CNT 4714 – Project 2 – Spring 2016

Title: "Project 2: An Application Using Cooperating and Synchronized Multiple Threads In Java Using Locks"
Points: 100 points
Due Date: Sunday February 14, 2016 by 11:59 pm (WebCourses time)

Objectives: To practice programming cooperating, synchronized multiple threads of execution.

Description: In this programming assignment you will simulate the deposits and withdrawals made to a fictitious bank account (I'll let you use my real bank account if you promise to make only deposits! ☺). In this case the deposits and withdrawals will be made by synchronized threads. Synchronization is required for two reasons – (1) mutual exclusion (updates cannot be lost) and (2) because a withdrawal cannot occur if the amount of the withdrawal request is greater than the current balance in the account. This means that access to the account (the shared object) must be synchronized. This application requires cooperation and communication amongst the various threads (cooperating synchronized threads). (In other words, this problem is similar to the producer/consumer problem where there is more than one producer and more than one consumer process active simultaneously.) If a withdrawal thread attempts to withdraw an amount greater than the current balance in the account – then it must block itself and wait until a deposit has occurred before it can try again. As we covered in the lecture notes, this will require that the deposit threads signal all waiting withdrawal threads whenever a deposit is completed.

1. To keep things relatively simple as well as to see immediate results from a series of transactions (deposits and withdrawals) assume that deposits are made in amounts ranging from $1 to $200 (even dollars only) and withdrawals are made in amounts ranging from $1 to $50 (again, even dollars only).

2. You should have three deposit threads and six withdrawal threads executing simultaneously.

3. Once a deposit thread has executed, put it to sleep for few milliseconds or so (depends a little bit on the speed of your system as to how long you will want to sleep the depositor threads - basically we want to ensure a lot more withdrawals than deposits) to allow other threads to execute. This is the only situation in which a deposit thread will block.

4. For withdrawal threads, things will be a bit different depending on whether you are working on a single or multi-core processor.

    a. For single core processors, once a withdrawal thread has executed, have it yield to another thread. Since the thread is giving up the processor voluntarily, it will be unlikely to run again (attempt a second withdrawal in a row), before another thread runs. Note however, that it does not prevent it from running again, if all other withdrawal threads are blocked and all depositors are sleeping, it will run again.
    b. For multi-core processors, once a withdrawal thread has executed, have it sleep for some random period of time (again, a few milliseconds should be fine). Depending on which core a thread is executing, yielding the CPU won't ensure that the same thread will not run again immediately. While, sleeping the thread will also not ensure that it will not run two or more times in succession, it is less likely to do so in the multi-core environment.
    c. What we don't want to happen is either a single withdrawal thread or a group of withdrawal threads gaining the CPU and then executing a long sequence of withdrawal operations. Recall though that withdrawal threads block if they attempt to withdraw more than the current balance in the account.
    d. Similarly, we don't want depositor threads monopolizing the CPU either and causing the balance in the account to grow continuously. See page 7 for an illustration of this.

5. Assume all threads have the same priority.

6. The output from your program must look reasonably similar to the sample output shown below.

7. **Do not put the threads into a counted loop for your simulation.** In other words, the `run()` method should be an infinite loop.

8. **Do not use the Java synchronized statement.** I want you to handle the locking and signaling yourself. No monitors!

References:

Notes: Lecture Notes for Multithreaded Applications.

Restrictions:
Your source files shall begin with comments containing the following information:

```
/*  Name:
    Course: CNT 4714 Spring 2016
    Assignment title: Project 2 – Synchronized, Cooperating Threads Under Locking
    Due Date: February 14, 2016
*/
```

Input Specification: Internal to the program.

Output Specification: Console based. Your output should appear reasonably similar to the output shown below.

