COMP 1011: Advance Java Programming

Assignment #2

Due Date: (Mid-night – 11.59 pm) 18th Feb. **Marks/Weightage: 30/7.5%**
End Date: (Mid-night – 11.59 pm) 21st Feb. with 20% deduction/penalty. No exceptions.

Purpose: The purpose of this assignment is to:

- Practice the use of Multi-threading, Concurrency, Lambdas and Streams

References: Learning materials for chapters 17 and 23 of textbook, and other internet references (if any).

IDE: eclipse Java Enterprise Developer edition – 2022-12/2023-03 and Windows 10/11 OS.

Step1: At the start of eclipse, you must name your Eclipse workspace according to the following rule:

YourFullName_COMP1011SectionNumber_A02
For Example: **JohnSmith_COMP1011Sec002_A02 (if your section is Sec002)**

Step 2: And after that, you must name your Eclipse project according to the following rule:

YourFullName_COMP1011SectionNumber
For Example: **JohnSmith_COMP1011Sec002**

Step 3: And your package should be named as follows:

YourFullName_SectionNumber_Ex01
For Example: **johnsmith_sec002_ex01**

Submission/Upload Instructions:

After you complete, run and test your code, you need to do the following:

a) Close the eclipse, go to your **workspace** folder which you created in Step 1.
b) Zip it up. You should get a zip file like this– **JohnSmith_COMP1011Sec002_A02.zip. You should only be submitting it in the .zip file format (and not .rar or any other format)**
c) Upload this zip file using the **Assignment-02** link in blackboard.

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character for the first word and uppercase for every other word
- *classes* start with an *uppercase* character of every word
- **packages** use only *lowercase* characters
- *methods* start with a *lowercase* character for the first word and uppercase for every other word

Note: Late submissions are accepted until up to 3 days past due date with 20% deductions. After that no submission will be considered.

Exercise 01: This is based on the multi-threading. *[10 marks]*

Write a Java application that handles multiple ATM transactions (withdraw, deposit) at the same time.

a) Create an **Account** class and implement both **deposit** and **withdraw** operations.
b) Synchronize the operations to allow thread synchronization.
c) Use Java **Runnable** interface to implement a **Transaction** class.
d) Perform withdraw and deposit operations in **run** method.
e) Create an **AccountTest** class to test multiple transactions (threads). Use an ArrayList to create a list of three or more Transaction objects. Use method execute of **ExecutorService** to execute the threads. Display the results.

Exercise 02: This is based on manipulating a Stream<Invoice>. *[10 marks]*

For this exercise, use the following sample data. You need an invoice class (invoice.java), which is provided to you along with this.

Part Number	Part Description	Quantity	Price
87	Electric Sander	7	57.98
24	Power Saw	18	99.99
7	Sledge Hammer	11	21.50
77	Hammer	76	11.99
39	Lawn Mower	3	79.50
68	Screw Driver	106	6.99
56	Jig saw	21	11.00

Class Invoice includes four instance variables – partNumber (type String), part description (type String), a quantity of the item being purchased (type int) and pricePerItem (type double) and corresponding get methods.

Perform the following queries on the array of Invoice objects and display the results:

a) Use streams to sort the Invoice objects by partDescsription, then display the results. *[1.5 marks]*
b) Use streams to sort the Invoice objects by pricePerItem, then display the results. *[1.5 marks]*
c) Use streams to map each Invoice to its partDescription and quantity, sort the results by quantity, then display the results. *[2 marks]*
d) Use streams to map each Invoice to its partDescription and value of the Invoice (i.e. quantity * pricePerItem). Order the results by Invoice Value. *[2 marks]*
e) Modify the part(d) to select the invoice values in the range $200 to $500. *[2 marks]*
f) Find any one invoice in which the partDescription contains the word “saw”. *[1 mark]*

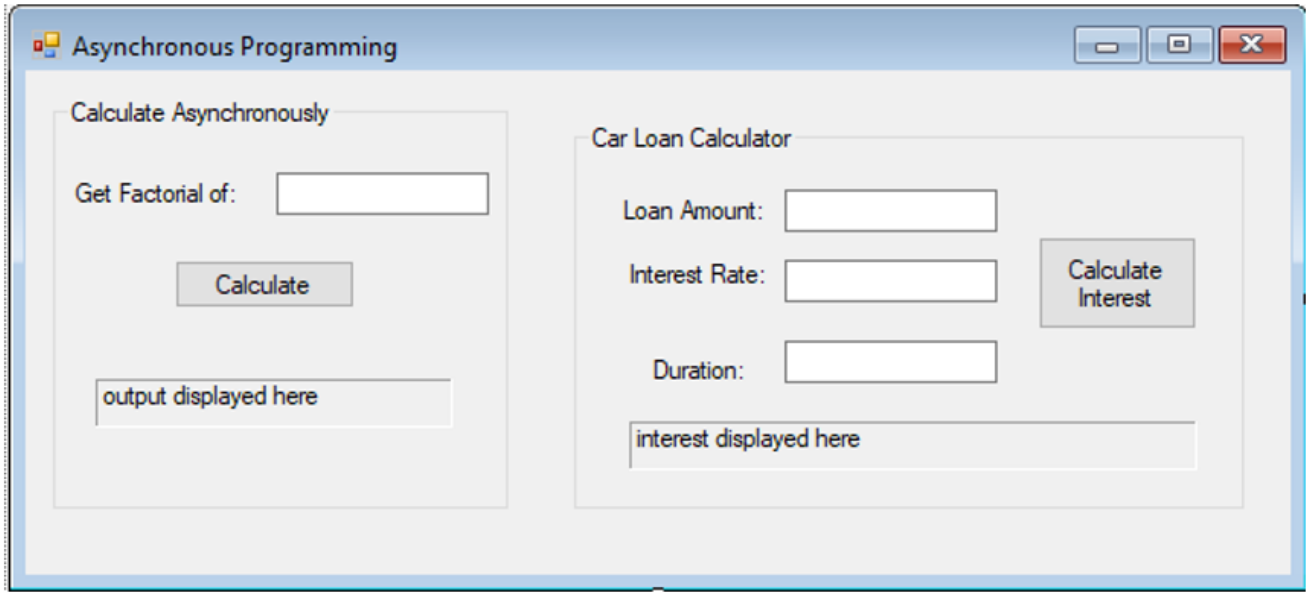
Exercise 03: *[10 marks]*

Build the GUI, using JavaFx/Scenebuilder as shown below.

Here it shows two functionalities, one for calculating the factorial of a large integer value asynchronously or in parallel to other functionality – Car Loan Calculator.

Since calculating a factorial value for large int value may take some time. This task you need to start on a separate thread, and while it calculates factorial, user should be able to calculate the interest on a car loan.

Note: Do some research, how to handle large numeric values, you may find some pre-defined some structs or types to handle large values in Java.



Evaluation/Rubric:

Functionality	
(a) Correct implementation of business requirements, GUI, code logic, functionality etc.	90%
(b) Correct implementation of exception handling and intended results	5%
(c) Comments, correct naming of variables, methods, classes, etc.	2.5%
(d) Friendly input/output	2.5%
Total	100%