CNT 4714 – Project 2 – Fall 2019

Title: "Project 2: An Application Employing Synchronized/Cooperating Multiple Threads In Java Using Locks – A Banking Simulator"
Points: 100 points
Due Date: Sunday October 6, 2019 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 $250 (whole dollars only) and withdrawals are made in amounts ranging from $1 to $50 (again, whole dollars only).

2. You should have four deposit threads and eight withdrawal threads simultaneously executing.

3. Once a deposit thread has executed, put it to sleep for few milliseconds (randomly generate this number – don't use a constant sleep time) 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 a single withdrawal thread 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. This would most likely occur when the withdrawal threads are sleeping too long in comparison to the average sleep time of the deposit threads. See page 7 for an illustration of this.

5. Assume all threads have the same priority. Do not give different priority to depositor and withdrawal threads.

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. Just stop the simulation from your IDE after a few seconds.

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

9. You must utilize a reentrant lock from the java.util.concurrent.locks package for implementing your locking protocols. We will specify no fairness policy for this application. Do not create your own lock using a Boolean or any other type of variable.

References:
Notes: Lecture Notes for Multithreaded Applications.

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

```
/* Name:
   Course: CNT 4714 Fall 2019
   Assignment title: Project 2 – Synchronized, Cooperating Threads Under Locking
   Due Date: October 6, 2019
*/
```

Input Specification: Internal to the program.

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

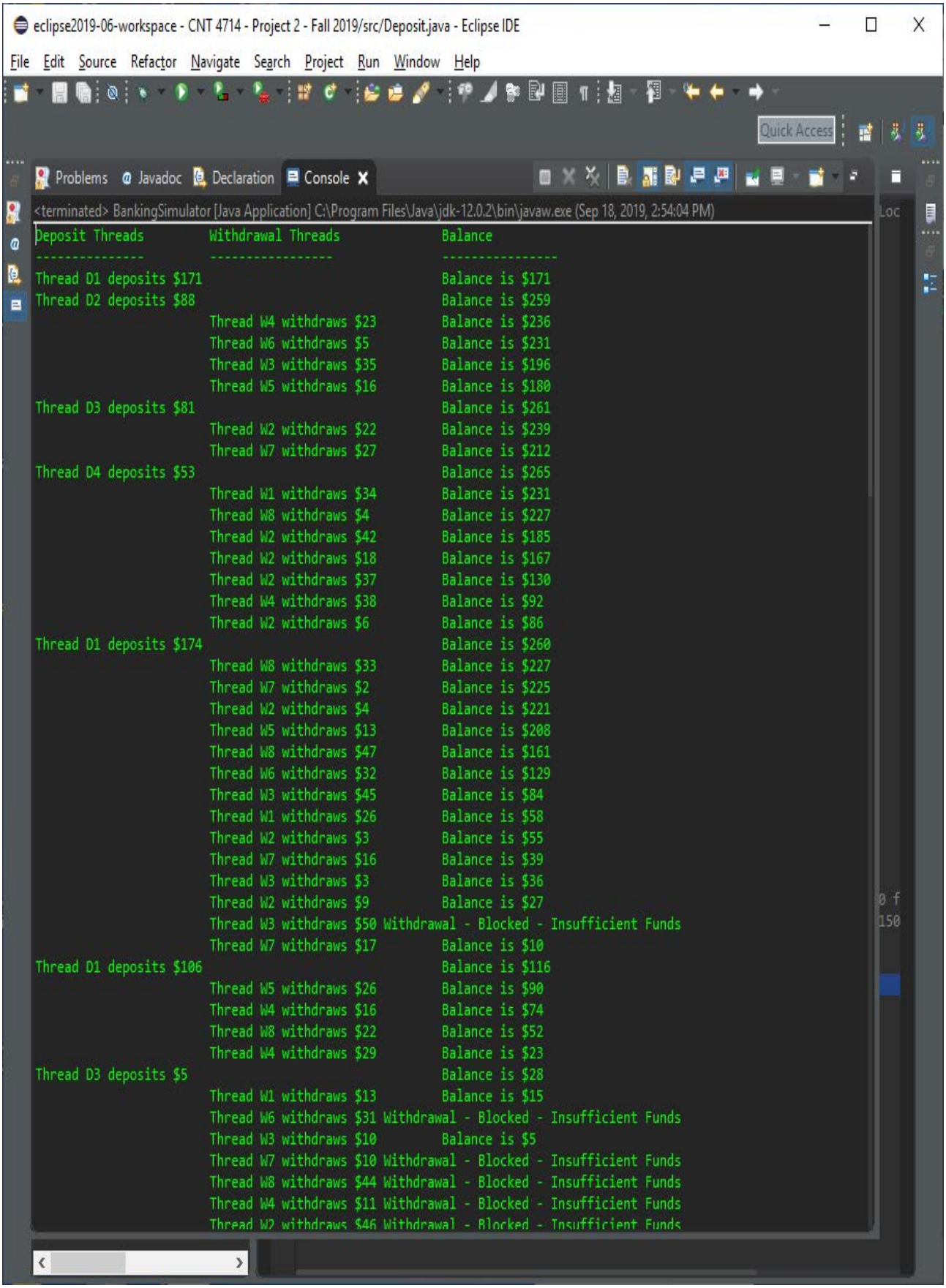
Deliverables:

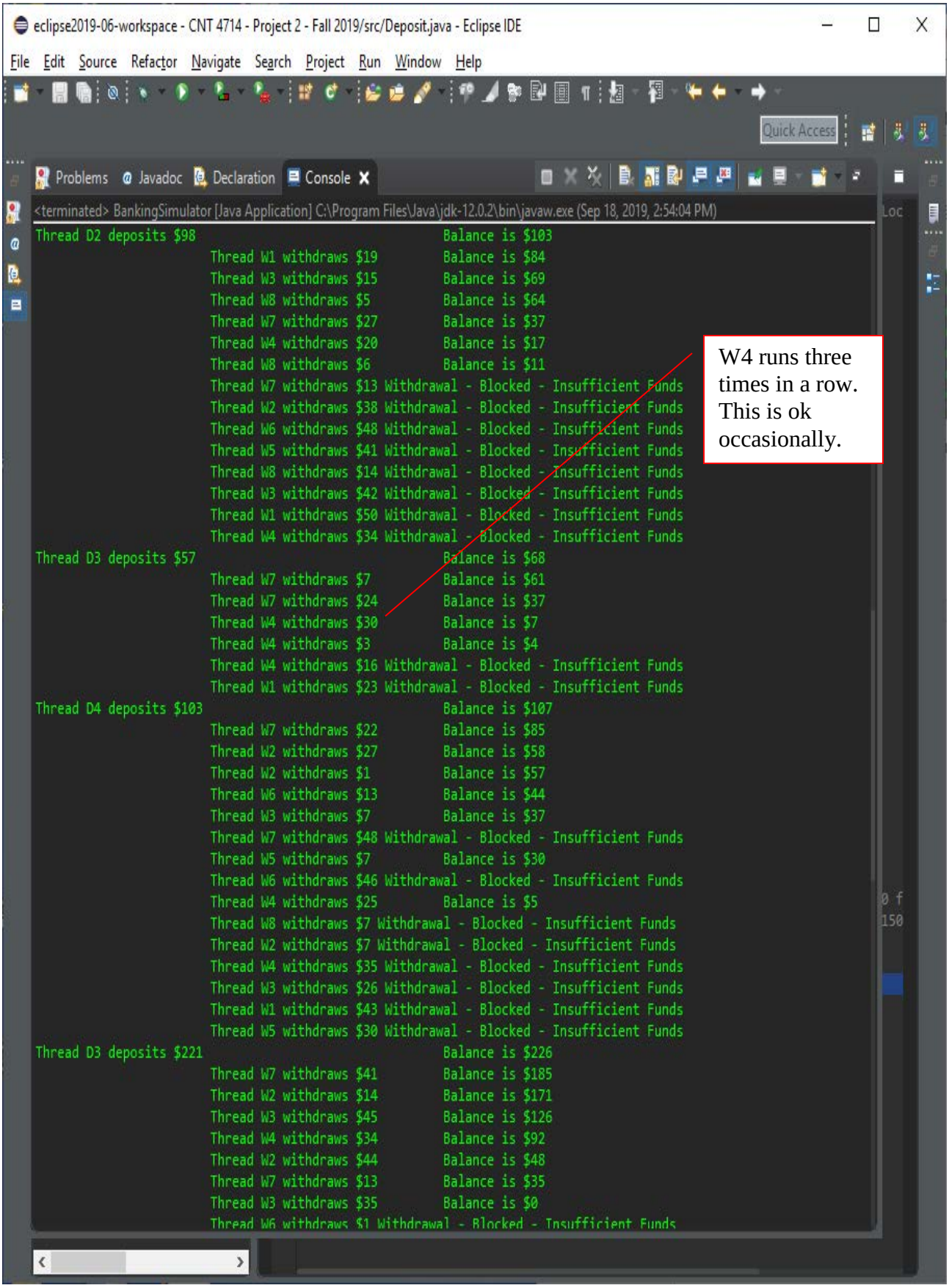
(1) Zip up all of your .java files and submit them via WebCourses no later than 11:59pm Sunday October 6, 2019.