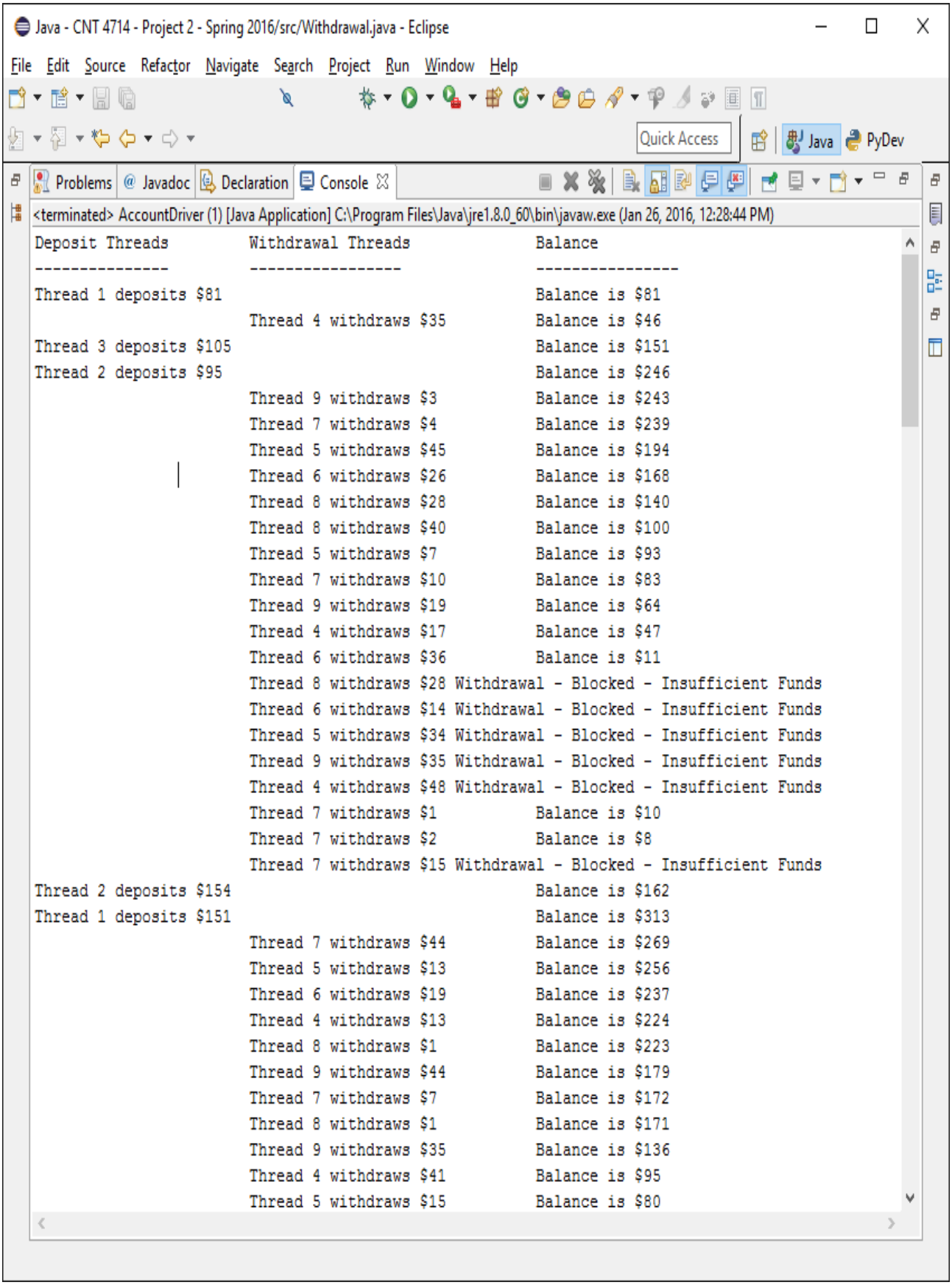
Deliverables:

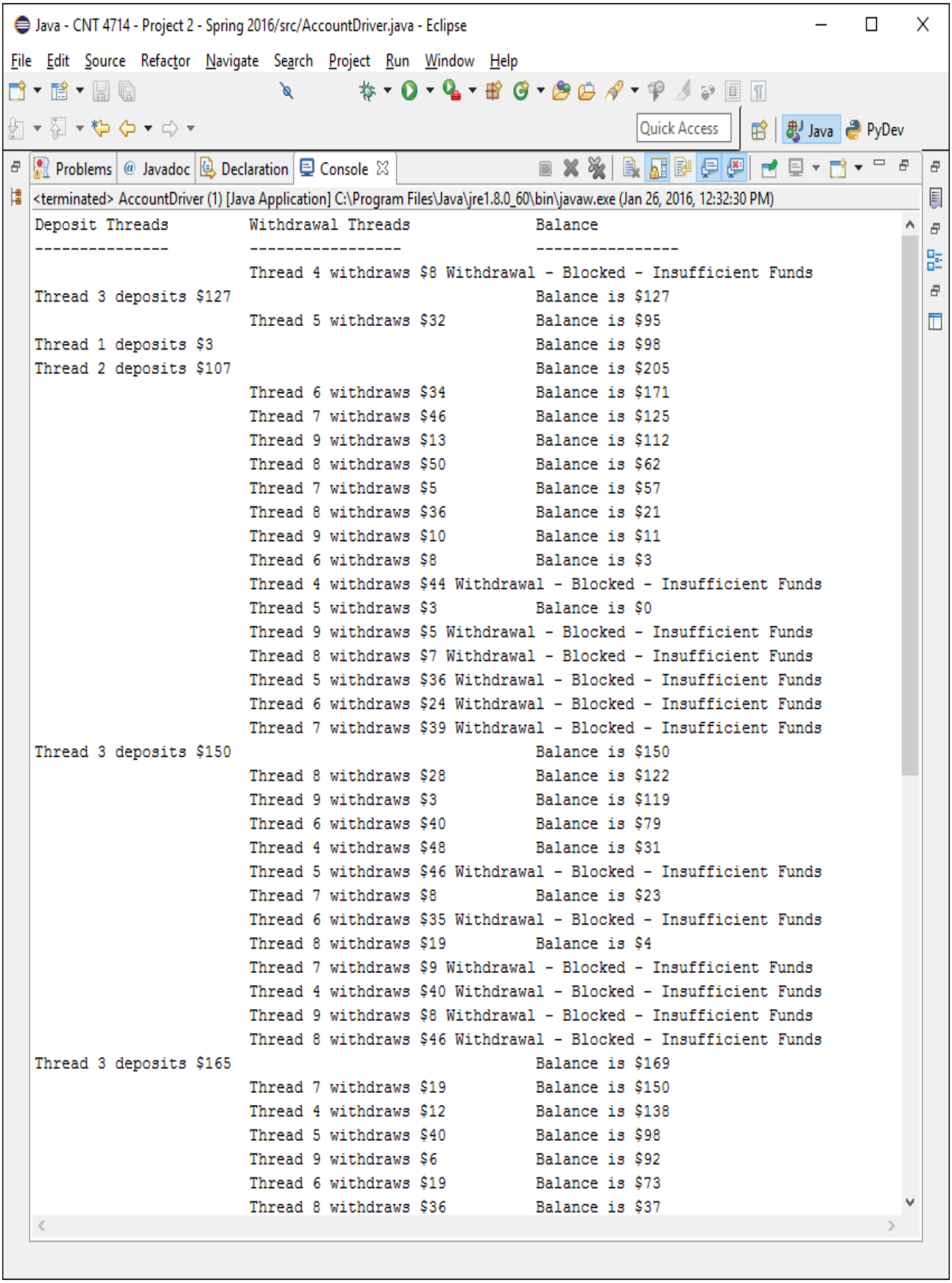
(1) Zip up all of your .java files and submit them via WebCourses no later than 11:59pm Sunday February 14, 2016.

