ENSF 409 – Principles of Software Development
Summer 2021

Lab Assignment #7: Java Generics, and Multithreading

Due Dates
Submit electronically on D2L **before 1:00 PM on Friday July 30th**

The objectives of this lab are to understand the concepts of:

1. Java generic types and generic methods
2. Multithreading and Java Threads


The following rules apply to this lab and all other lab assignments in future:

1. B Before submitting your lab reports, take a moment to make sure that you are handing in all the material that is required. If you forget to hand something in, that is your fault; you can't use `I forgot' as an excuse to hand in parts of the assignment late.
2. **20% marks** will be deducted from the assignments handed in up to **24 hours** after each due date. It means if your mark is X out of Y, you will only gain 0.8 times X. There will be no credit for assignments turned in later than 24 hours after the due dates; they will be returned unmarked. Exceptions can be made but the students must inform that Instructor beforehand to accommodate if acceptable.


Exercise - 1: Communicating between threads (10 Marks)

Communication between threads can be accomplished by passing values in a shared variable or a shared data area. In C or C++, several threads could share a global variable. However, Java does not have global variables. In Java, each thread's local variables and methods are private to it because each thread has its own private activation stack. However, threads are not given private copies of static class variables or object instance variables. Therefore, you might ask what's a good way to share a variable in Java?

Task 1 – Download the `SimpleThread.java` and the `Resource.java` classes from D2L. This program has two threads that are supposed to increment the counter of the resource and print the new value. Carefully read the code, and Run the `SimpleThread` program, and consider the output. why doesn't the value of the `counter` increase in some steps?

Whenever, two or more threads (or processes) share a variable (or data area), any computation on the shared variable by one thread must be protected from interference by the other threads. Only by careful analysis can we identify these critical regions and then use a synchronization primitive to protect them. In Java, this synchronization primitive is a Hoare-style monitor associated with the object. The monitor guarantees that at most one thread can execute within a critical region for that object at any time.

Task 2 – Fix this program to avoid the above issue. If you run the program after fixing it, you should see each number only once in the console.

Task 3 – Change the program to use the runnable interface in the implementation of your multi-thread program

What to Hand in: Please submit **all the java files** including the files you have modified in a zip folder. For this exercise, you don't need to add any inline comment or method-interface-documentation.


Exercise - 2: Practice with Threads (20 Marks)

Task 1 - Randomized Number (10 Marks)

What to Do: Write a program that runs 5 threads. Each thread is responsible to generate a random number in the range of (1, 100). The program waits for all the threads to finish, and then calculates the sum of the numbers (which were randomized), and prints their sum. Implement a Runnable class that randomizes a number and store it in a member field. Your main program can go over all the objects and check the stored values in each object.

Task 2 - Modified Randomized Number I (5 Marks)

What to Do: Modify the program from Task1 of this exercise, so that instead of each object keeping its own score, you will use one collection to store all the results in.

Task 3 - Modified Randomized Number II (5 Marks)

What to Do: Modify the program from Task 2 of this exercise, so that the program uses a fixed thread pool instead of individual threads. The thread pool should have 5 threads and be instantiated using an `ExecutorService` object.

What to Hand in: Submit your programs for Tasks 1, 2 and 3, and a sample output of your programs. For this exercise, you don't need to add any inline comment or method-interface-documentation.


Exercise - 3: Generic Linked List (15 Marks)

First, download the following files from D2L: `Demo5.java`, `LinkedList5.java`, `Node5.java`, `Date5.java`, `Point5.java`, and `Product5.java`. Then, create a project and import the given files into your project. If you read the given files carefully, you will see the definition of a linked list class and its node that can only create linked lists of `Double` data type objects. One of the features of this `linkedlist` is that in addition to `itemM` data field (type `Double`), there is also another data field called `keyM` (type `Integer`). Now, compile and run the program. Everything should work and program should produce an output.

What to do: Your job in this exercise is to modify the `linkedlist` and covert it to a `generic linkedlist`. In other words, you are supposed to change the program to be able to create `linkedlists` of virtually any types. For example, you should be able to declare `linkedlist` of objects such as `Point, Date, Integer, Double, String`, etc.

```
LinkedList <Date> w;
LinkedList <Point> x;
LinkedList <Product> y;
LinkedList <Integer> z;
LinkedList <String> s;
```

The following paragraphs explains further steps to get this job done:

1. Modify class Node to a generic type that can hold an item of any data type.
2. Modify the code to convert the `LinkedList` class to a generic class.
3. Modify the functions `try_to_find()` to a generic function. The class `Demo` should remain as a regular class (non-generic).
4. Modify the method print to use Wild Card to print the data in a generic linked list **with one type** parameter. Note: don't make this method a generic-method.
5. Compile and test your program for creating objects of parameterized `LinkedList`. Uncomment the test method in `class Demo` to test your program.

What to hand in: Please submit **all the java files** including the files you have modified in a zip folder and a PDF file containing the output of your modified program.. You don't need to submit javadoc HTML files.