CNT 4714 – Project 2 – Summer 2020

Title: "Project 2: An Application Employing Synchronized/Cooperating Multiple Threads In Java Using Locks – A Banking Simulator"
Points: 100 points
Due Date: Sunday June 14, 2020 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 depositor threads signal all waiting withdrawal threads whenever a deposit is completed.

1. You should have five depositor threads and nine withdrawal threads simultaneously executing.

2. 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). Since we have more withdrawal threads than depositor threads, the account balance should constantly decrease over time.

3. Once a depositor 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. So occasional back-to-back runs of withdrawal threads might occur.
    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 Summer 2020
    Assignment title: Project 2 – Synchronized, Cooperating Threads Under Locking
    Due Date: June 14, 2020
*/
```

Input Specification: Internal to the program.

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

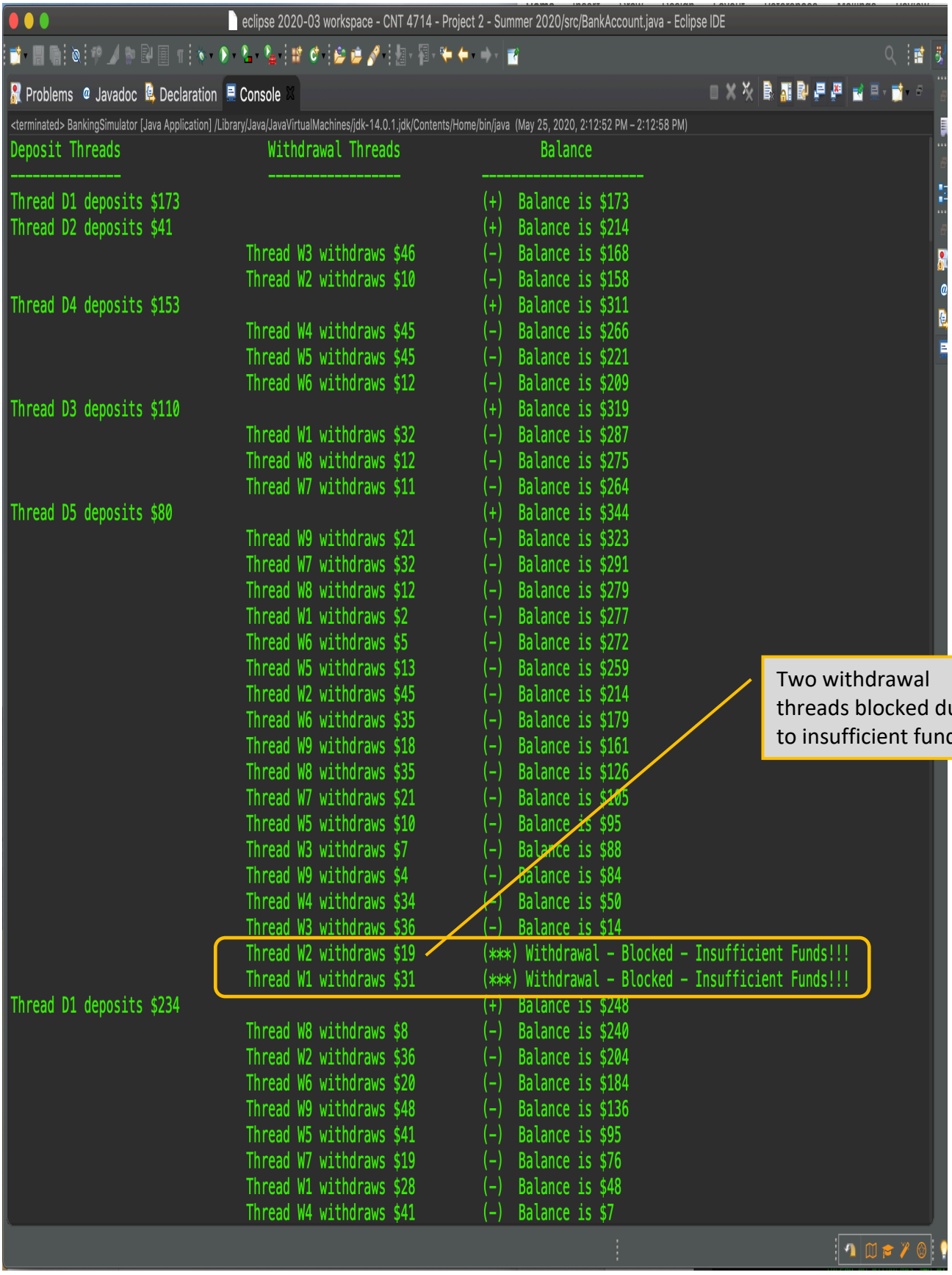
Deliverables:

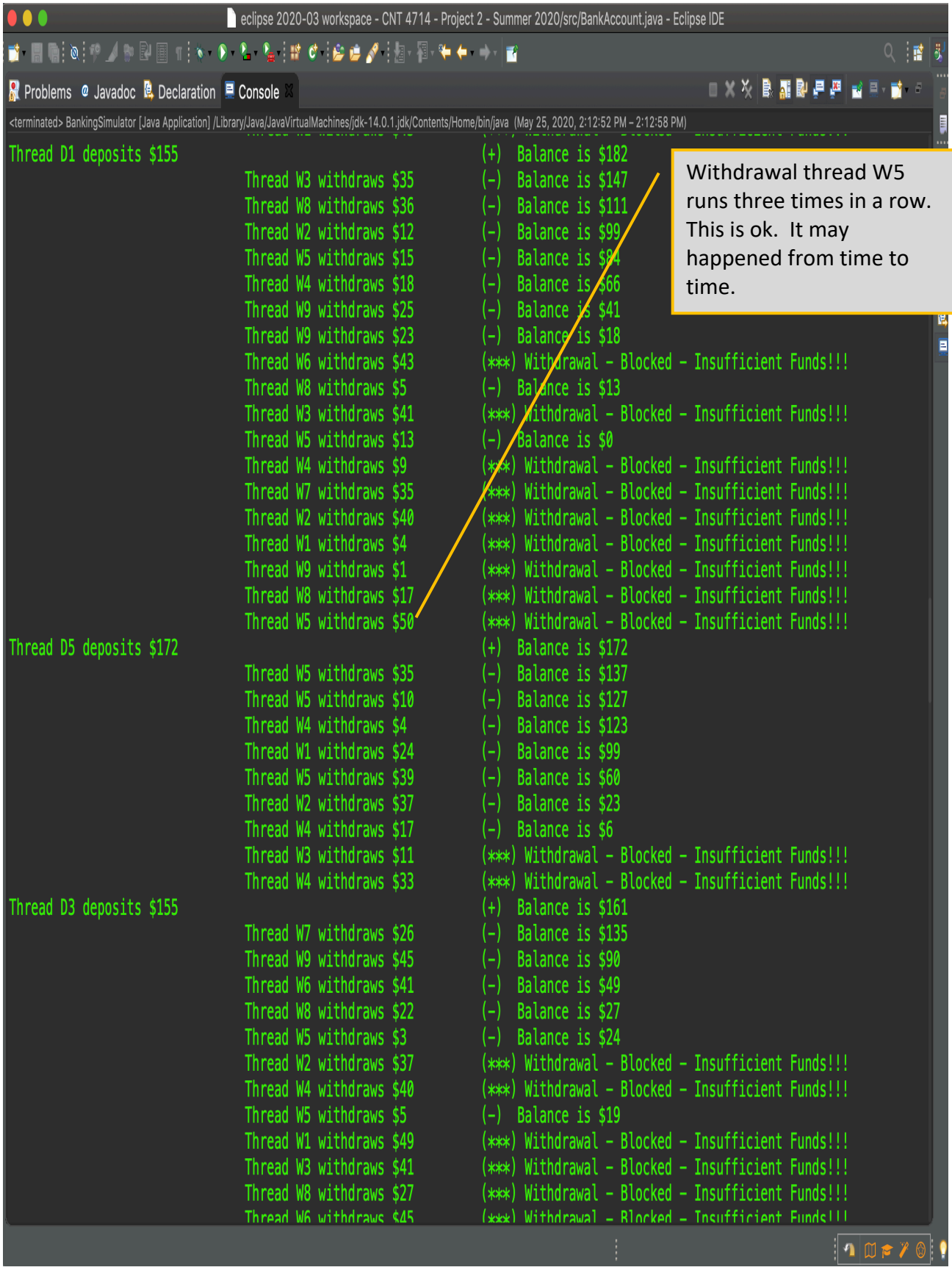
(1) Zip up all of your .java files and submit them via WebCourses no later than 11:59pm Sunday June 14, 2020.