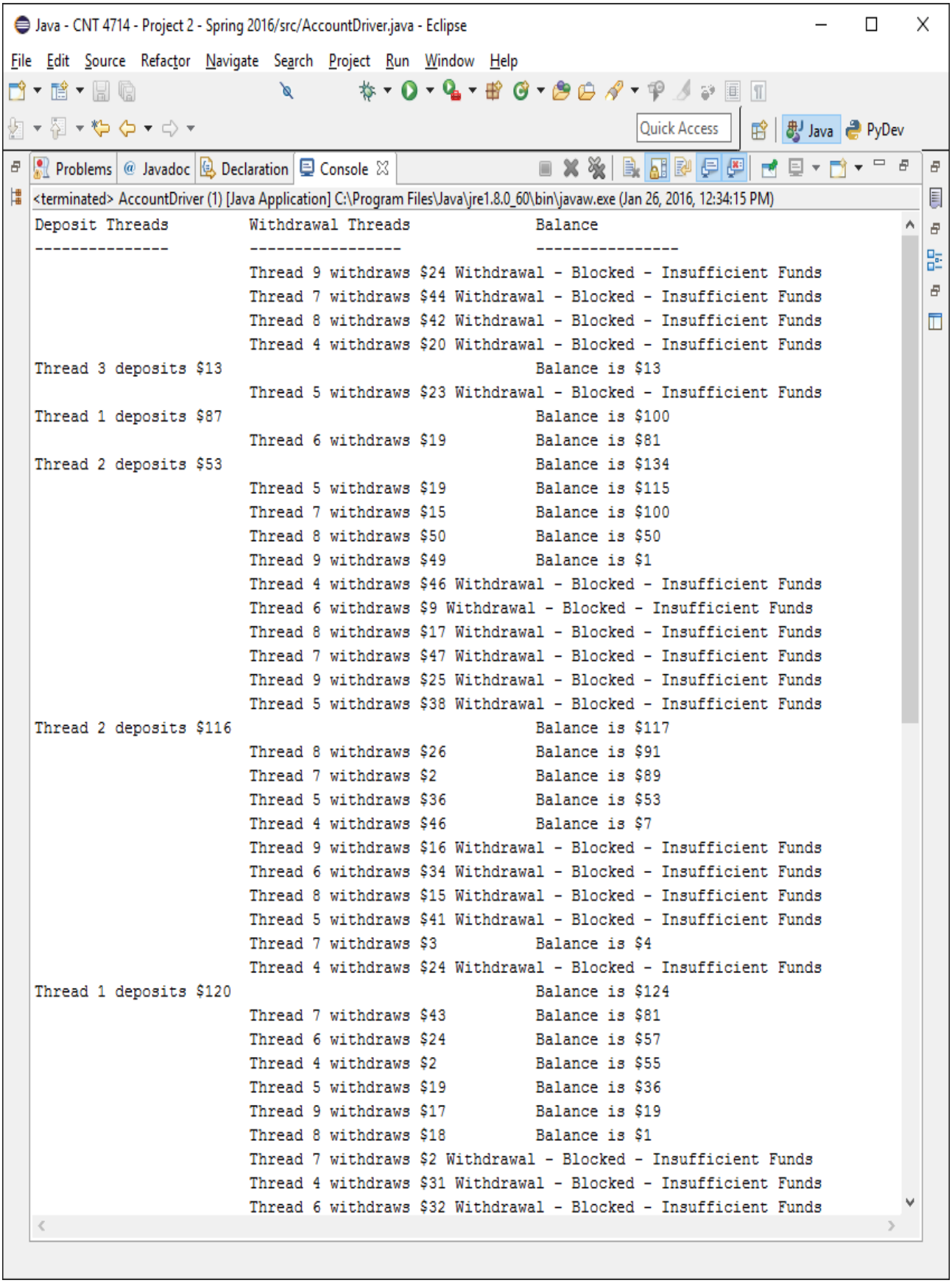
(2) Include at least one screen shot which illustrates the execution of your synchronized threaded application. See below for some representative examples. You can either do a screen shot of the console window like I did below or redirect your output to a file and take a screen shot from an editor.

