Read Me:-

i. Before going through below exercises please visit the link given below, where you can experience the coding standard that each and every developer should follow.

ii. This Code Conventions for the Java Programming Language document contains the standard conventions that Sun follow and recommend that we should follow. It covers filenames, file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices and includes a code example.

iii. LINK - http://www.oracle.com/technetwork/java/codeconv-138413.html

Multithreading Lab Exercise Day-3
Duration: 2 Hours

Program:1

A typical example of simultaneous threads is that of a bank account where two people could be operating at the same time. In this problem, we are writing a Account class.

* A Account class is given to you. This class is used to deposit and withdraw from the account.
* Declare an integer variable called balance. This variable holds the current balance of the account.
* Declare a method deposit(int amount):
* This method should add the given amount to balance. There is no need to validate amount. This method returns void.
* Declare a method withdraw(int amount):
* This method should subtract the given amount from balance. There is no need to validate amount. This method returns void.
* If Multiple threads are trying to use this class for operating the account, this class should be able to handle the multiple threads such that there is no thread interference or error in the balance.
* Use the class AccountTester.java to test the Account class and its methods. Make sure you test the class for single and multi threaded operations.

Program:2

A typical example of simultaneous threads is that of an online reservation system. Multiple users are trying to book seats/products at the same time. In this problem, we are writing a Ticker Dispenser in a railway reservation system.

* A TicketDispenser class is given to you. This class can give out a maximum of 100 tickets.
* In this class the maximum seats is already declared and assigned to the value of 100. Do not allot seats greater than this.
* Declare an integer variable called allotedSeats. This variable holds the current seat that is alloted.
* Declare a method allotSeatNumber():
* This method should return the alloted seat number to the caller.
* This method should start allotting seat numbers from 1 and allot seats in the increasing order.
* If all 100 seats have been allotted, return -1.
* If Multiple threads are trying to use this class for seat allotment, this class should be able to handle the multiple threads such that there is no thread interference or error.

* Use the class Tester.java to test the TicketDispenser class and its methods. Make sure you test the class for single and multi threaded operations.

All the Best