $2.00

Ice-creams sold by truck4: 1 Total Sales for truck4: $2.00

Ice-creams sold by truck5: 1 Total Sales for truck5: $2.00

Total Ice-cream sold by all trucks: 6

Total sales: $12.00

Average sales per truck: $2.40

**Exercise 2**

Define the class Rectangle that contains double instance variables named width and height, and a static variable named printChar. Include the following:

* A no-arg constructor that creates a default rectangle with 1.0 for both width and height.
* A constructor that creates a rectangle with the specified width, and height.
* Get and set methods for all the instance variables.
* A method getArea() that returns the area of the rectangle.
* A static method setPrintChar() that modifies the static variable
* A toString() method that returns a String that represents an outline drawing of the rectangle using the character given by printChar. For example, if printChar were ‘c’ and one Rectangle had a width of 5 and a height of 4, then printing the toString() method to output would give  
    
  ccccc  
  c c  
  ccccc

Write a test program that first sets the printChar for the Rectangle class as ‘\*’, then creates two Rectangle Objects of reasonable sizes. Print out each rectangle separately to output using the toString() method, followed by the value of that rectangle’s area.

Change the printChar to a reasonable (non-white-space) character of your choosing, modify both instance variables for both Rectangle Objects, and then repeat the output process: print out each rectangle separately to output followed by the value of that rectangle’s area.