CS 251: [Basic Java:] Numbers, and Threads (Outlab)

- Handed out: 9/26 Due: 9/29 11pm.
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on the web page.
- For all the questions given below, make different Java files for every different object.

Overview

Java is the world's #1 programming language (so says Oracle :) Java is characterized by the buzzwords: Simple, Object oriented, Distributed, Multithreaded, Dynamic, Architecture neutral, Portable, High performance Robust, and Secure. The official language for Android development is Java. For more information on learning Java, visit `http://docs.oracle.com/javase/tutorial/tutorialLearningPaths.html` The goal of this lab is to learn a tiny aspect of `Java`. We will do Java IDE in a subsequent lab.

Tasks:

1. **[Big Integer:]** In this task, you will learn use of `BigInteger` class of `Java`.

   In C and C++, you have worked on various data types: int, float, long, double etc. But all of these can store integer values lies only in a fixed range. `long` will be enough as long as (no pun intended) you don't go over $2^{63} - 1$. But what if I want to find the value of 100!? Luckily `Java` has the `BigInteger` class which can contain an integer, where the digits of precision is limited only by your computer memory.

   - `BigInteger` or `BigDecimal` class does not have the function for calculating the square root or the n^{th} root of number. Make your own function `NthRoot.java` which should return the n^{th} root rounded to 6 decimal places:

     ```
     BigDecimal NthRoot(BigInteger num,int n)
     ```

     Call this function from `main` with input taken from the standard input, and print the output to stdout. (Hint: recall Lab01 for the Newton's method.)
   - Write a Java program `SolveBigEquation.java` to determine A and B such that

     $$|A - B| = 50!$$
     $$A * B = 100!$$

     Print all pairs (A, B). Suppose (A1,B1), (A2,B2),.... (An,Bn) are the solutions of above set of equations, then your output should look like (sorted in increasing lexicographic order):
     A1,B1
     A2,B2
     .
     .
     .
     An,Bn

We think one of your solutions should match the following pair:
9660549437994914524283407846946037882198865631919262771306851407525729156558173.945,
9660549437994944938376609560324081494807031696688107148948420368037729156558173.945

(BTW 100! has 158 digits. Also BigInteger calculations are slower than normal calculations.)

2. **[Threads:]** A `Book` is an object with two methods: `read` and `write`. For our purposes, the content of the book is a single integer stored as a private variable. The constructor initializes the content to zero.

   Your task is to create `Book` and two other entities: `Reader` and `Writer`. There are multiple instances of `Reader` and exactly one `Writer`. Every `Reader` reads a `Book` and that takes 5 seconds. The reader takes rest for some random time between 0 and 3 seconds before he starts reading. The `Writer` object modifies `Book`; the content is increased by 1 and that takes 10 seconds. After writing, the `Writer` thinks for 2 seconds. The `Writer` writes 5 times.

   There is exactly one (instance of) `Book` in our system shared among the readers and the writer. When the program exits, the Book has 5 as its content.

   (a) Create the `ReaderWriter` class with the `main` function. In the main function, initialize a `Book` object. Initialize 20 Threads for Reader and 1 for Writer, and start them. Assign a number to every Reader. Your program should print which Reader is entering or leaving at the time the Reader starts (ends) reading. Your program should also print "Writer is entering" (respectively, "Writer is leaving") when the writer starts (ends) writing the Book. Refer to `ReaderWriteSampleOutput1.txt` for sample output.Readers and the writer interleave in this part of the task (chaos prevails).

   (b) In the above step, it is possible that different readers are reading different versions of the book. Change the code as follows:

   - Ensure if readers are reading a book then the writer can not perform the write operation. Multiple readers should be able to read the book simultaneously.
   - Ensure if the writer is writing, then no reader can read book.
   - Ensure each reader reads exactly two different versions of the book. For example, when the second time Reader starts reading, he checks if the version has changed. If version is unchanged then he waits for some random time between 0 and 5 seconds. If the version is changed he starts reading. In addition to the print statements you are printing in part(a), your output should also print which version Reader is reading. Refer to `ReaderWriteSampleOutput2.txt` for sample output.

   Hint: Use `wait`, `notify`, `notifyAll` and `synchronized` block.

3. **[Extra credit. 25% Bonus:]** Will be graded only if previous tasks are completed. In this task, we will create a multi-threaded `BankApplication`. You are required to create 2 classes: `Bank` and `Account`. Each `Account` must have 2 attributes: Balance, AccountNumber. `Bank` must maintain a list of Accounts. `Bank` must be able to do the following operations: Add a new Account, deposit money into specific account, withdraw money from specific account.

   Constraints:

- Your program should be able to handle simultaneous requests for adding Account, deposit, and withdrawl from multiple users. For this, you use three different types of Thread: `AddAccountThread`, `DepositThread`, and `WithdrawlThread`.
- If there are simultaneous multiple requests pertaining to a unique account number, process the requests correctly.

Your program should print:

- Which account number is created at the time when new Account is created. e.g. for Account number 3, it should print: Account with Account Number 3 has been created
- On Deposit (Withdrawl), It should print Account Number, Amount added (deducted) and New Balance. Some sample output is given below:
  836 Amount has been added in Account: 1, New Balance is:836
  235 Amount has been deducted from Account: 2 New Balance is:460
- It should print the error messages in the following situations:
  (a) When user is requesting to deposit(withdrawl) into(from) wrong Account Number.
  (b) When user is requesting to withdrawl amount more than balance.

  Test your code by starting some threads in the main function of `BankApplication` class.

Submission Guidelines:

Submit the following documents: Submit

1. `NthRoot.java`, `SolveBigEquation.java` written in Question 1
2. `ReaderWriter.java`, `Reader.java`, `Writer.java` and `Book.java` written in Question 2.
3. `BankApplication.java`, `Bank.java`, `Account.java`, `AddAccountThread.java`, `DepositThread.java`, and `WithdrawlThread.java` written in Question 3.
4. Do not forget to put readme.txt file in a folder. The folder and its compressed version should both be named `lab08_groupXY_final`. Hence, you submit a tar.gz named `lab08_group07_final.tar.gz` if your group number is 7.