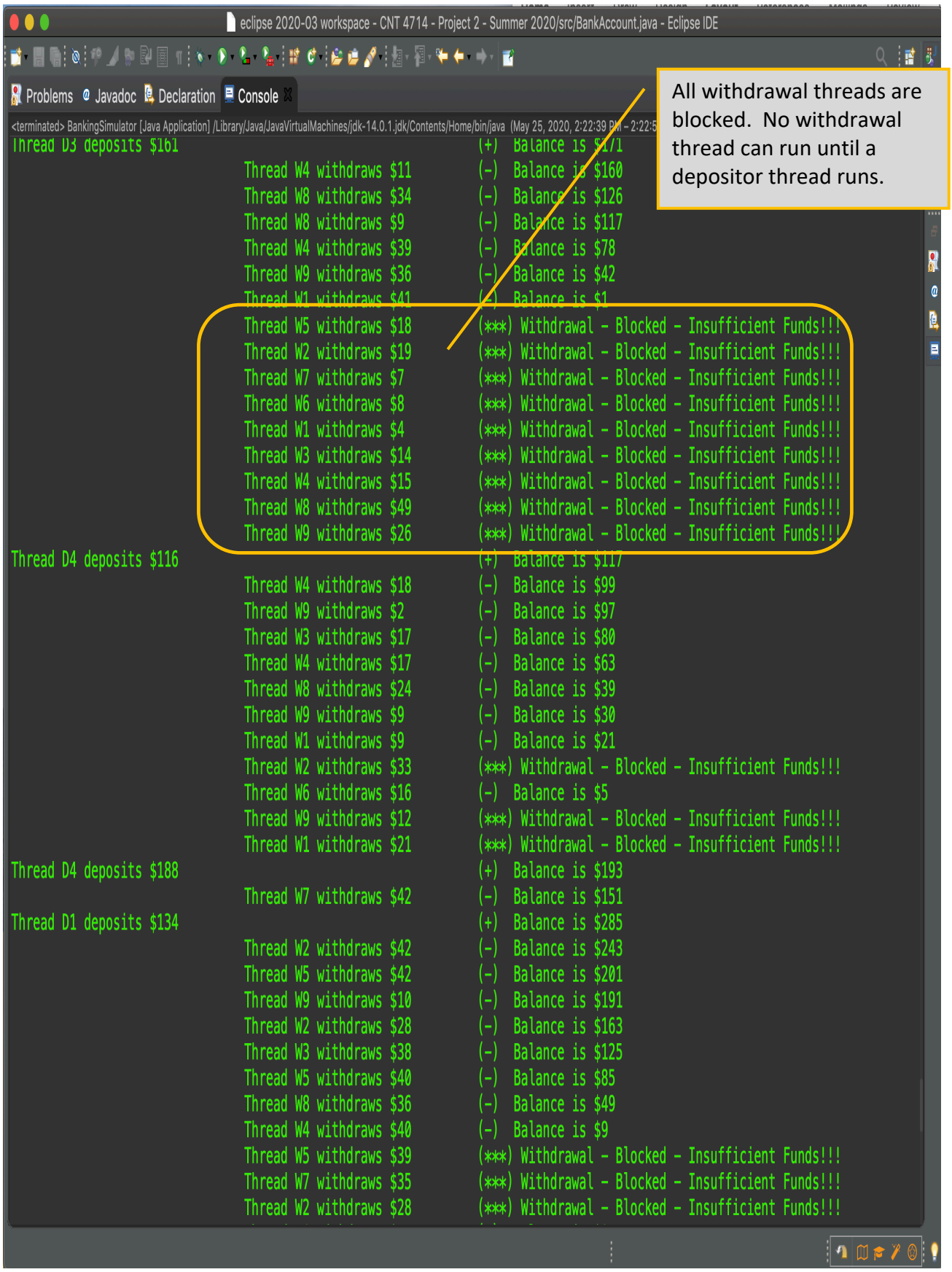
(2) Include at least one screen shot which illustrates the execution of your synchronized threaded application. See below for some representative examples. You can either do a screen shot of the console window like I did below or redirect your output to a file and take a screen shot from an editor.