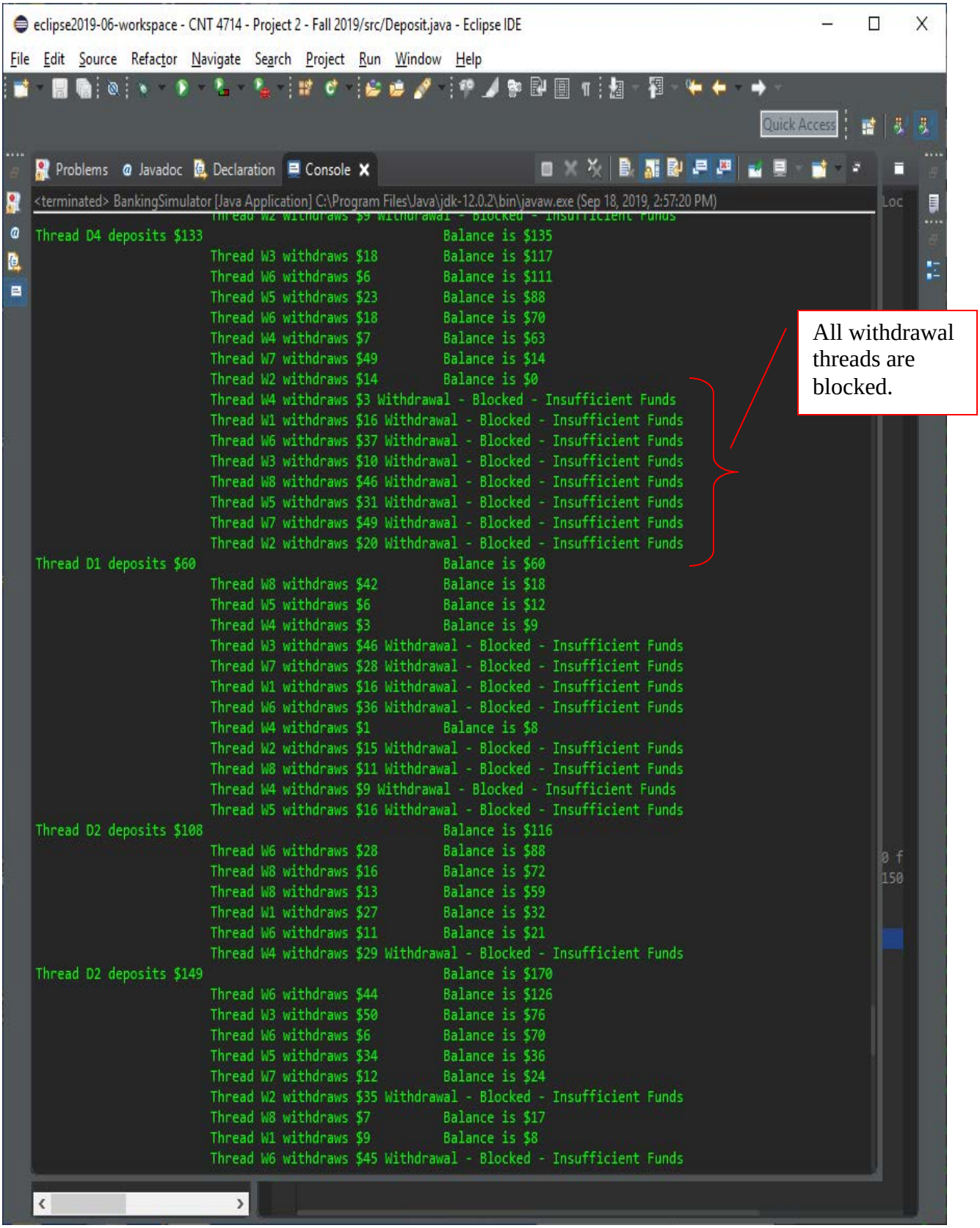
(2) Include at least one screen shot which illustrates the execution of your synchronized threaded application. See below for some representative examples. You can either do a screen shot of the console window like I did below or redirect your output to a file and take a screen shot from an editor.