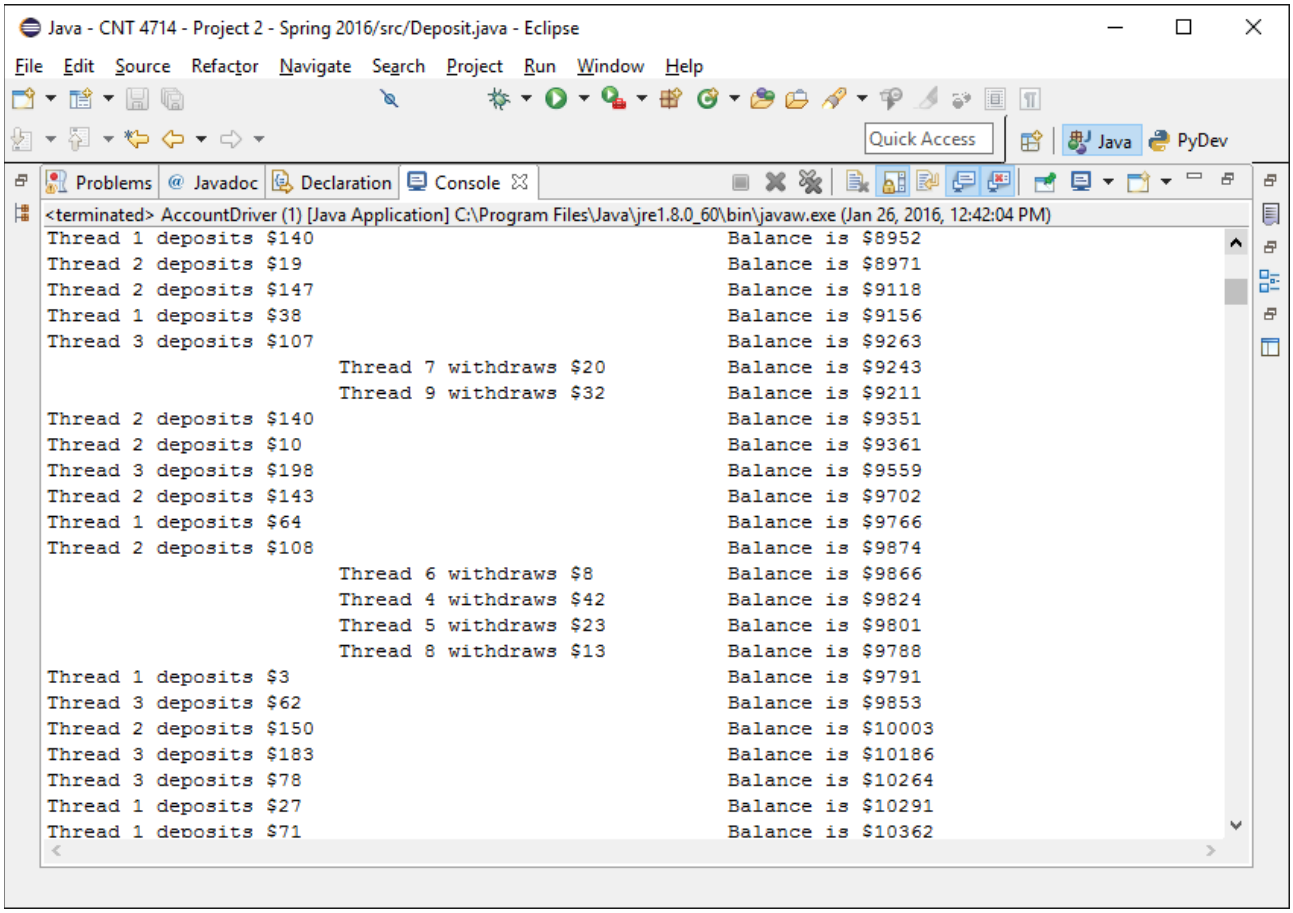
Additional Information:

Shown below are three example screen shots of the output from this program to help illustrate how your application is to operate and display the results. The last page illustrates execution runs that you do not want to produce.