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.


eclipse 2020-03 workspace - CNT 4714 - Project 2 - Summer 2020/src/BankAccount.java - Eclipse IDE
Problems
Javadoc
Declaration
Console
<terminated> BankingSimulator [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java (May 25, 2020, 2:12:52 PM – 2:12:58 PM)
Deposit Threads
Withdrawal Threads
Balance
Thread D1 deposits $173 (+) Balance is $173
Thread D2 deposits $41 (+) Balance is $214
Thread W3 withdraws $46 (-) Balance is $168
Thread W2 withdraws $10 (-) Balance is $158
Thread D4 deposits $153 (+) Balance is $311
Thread W4 withdraws $45 (-) Balance is $266
Thread W5 withdraws $45 (-) Balance is $221
Thread W6 withdraws $12 (-) Balance is $209
Thread D3 deposits $110 (+) Balance is $319
Thread W1 withdraws $32 (-) Balance is $287
Thread W8 withdraws $12 (-) Balance is $275
Thread W7 withdraws $11 (-) Balance is $264
Thread D5 deposits $80 (+) Balance is $344
Thread W9 withdraws $21 (-) Balance is $323
Thread W7 withdraws $32 (-) Balance is $291
Thread W8 withdraws $12 (-) Balance is $279
Thread W1 withdraws $2 (-) Balance is $277
Thread W6 withdraws $5 (-) Balance is $272
Thread W5 withdraws $13 (-) Balance is $259
Thread W2 withdraws $45 (-) Balance is $214
Thread W6 withdraws $35 (-) Balance is $179
Thread W9 withdraws $18 (-) Balance is $161
Thread W8 withdraws $35 (-) Balance is $126
Thread W7 withdraws $21 (-) Balance is $105
Thread W5 withdraws $10 (-) Balance is $95
Thread W3 withdraws $7 (-) Balance is $88
Thread W9 withdraws $4 (-) Balance is $84
Thread W4 withdraws $34 (-) Balance is $50
Thread W3 withdraws $36 (-) Balance is $14
Thread W2 withdraws $19 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W1 withdraws $31 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread D1 deposits $234 (+) Balance is $248
Thread W8 withdraws $8 (-) Balance is $240
Thread W2 withdraws $36 (-) Balance is $204
Thread W6 withdraws $20 (-) Balance is $184
Thread W9 withdraws $48 (-) Balance is $136
Thread W5 withdraws $41 (-) Balance is $95
Thread W7 withdraws $19 (-) Balance is $76
Thread W1 withdraws $28 (-) Balance is $48
Thread W4 withdraws $41 (-) Balance is $7
Two withdrawal threads blocked due to insufficient funds.