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.


eclipse2019-06-workspace - CNT 4714 - Project 2 - Fall 2019/src/Deposit.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Problems Javadoc Declaration Console
<terminated> BankingSimulator [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (Sep 18, 2019, 2:54:04 PM)
Deposit Threads Withdrawal Threads Balance
Thread D1 deposits $171 Balance is $171
Thread D2 deposits $88 Balance is $259
Thread W4 withdraws $23 Balance is $236
Thread W6 withdraws $5 Balance is $231
Thread W3 withdraws $35 Balance is $196
Thread W5 withdraws $16 Balance is $180
Thread D3 deposits $81 Balance is $261
Thread W2 withdraws $22 Balance is $239
Thread W7 withdraws $27 Balance is $212
Thread D4 deposits $53 Balance is $265
Thread W1 withdraws $34 Balance is $231
Thread W8 withdraws $4 Balance is $227
Thread W2 withdraws $42 Balance is $185
Thread W2 withdraws $18 Balance is $167
Thread W2 withdraws $37 Balance is $130
Thread W4 withdraws $38 Balance is $92
Thread W2 withdraws $6 Balance is $86
Thread D1 deposits $174 Balance is $260
Thread W8 withdraws $33 Balance is $227
Thread W7 withdraws $2 Balance is $225
Thread W2 withdraws $4 Balance is $221
Thread W5 withdraws $13 Balance is $208
Thread W8 withdraws $47 Balance is $161
Thread W6 withdraws $32 Balance is $129
Thread W3 withdraws $45 Balance is $84
Thread W1 withdraws $26 Balance is $58
Thread W2 withdraws $3 Balance is $55
Thread W7 withdraws $16 Balance is $39
Thread W3 withdraws $3 Balance is $36
Thread W2 withdraws $9 Balance is $27
Thread W3 withdraws $50 Withdrawal - Blocked - Insufficient Funds
Thread W7 withdraws $17 Balance is $10
Thread D1 deposits $106 Balance is $116
Thread W5 withdraws $26 Balance is $90
Thread W4 withdraws $16 Balance is $74
Thread W8 withdraws $22 Balance is $52
Thread W4 withdraws $29 Balance is $23
Thread D3 deposits $5 Balance is $28
Thread W1 withdraws $13 Balance is $15
Thread W6 withdraws $31 Withdrawal - Blocked - Insufficient Funds
Thread W3 withdraws $10 Balance is $5
Thread W7 withdraws $10 Withdrawal - Blocked - Insufficient Funds
Thread W8 withdraws $44 Withdrawal - Blocked - Insufficient Funds
Thread W4 withdraws $11 Withdrawal - Blocked - Insufficient Funds
Thread W2 withdraws $46 Withdrawal - Blocked - Insufficient Funds


eclipse2019-06-workspace - CNT 4714 - Project 2 - Fall 2019/src/Deposit.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Problems Javadoc Declaration Console
<terminated> BankingSimulator [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (Sep 18, 2019, 2:54:04 PM)
Thread D2 deposits $98 Balance is $103
Thread W1 withdraws $19 Balance is $84
Thread W3 withdraws $15 Balance is $69
Thread W8 withdraws $5 Balance is $64
Thread W7 withdraws $27 Balance is $37
Thread W4 withdraws $20 Balance is $17
Thread W8 withdraws $6 Balance is $11
Thread W7 withdraws $13 Withdrawal - Blocked - Insufficient Funds
Thread W2 withdraws $38 Withdrawal - Blocked - Insufficient Funds
Thread W6 withdraws $48 Withdrawal - Blocked - Insufficient Funds
Thread W5 withdraws $41 Withdrawal - Blocked - Insufficient Funds
Thread W8 withdraws $14 Withdrawal - Blocked - Insufficient Funds
Thread W3 withdraws $42 Withdrawal - Blocked - Insufficient Funds
Thread W1 withdraws $50 Withdrawal - Blocked - Insufficient Funds
Thread W4 withdraws $34 Withdrawal - Blocked - Insufficient Funds
Thread D3 deposits $57 Balance is $68
Thread W7 withdraws $7 Balance is $61
Thread W7 withdraws $24 Balance is $37
Thread W4 withdraws $30 Balance is $7
Thread W4 withdraws $3 Balance is $4
Thread W4 withdraws $16 Withdrawal - Blocked - Insufficient Funds
Thread W1 withdraws $23 Withdrawal - Blocked - Insufficient Funds
Thread D4 deposits $103 Balance is $107
Thread W7 withdraws $22 Balance is $85
Thread W2 withdraws $27 Balance is $58
Thread W2 withdraws $1 Balance is $57
Thread W6 withdraws $13 Balance is $44
Thread W3 withdraws $7 Balance is $37
Thread W7 withdraws $48 Withdrawal - Blocked - Insufficient Funds
Thread W5 withdraws $7 Balance is $30
Thread W6 withdraws $46 Withdrawal - Blocked - Insufficient Funds
Thread W4 withdraws $25 Balance is $5
Thread W8 withdraws $7 Withdrawal - Blocked - Insufficient Funds
Thread W2 withdraws $7 Withdrawal - Blocked - Insufficient Funds
Thread W4 withdraws $35 Withdrawal - Blocked - Insufficient Funds
Thread W3 withdraws $26 Withdrawal - Blocked - Insufficient Funds
Thread W1 withdraws $43 Withdrawal - Blocked - Insufficient Funds
Thread W5 withdraws $30 Withdrawal - Blocked - Insufficient Funds
Thread D3 deposits $221 Balance is $226
Thread W7 withdraws $41 Balance is $185
Thread W2 withdraws $14 Balance is $171
Thread W3 withdraws $45 Balance is $126
Thread W4 withdraws $34 Balance is $92
Thread W2 withdraws $44 Balance is $48
Thread W7 withdraws $13 Balance is $35
Thread W3 withdraws $35 Balance is $0
Thread W6 withdraws $1 Withdrawal - Blocked - Insufficient Funds
W4 runs three times in a row. This is ok occasionally.