Java - CNT 4714 - Project 2 - Spring 2016/src/Withdrawal.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
PyDev
Problems
Javadoc
Declaration
Console
<terminated> AccountDriver (1) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Jan 26, 2016, 12:28:44 PM)
Deposit Threads Withdrawal Threads Balance
Thread 1 deposits $81 Balance is $81
Thread 4 withdraws $35 Balance is $46
Thread 3 deposits $105 Balance is $151
Thread 2 deposits $95 Balance is $246
Thread 9 withdraws $3 Balance is $243
Thread 7 withdraws $4 Balance is $239
Thread 5 withdraws $45 Balance is $194
Thread 6 withdraws $26 Balance is $168
Thread 8 withdraws $28 Balance is $140
Thread 8 withdraws $40 Balance is $100
Thread 5 withdraws $7 Balance is $93
Thread 7 withdraws $10 Balance is $83
Thread 9 withdraws $19 Balance is $64
Thread 4 withdraws $17 Balance is $47
Thread 6 withdraws $36 Balance is $11
Thread 8 withdraws $28 Withdrawal - Blocked - Insufficient Funds
Thread 6 withdraws $14 Withdrawal - Blocked - Insufficient Funds
Thread 5 withdraws $34 Withdrawal - Blocked - Insufficient Funds
Thread 9 withdraws $35 Withdrawal - Blocked - Insufficient Funds
Thread 4 withdraws $48 Withdrawal - Blocked - Insufficient Funds
Thread 7 withdraws $1 Balance is $10
Thread 7 withdraws $2 Balance is $8
Thread 7 withdraws $15 Withdrawal - Blocked - Insufficient Funds
Thread 2 deposits $154 Balance is $162
Thread 1 deposits $151 Balance is $313
Thread 7 withdraws $44 Balance is $269
Thread 5 withdraws $13 Balance is $256
Thread 6 withdraws $19 Balance is $237
Thread 4 withdraws $13 Balance is $224
Thread 8 withdraws $1 Balance is $223
Thread 9 withdraws $44 Balance is $179
Thread 7 withdraws $7 Balance is $172
Thread 8 withdraws $1 Balance is $171
Thread 9 withdraws $35 Balance is $136
Thread 4 withdraws $41 Balance is $95
Thread 5 withdraws $15 Balance is $80


Java - CNT 4714 - Project 2 - Spring 2016/src/AccountDriver.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
PyDev
Problems
Javadoc
Declaration
Console
<terminated> AccountDriver (1) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Jan 26, 2016, 12:32:30 PM)
Deposit Threads Withdrawal Threads Balance
Thread 4 withdraws $8 Withdrawal - Blocked - Insufficient Funds
Thread 3 deposits $127 Balance is $127
Thread 5 withdraws $32 Balance is $95
Thread 1 deposits $3 Balance is $98
Thread 2 deposits $107 Balance is $205
Thread 6 withdraws $34 Balance is $171
Thread 7 withdraws $46 Balance is $125
Thread 9 withdraws $13 Balance is $112
Thread 8 withdraws $50 Balance is $62
Thread 7 withdraws $5 Balance is $57
Thread 8 withdraws $36 Balance is $21
Thread 9 withdraws $10 Balance is $11
Thread 6 withdraws $8 Balance is $3
Thread 4 withdraws $44 Withdrawal - Blocked - Insufficient Funds
Thread 5 withdraws $3 Balance is $0
Thread 9 withdraws $5 Withdrawal - Blocked - Insufficient Funds
Thread 8 withdraws $7 Withdrawal - Blocked - Insufficient Funds
Thread 5 withdraws $36 Withdrawal - Blocked - Insufficient Funds
Thread 6 withdraws $24 Withdrawal - Blocked - Insufficient Funds
Thread 7 withdraws $39 Withdrawal - Blocked - Insufficient Funds
Thread 3 deposits $150 Balance is $150
Thread 8 withdraws $28 Balance is $122
Thread 9 withdraws $3 Balance is $119
Thread 6 withdraws $40 Balance is $79
Thread 4 withdraws $48 Balance is $31
Thread 5 withdraws $46 Withdrawal - Blocked - Insufficient Funds
Thread 7 withdraws $8 Balance is $23
Thread 6 withdraws $35 Withdrawal - Blocked - Insufficient Funds
Thread 8 withdraws $19 Balance is $4
Thread 7 withdraws $9 Withdrawal - Blocked - Insufficient Funds
Thread 4 withdraws $40 Withdrawal - Blocked - Insufficient Funds
Thread 9 withdraws $8 Withdrawal - Blocked - Insufficient Funds
Thread 8 withdraws $46 Withdrawal - Blocked - Insufficient Funds
Thread 3 deposits $165 Balance is $169
Thread 7 withdraws $19 Balance is $150
Thread 4 withdraws $12 Balance is $138
Thread 5 withdraws $40 Balance is $98
Thread 9 withdraws $6 Balance is $92
Thread 6 withdraws $19 Balance is $73
Thread 8 withdraws $36 Balance is $37