eclipse 2020-03 workspace - CNT 4714 - Project 2 - Summer 2020/src/BankAccount.java - Eclipse IDE
Problems
Javadoc
Declaration
Console
<terminated> BankingSimulator [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java (May 25, 2020, 2:12:52 PM – 2:12:58 PM)
Thread D1 deposits $155 (+) Balance is $182
Thread W3 withdraws $35 (-) Balance is $147
Thread W8 withdraws $36 (-) Balance is $111
Thread W2 withdraws $12 (-) Balance is $99
Thread W5 withdraws $15 (-) Balance is $84
Thread W4 withdraws $18 (-) Balance is $66
Thread W9 withdraws $25 (-) Balance is $41
Thread W9 withdraws $23 (-) Balance is $18
Thread W6 withdraws $43 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W8 withdraws $5 (-) Balance is $13
Thread W3 withdraws $41 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W5 withdraws $13 (-) Balance is $0
Thread W4 withdraws $9 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W7 withdraws $35 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W2 withdraws $40 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W1 withdraws $4 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W9 withdraws $1 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W8 withdraws $17 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W5 withdraws $50 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread D5 deposits $172 (+) Balance is $172
Thread W5 withdraws $35 (-) Balance is $137
Thread W5 withdraws $10 (-) Balance is $127
Thread W4 withdraws $4 (-) Balance is $123
Thread W1 withdraws $24 (-) Balance is $99
Thread W5 withdraws $39 (-) Balance is $60
Thread W2 withdraws $37 (-) Balance is $23
Thread W4 withdraws $17 (-) Balance is $6
Thread W3 withdraws $11 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W4 withdraws $33 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread D3 deposits $155 (+) Balance is $161
Thread W7 withdraws $26 (-) Balance is $135
Thread W9 withdraws $45 (-) Balance is $90
Thread W6 withdraws $41 (-) Balance is $49
Thread W8 withdraws $22 (-) Balance is $27
Thread W5 withdraws $3 (-) Balance is $24
Thread W2 withdraws $37 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W4 withdraws $40 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W5 withdraws $5 (-) Balance is $19
Thread W1 withdraws $49 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W3 withdraws $41 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W8 withdraws $27 (***) Withdrawal - Blocked - Insufficient Funds!!!
Withdrawal thread W5 runs three times in a row. This is ok. It may happened from time to time.


eclipse 2020-03 workspace - CNT 4714 - Project 2 - Summer 2020/src/BankAccount.java - Eclipse IDE
Problems
Javadoc
Declaration
Console
<terminated> BankingSimulator [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java (May 25, 2020, 2:22:39 PM – 2:22:5
Thread D3 deposits $161 (+) Balance is $171
Thread W4 withdraws $11 (-) Balance is $160
Thread W8 withdraws $34 (-) Balance is $126
Thread W8 withdraws $9 (-) Balance is $117
Thread W4 withdraws $39 (-) Balance is $78
Thread W9 withdraws $36 (-) Balance is $42
Thread W1 withdraws $41 (-) Balance is $1
Thread W5 withdraws $18 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W2 withdraws $19 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W7 withdraws $7 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W6 withdraws $8 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W1 withdraws $4 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W3 withdraws $14 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W4 withdraws $15 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W8 withdraws $49 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W9 withdraws $26 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread D4 deposits $116 (+) Balance is $117
Thread W4 withdraws $18 (-) Balance is $99
Thread W9 withdraws $2 (-) Balance is $97
Thread W3 withdraws $17 (-) Balance is $80
Thread W4 withdraws $17 (-) Balance is $63
Thread W8 withdraws $24 (-) Balance is $39
Thread W9 withdraws $9 (-) Balance is $30
Thread W1 withdraws $9 (-) Balance is $21
Thread W2 withdraws $33 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W6 withdraws $16 (-) Balance is $5
Thread W9 withdraws $12 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W1 withdraws $21 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread D4 deposits $188 (+) Balance is $193
Thread W7 withdraws $42 (-) Balance is $151
Thread D1 deposits $134 (+) Balance is $285
Thread W2 withdraws $42 (-) Balance is $243
Thread W5 withdraws $42 (-) Balance is $201
Thread W9 withdraws $10 (-) Balance is $191
Thread W2 withdraws $28 (-) Balance is $163
Thread W3 withdraws $38 (-) Balance is $125
Thread W5 withdraws $40 (-) Balance is $85
Thread W8 withdraws $36 (-) Balance is $49
Thread W4 withdraws $40 (-) Balance is $9
Thread W5 withdraws $39 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W7 withdraws $35 (***) Withdrawal - Blocked - Insufficient Funds!!!
Thread W2 withdraws $28 (***) Withdrawal - Blocked - Insufficient Funds!!!
All withdrawal threads are blocked. No withdrawal thread can run until a depositor thread runs.

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 or the withdrawal threads are sleeping too long.