COE618 Lab 1: An Introduction to Netbeans

General Lab Rules
1. Each lab must have its own directory (folder). For the first lab, the directory is lab1, for the second lab2, etc.
2. To submit a lab, create a zip file of the entire directory and sent it as an attachment to your TA.
3. The deadline for submission is 24 hours before the beginning of the next lab. For example, if you start lab 1 (a 2-week lab) at 10 AM Tuesday, January 17, you have to submit lab1 by 10 AM Monday, January 30.

Objectives
- Review Java, jUnit and Netbeans.
- Write subclasses.
- Use composition as well.
- Use ordinary and static instance variables.
- Duration: 2 weeks.

REQUIRED EXERCISE
To get started you must perform the following operations.

1. Create your course directory with the command:
   mkdir coe618

2. Change to your course directory with the command:
   cd coe618

3. Create your lab1 sub-directory with the command:
   mkdir lab1

4. Change to your lab1 directory with the command:
   cd lab1

Exercise 1. (70 marks)

In the Netbeans program, click on Project > New Project and save it as "Exercise1" on your lab1 directory.

Create a new class called InvoiceItem.
Assume that this InvoiceItem class is one that a store might use to represent an item sold at the store. An InvoiceItem should include four pieces of information as instance variables:
- a part number (name: partNum; type :String),
- a part description (name: desc; type: String),
- a quantity of the item being purchased (name: quantity; type int) and
- a price per item (name: price; type double).

The class should have a constructor that takes four arguments (part number, part description, quantity of the item being purchased and price per item) and initializes the four instance variables. Hint: Instead of typing in the constructor yourself, make netbeans do it for you! Select Insert Code in the Source menu. Then choose Generate constructor and click on all the instance variables to include.

● To complete the constructor, the values of quantity and price have to be verified. If either is negative, an IllegalArgumentException should be thrown to abort the construction.

● Provide getter methods for each instance variable and a setter method for the quantity instance variable.

● The setQuantity method (like the constructor) should throw an IllegalArgumentException if the quantity is negative.

● Provide a method named getTotal that calculates the total cost (i.e. multiplies the quantity by the price per item), then returns the amount as a double value.

● Implement the toString method for this class that returns a string containing the values of the four instance variables separated by commas. For example, an invoice item created with new InvoiceItem("1234a", "a widget", 5.12, 4) (i.e. 4 items described as "a widget" with part number "1234a" and a cost of 5.12 each) should produce the String "1234a,a widget,5.12,4".

Test the InvoiceItem class

Once the InvoiceItem class has no compilation errors, test the methods toString(), setQuantity and getTotal using junit.

Netbeans can create the boilerplate code for unit tests.

- From the Tools menu, select "Create Junit tests".
- Select Junit 4 (which is superior to version 3 that you used last year.)
- The generated code will include test methods for each public method. These methods are annotated with @Test

Note: you do not have to test Exceptions at this point.

- The generated tests are guaranteed to fail (which reminds you to fix them!) An example of a test method will look something like:

```
@Test
public void testGetQuantity() {
System.out.println("getQuantity");
InvoiceItem instance = new InvoiceItem("x", "wid", 1.2, 3);
int expResult = 3;
int result = instance.getQuantity();
assertEquals(expResult, result);
}
```

Design, Implement and test the Invoice class

An Invoice class groups InvoiceItems into a single object. Each Invoice has a unique serial number:

the first Invoice has serial number 1, the second has serial number 2, etc. (Hint: implementing this will require a static variable.) A newly created invoice will have no InvoiceItems. A public method
public void add(InvoiceItem item) will add that item to the Invoice. Other public method public double getTotal() will return the total of all the InvoiceItems.
Other methods to design and implement are:
- getNumItems(): returns the total number of items.
- InvoiceItem getItem(int i): return the i'th Invoice or throw an exception if i is illegal.

Proceed as follows:
1. Create the class Invoice.
2. Write the skeleton code for the public methods.
3. Implement the methods adding whatever private instance variables you need.
4. Test the public methods with junit.
5. Do at least one test to verify that Exceptions are thrown when they should be.

Design, Implement and test the RefundItem class
If a customer returns merchandise, a refund is added to the Invoice. A RefundItem is very similar to an InvoiceItem and should be a subclass of it. The only difference is that the calculation of getTotal() should return a negative number. For example, if someone returns 2 items that each cose 10.50, the getTotal() method should return -21.00. (Note that the constructor for the RefundItem still gives the price and quantity as positive numbers just as in InvoiceItem. Only the getTotal method should differ.

Exercise 2. (30 marks)
In the Netbeans program, click on Project > New Project and save it as "Exercise2" on your lab1 directory.

TwoDShape class is an abstract class that has two instance variables: x and y. The variables x and y represents the x-coordinate and y-coordinate of the top left corner of a two dimensional shape object.

```
public abstract class TwoDShape {
private int x;
private int y;
/**
* Constructor
* @param x the x-coordinate of the top left corner
* @param y the y-coordinate of the top left corner
*/
public TwoDShape( int x, int y ){
   this.x = x;
   this.y = y; }
```

```
public int getX() {
   return x;}

public int getY(){
   return y;}

public void setX(int x){
   this.x = x;}

public void setY(int y){
   this.y = y;}

public String toString (){
   return (x+ "" + y);}

public abstract void scale( double factor );
}
```

a) Create a new class called "Circle" as a subclass of TwoDShape. It has one instance variable: radius.

Provide a constructor that initializes the instance variables. It takes three parameters: x, y, radius. If the specified radius is negative, then it should throw an IllegalArgumentException.

Implement the scale(double factor) method. If the specified factor is positive, then this method should scale its radius by a factor of factor. Otherwise, it should throw an IllegalArgumentException.

Provide getter method for radius.

b) Write a unit test method called scaleTest that test method scale of Circle class. Please test the exceptional cases, for e.g. invoke the scale method with a negative argument and check if it really throws an IllegalArgumentException. (Use JUnit framework for implementing the test class).

Submitting your lab

(1) Zip your lab1 assignment within your coe618 directory

```
zip –r lab1 lab1
```

(2) Attach your lab1.zip file and send e-mail to your lab instructor, subject of your e-mail must be lab1 followed by your name, student id and section #.

Mr. Naimul Mefraz Khan: n77khan@ryerson.ca
Mr. Kevin Tang: ktang@ryerson.ca