Langara
THE COLLEGE OF HIGHER LEARNING.

Department of Computing Science & Information Systems
CPSC 1181
Lab#10
Nov 17, 2022

Objectives:

Practice thread programming and race condition.

Preparation:

Study thread section from your text book and class notes.

Due date:

Due Date: 11:00 PM on Wednesday Nov 23, 2022

Where to upload:

zip folder to yourStudentID.zip and upload it to Lab10 in D2L.

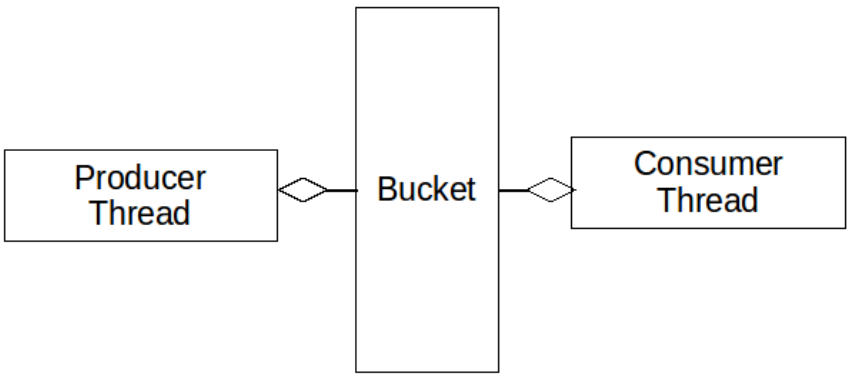
No tutorial this week.

Multi-thread programming

The goal of this lab assignment is to create two threads, producer and consumer. The producer thread generates random integer numbers and consumer thread consumes the numbers generated by the producer.

The Bucket class is already developed and you can download it from here. The Bucket class internally uses an ArrayList of integers to add integer numbers to the bucket, finds their median value, and reset the bucket. An instance of this class is shared between both producer and consumer classes.

The rough UML diagram of the the system is shown below.



Phase 1: [20 marks]

Develop class Producer that implements the runnable interface. The class takes an instance of the Bucket class as an argument at construction time. The class performs following tasks *while the Thread is not interrupted*:

- The run() method of the class generates random integer numbers in range from 0 to 255 and

adds them to the bucket by invoking add(...) method of the Bucket class.
- Puts the thread to sleep for 1ms after adding each element to the ArrayList.
- Repeats the process while the thread is not interrupted.

Develop class Consumer that implements the runnable interface. The class takes an instance of the Bucket class as an argument at construction time. Note that one single instance of the Bucket class is shared among Producer and Consumer classes. The class performs following tasks *while the Thread is not interrupted*:

- The run() method of the class invokes the median() method of the bucket and prints the returned value on the screen. The sample run of the program:

```
median : 140, size : 5
median : 99, size : 4
median : 192, size : 4
median : 133, size : 5
```

- puts the thread to sleep for 5ms
- Repeat the process while the thread is not interrupted.

Develop class BucketThreadTester that contains a main method. The class performs following tasks:

- Creates an instance of the Bucket class.
- Creates an instance of the Producer and Consumer classes.
- Starts both of the threads.

Run the BucketThreadTester program. After a few iterations the program will crash with a message like following message:

```
Exception in thread "Thread-1" java.util.ConcurrentModificationException
     at java.base/java.util.ArrayList.sort(ArrayList.java:1751)
     at java.base/java.util.Collections.sort(Collections.java:145)
     at Bucket.median(Bucket.java:40)
     at Consumer.run(Consumer.java:12)
```

The program crashes due to the problem known as Race-condition.

Pahse2:[10 marks]

Modify the Bucket class by adding ReentrantLock to the class. Implement the locking mechanism to prevent the program from being crashed.

Run the program for at least 15 seconds to be sure that your program is not crashing anymore.

Pahse3:[10 marks]

The program runs forever. Develop a timer in class BucketThreadTester that interrupts both producer and consumer after 10 seconds to stop the program.

Pahse4:[10 marks]

Add new conditional locking mechanism to acheive the following

1. the producer will not add more that 50 elements to the bucket.

2. The consumer will return the median of the bucket when the size of the bucket is exactly equal to 50. Nor more, no less.

3. Stop the both producer and consumer thread exactly after 30 seconds. Note you use use interrupt to stop them in the main method;

Sample run of the program in this case:

```
median : 128, size : 50
median : 140, size : 50
median : 114, size : 50
median : 100, size : 50
median : 127, size : 50
median : 100, size : 50
median : 119, size : 50
median : 143, size : 50
```

Notes:
1. Rrefer to example 5 in your notes.
2. Check TimerDemo.java program to setup and use Timer class.

What to upload

Create one page UML diagram of your design, and save it either as PDF or as word 2003 (JPEG Inserted into word document).

Zip your source files and UML diagram into *your StudentID.zip* file and then upload it to lab10 in D2L Dropbox.

TOTAL MARK: 50