eclipse2019-06-workspace - CNT 4714 - Project 2 - Fall 2019/src/Deposit.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Problems Javadoc Declaration Console
<terminated> BankingSimulator [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (Sep 18, 2019, 2:57:20 PM)
Thread D4 deposits $133 Balance is $135
Thread W3 withdraws $18 Balance is $117
Thread W6 withdraws $6 Balance is $111
Thread W5 withdraws $23 Balance is $88
Thread W6 withdraws $18 Balance is $70
Thread W4 withdraws $7 Balance is $63
Thread W7 withdraws $49 Balance is $14
Thread W2 withdraws $14 Balance is $0
Thread W4 withdraws $3 Withdrawal - Blocked - Insufficient Funds
Thread W1 withdraws $16 Withdrawal - Blocked - Insufficient Funds
Thread W6 withdraws $37 Withdrawal - Blocked - Insufficient Funds
Thread W3 withdraws $10 Withdrawal - Blocked - Insufficient Funds
Thread W8 withdraws $46 Withdrawal - Blocked - Insufficient Funds
Thread W5 withdraws $31 Withdrawal - Blocked - Insufficient Funds
Thread W7 withdraws $49 Withdrawal - Blocked - Insufficient Funds
Thread W2 withdraws $20 Withdrawal - Blocked - Insufficient Funds
Thread D1 deposits $60 Balance is $60
Thread W8 withdraws $42 Balance is $18
Thread W5 withdraws $6 Balance is $12
Thread W4 withdraws $3 Balance is $9
Thread W3 withdraws $46 Withdrawal - Blocked - Insufficient Funds
Thread W7 withdraws $28 Withdrawal - Blocked - Insufficient Funds
Thread W1 withdraws $16 Withdrawal - Blocked - Insufficient Funds
Thread W6 withdraws $36 Withdrawal - Blocked - Insufficient Funds
Thread W4 withdraws $1 Balance is $8
Thread W2 withdraws $15 Withdrawal - Blocked - Insufficient Funds
Thread W8 withdraws $11 Withdrawal - Blocked - Insufficient Funds
Thread W4 withdraws $9 Withdrawal - Blocked - Insufficient Funds
Thread W5 withdraws $16 Withdrawal - Blocked - Insufficient Funds
Thread D2 deposits $108 Balance is $116
Thread W6 withdraws $28 Balance is $88
Thread W8 withdraws $16 Balance is $72
Thread W8 withdraws $13 Balance is $59
Thread W1 withdraws $27 Balance is $32
Thread W6 withdraws $11 Balance is $21
Thread W4 withdraws $29 Withdrawal - Blocked - Insufficient Funds
Thread D2 deposits $149 Balance is $170
Thread W6 withdraws $44 Balance is $126
Thread W3 withdraws $50 Balance is $76
Thread W6 withdraws $6 Balance is $70
Thread W5 withdraws $34 Balance is $36
Thread W7 withdraws $12 Balance is $24
Thread W2 withdraws $35 Withdrawal - Blocked - Insufficient Funds
Thread W8 withdraws $7 Balance is $17
Thread W1 withdraws $9 Balance is $8
Thread W6 withdraws $45 Withdrawal - Blocked - Insufficient Funds
All withdrawal threads are blocked.

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.