Instructions

Please read the instructions carefully and follow the naming conventions specified for each question. Solutions must be submitted in the Blackboard Dropbox created for Lab 3.

Note that the deadline is strictly enforced. The system tracks the exact time that submissions are uploaded. **There is a 10% per day penalty for late submissions.**

Additional Notes

- No JUnit testing is required for this lab
- You will be required to present and explain your solution to the professor during the lab period.
- You may use any IDE for development but note that professor testing will be done exclusively on the command line.

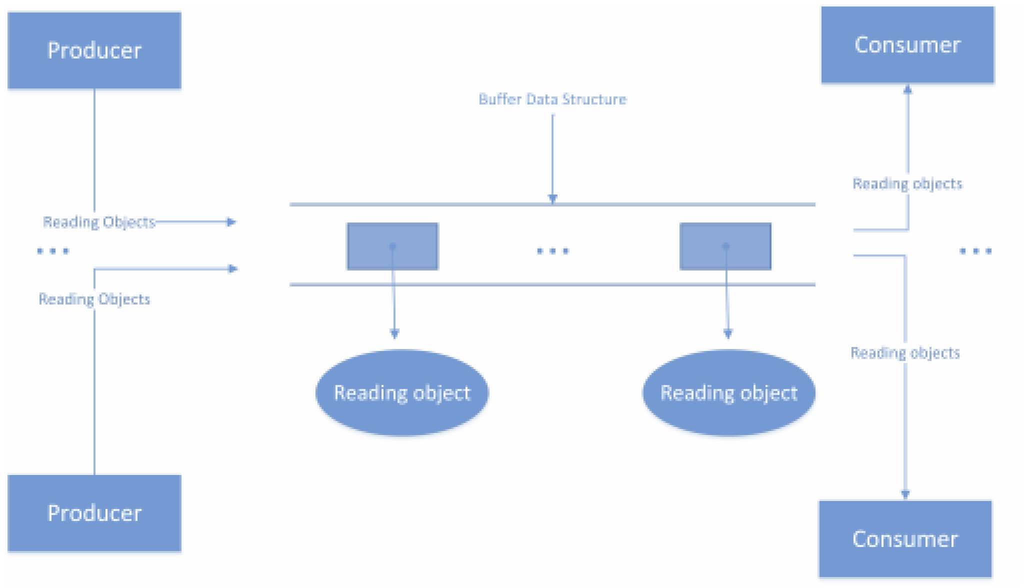

Figure 1: High Level View of Data Flow for the Producer/Consumer Application

Program Requirements

Question 1) In our lectures, there was discussion about the theoretical benefits of different collection container implementatons. In practice, these ideas may or may not hold depending on various optomizations added by the library designer. In this lab, you *compare the performance of various Java List implementations.*

Write a program in a class named ListTest which loads, sorts and accesses values in a List collection. You will be comparing the ArrayList, LinkedList and Vector implementations

- the program must contain the following functionality
  - load a given number of randomly selected long values into the List
  - sort a List of given size
  - walk through and access a List of a given size in sequential order
  - access random elements in List of a given size x, "y" times
- each operation above shall be timed when executed. Each should be triggered and timed independently. Ensure that if steps are needed before the operation can occur (like pre-population), that they are not included in the timing results.
- the program's JavaFX graphical user interface should allow a user to select various options in order to configure a test run. In particular, the user must be able to select an List implementation, container size, and other variables (depending on the operation) for a run. Timing results should be presented to the user. The interface shall automatically update the screen as tests are complete.
- Do multiple runs with different parameters and critically analyze your results. *How did you configure your runs? Why did you choose them? What did you expect to observe?* Plotting your results in a table and/or graphs will assist you (and the reader) with indentifying definite patterns that can be attributed to implementation differences. You should review documentation of the class and related methods you have employed. Explain your findings in the pdf document to be submitted.
- The code should all be placed in a package called btp400.lab3.q1

Question 2) In this question, you will create a multithreaded application which has multiple threads accessing a shared resource.

Review a desciription of the Producer/Consumer pattern found here (`https://en.wikipedia.org/wiki/Producer-consumer_problem`). You will implement an instance of this pattern where a Producer generates Reading objects (provided) subject to the following conditions.

(a) The Buffer should store 100 Reader objects. Two implementations shall be included.
   i. uses the ArrayBlockingQueue object for storage. (`https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ArrayBlockingQueue.html`).
   ii. uses a non-thread safe collection (synchronization issue must be handled by this implementation)

(b) Each Producer class object should generate 2000 Reading objects, insert them into the Buffer object and then terminate

(c) Consumer class objects remove Reading objects from the Buffer, and write their contents (obtained using the overloaded toString() method) to a file. Each Consumer object has its own file to write to. This process should terminate when it must wait for longer than 1 minute to remove an object.

(d) The ProducerTester class should contain a program that executes two instances of the problem (one using each type of Buffer implementation). The instances create 5 Producers threads and 3 Consumer threads.

(e) All code artifacts should be placed in a package called btp400.lab3.q2