Java - CNT 4714 - Project 2 - Spring 2016/src/AccountDriver.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
PyDev
Problems
Javadoc
Declaration
Console
<terminated> AccountDriver (1) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Jan 26, 2016, 12:34:15 PM)
Deposit Threads Withdrawal Threads Balance
Thread 9 withdraws $24 Withdrawal - Blocked - Insufficient Funds
Thread 7 withdraws $44 Withdrawal - Blocked - Insufficient Funds
Thread 8 withdraws $42 Withdrawal - Blocked - Insufficient Funds
Thread 4 withdraws $20 Withdrawal - Blocked - Insufficient Funds
Thread 3 deposits $13 Balance is $13
Thread 5 withdraws $23 Withdrawal - Blocked - Insufficient Funds
Thread 1 deposits $87 Balance is $100
Thread 6 withdraws $19 Balance is $81
Thread 2 deposits $53 Balance is $134
Thread 5 withdraws $19 Balance is $115
Thread 7 withdraws $15 Balance is $100
Thread 8 withdraws $50 Balance is $50
Thread 9 withdraws $49 Balance is $1
Thread 4 withdraws $46 Withdrawal - Blocked - Insufficient Funds
Thread 6 withdraws $9 Withdrawal - Blocked - Insufficient Funds
Thread 8 withdraws $17 Withdrawal - Blocked - Insufficient Funds
Thread 7 withdraws $47 Withdrawal - Blocked - Insufficient Funds
Thread 9 withdraws $25 Withdrawal - Blocked - Insufficient Funds
Thread 5 withdraws $38 Withdrawal - Blocked - Insufficient Funds
Thread 2 deposits $116 Balance is $117
Thread 8 withdraws $26 Balance is $91
Thread 7 withdraws $2 Balance is $89
Thread 5 withdraws $36 Balance is $53
Thread 4 withdraws $46 Balance is $7
Thread 9 withdraws $16 Withdrawal - Blocked - Insufficient Funds
Thread 6 withdraws $34 Withdrawal - Blocked - Insufficient Funds
Thread 8 withdraws $15 Withdrawal - Blocked - Insufficient Funds
Thread 5 withdraws $41 Withdrawal - Blocked - Insufficient Funds
Thread 7 withdraws $3 Balance is $4
Thread 4 withdraws $24 Withdrawal - Blocked - Insufficient Funds
Thread 1 deposits $120 Balance is $124
Thread 7 withdraws $43 Balance is $81
Thread 6 withdraws $24 Balance is $57
Thread 4 withdraws $2 Balance is $55
Thread 5 withdraws $19 Balance is $36
Thread 9 withdraws $17 Balance is $19
Thread 8 withdraws $18 Balance is $1
Thread 7 withdraws $2 Withdrawal - Blocked - Insufficient Funds
Thread 4 withdraws $31 Withdrawal - Blocked - Insufficient Funds
Thread 6 withdraws $32 Withdrawal - Blocked - Insufficient Funds



We don't want to see this sort of scenario where the depositors are monopolizing the account. Indication is the depositor threads aren't sleeping long enough.