CNT 4714 – Project 2 – Fall 2024

Title: "Project 2: An Application Employing Synchronized/Cooperating Multiple Threads In Java Using Locks – A Banking Simulator"
Points: 100 points
Due Date: Sunday September 22, 2024 by 11:59 pm (WebCourses time)

Objectives: To develop an application which requires cooperating, synchronized multiple threads of execution.

Project Overview: In this project you will simulate the deposits and withdrawals made to fictitious bank accounts (I'll let you use my real bank accounts if you promise to make only deposits! ☺). In this case the deposits and withdrawals will be made by user agents (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 agents (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 agent attempts to withdraw an amount greater than the current balance in the account – then it must block itself and wait until a depositing agent has added money to the account before it can try again. As we covered in the lecture notes, this will require that the depositing agents signal all waiting withdrawing agents whenever a deposit is completed.

Agents: There are five different types of agents in this simulation as follows:

1. Depositor Agents: These agents add(+) money to a bank account. To keep things relatively simple, assume that deposits are made in random amounts ranging from $1 to $600 (whole dollars only), Depositor agents never block.
2. Withdrawal Agents: These agents remove(-) money from a bank account. Withdrawal agents will make withdrawals in random amounts ranging from $1 to $99 (again, whole dollars only). Withdrawal agents block if the balance of the account in question is less than the amount the withdrawal

agent is attempting to withdraw. The withdrawal agent will remain blocked until signalled that a deposit has been made.

3. Transfer Agents: These agents withdraw(-) money from one account and deposit(+) the entire amount withdrawn into another account. This is an atomic operation, meaning that it is not treated as a withdrawal followed by a deposit. Hence, both accounts must be locked during the entire transfer operation. If the amount of money in the account from which the money is being withdrawn is less than the current balance in that account, then the transfer is aborted. If both accounts are not available at the time the transfer agent executes, then the transfer agent will simply try again later. Transfer agents do not block.
4. Internal Audit Agents: These agents belong to the bank and are run periodically to simply verify the current balance in all accounts. Similar to Transfer Agents, Internal Audit Agents are atomic operations that require holding locks on all accounts simultaneously. If all locks are not available, then any locks held by the Internal Audit Agents are released and it simply tries again later. Internal Audit Agents do not block.
5. U.S. Department of the Treasury Agents: These agents belong to the U.S. government and are run periodically to simply verify the current balance in all accounts owned by the bank. Similar to Internal Audit Agents, Treasury Agents are atomic operations that requiring holding locks on all accounts simultaneously. If all locks are not available, then any locks held by the Treasury Agents are released and it simply tries again later. Treasury Agents do not block.

Frequency of Agent Execution: We are simulating random events on bank accounts in the form of deposits, withdrawals, transfers, and audits. These events occur at random times. To simulate the randomness of these events, all agents are running in infinite loops. The loops are simple for each agent: perform activity, sleep, repeat. The randomness of event timing is determined by a random sleeping interval for each agent. We will discuss appropriate times in the Q&A sessions, but for our simulation, withdrawal agents will run most frequently (shortest sleep times), followed by depositor agents, followed by transfer agents, followed by internal audit agents, followed by treasury agents (longest sleep times).

Number of Agents: You should have five depositor agents, ten withdrawal agents, two transfer agents, one internal audit agent, and one treasury agent

simultaneously executing. Use a FixedThreadPool() and an Executor object to control the threads.

Simulation Parameters:

1. You will have two bank accounts to control in this simulation. Each agent will randomly select one of the accounts to operate on with equal probability.
2. Start the simulation with a balance of $0 in each bank account.
3. The account balances should constantly decrease over time. This will lead to withdrawal agents repeatedly blocking for insufficient funds. We do not want to see a situation where a bank account balance is constantly increasing. (See examples below that illustrate this situation.)
4. Do not use a constant (fixed) sleep time for any agent. All sleep times should be randomly generated between 0 ms and MAXSLEEP ms. As mentioned before, MAXSLEEP will vary for the different agents.
5. You should never see the situation where (a) one agent monopolizes operations and performs many operations in sequence before other agents operate (examples illustrated below).
6. Since we are constantly bleeding the accounts down and withdrawal agents (and transfer agents) block on insufficient funds, you will occasionally see the scenario where all withdrawing agents may be blocked at the same time.
7. All agents have the same priority. DO NOT ALTER the priority of any agent. Operational frequency is controlled simply be varied sleep times.
8. DO NOT use counted loops for your agents. All agents are running infinitely.
9. DO NOT use the Java synchronized modifier. No monitors.
10. 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.**
11. The Money Laundering Suppression Act, enacted by Congress in 1994, is a policy regulation that requires banking institutions to file currency transaction reports (CTRs) with the federal government (Department of the Treasury) for any deposits of $10,000 or more into a bank account. You are going to simulate this process by flagging any depositing transaction with a deposit value greater than $450.00 and any withdrawal amount greater than $90.00. You will flag the transaction in the normal output of the simulation as well as making an entry into a transaction log file (transactions.csv) which will keep track of all flagged transactions independently of the simulation. Each entry in the flagged transaction file will contain the transaction details, a timestamp

(the date and time at which the transaction occurred), and the transaction number. See below for more details.

12. The output from your program must look reasonably similar to the sample output shown below. The simulation output should show the action of each agent along with the account balance produced by the agent's transaction and the transaction number.
13. Every successful transaction made by a depositor, withdrawal, or transfer agent will have a transaction number (an integer initialized to 1). This transaction number is printed out in the simulation run with each completed transaction. See output examples below.
14. The Internal Auditor agent and the Treasury Auditor agent, simply verify the current balance in the accounts at random intervals and indicate how many transactions have occurred since the last audit of its type. The auditor agents do not make transactions on the account and do not affect the transaction number sequence. The auditor agents simply print the current account balance in each account into the simulation run and keep track of the number of transactions that have executed since the last auditor execution of that type. Note that the auditors should run much less frequently than either depositor or withdrawal agents (see above).

References:

Notes: Lecture Notes for Multithreaded Applications.

Restrictions:

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

```
/*  Name:
   Course: CNT 4714 Fall 2024
   Assignment title:
         Project 2 – Synchronized/Cooperating Threads – A  Banking Simulation
   Due Date: September 22, 2024
*/
```

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 September 22, 2024.

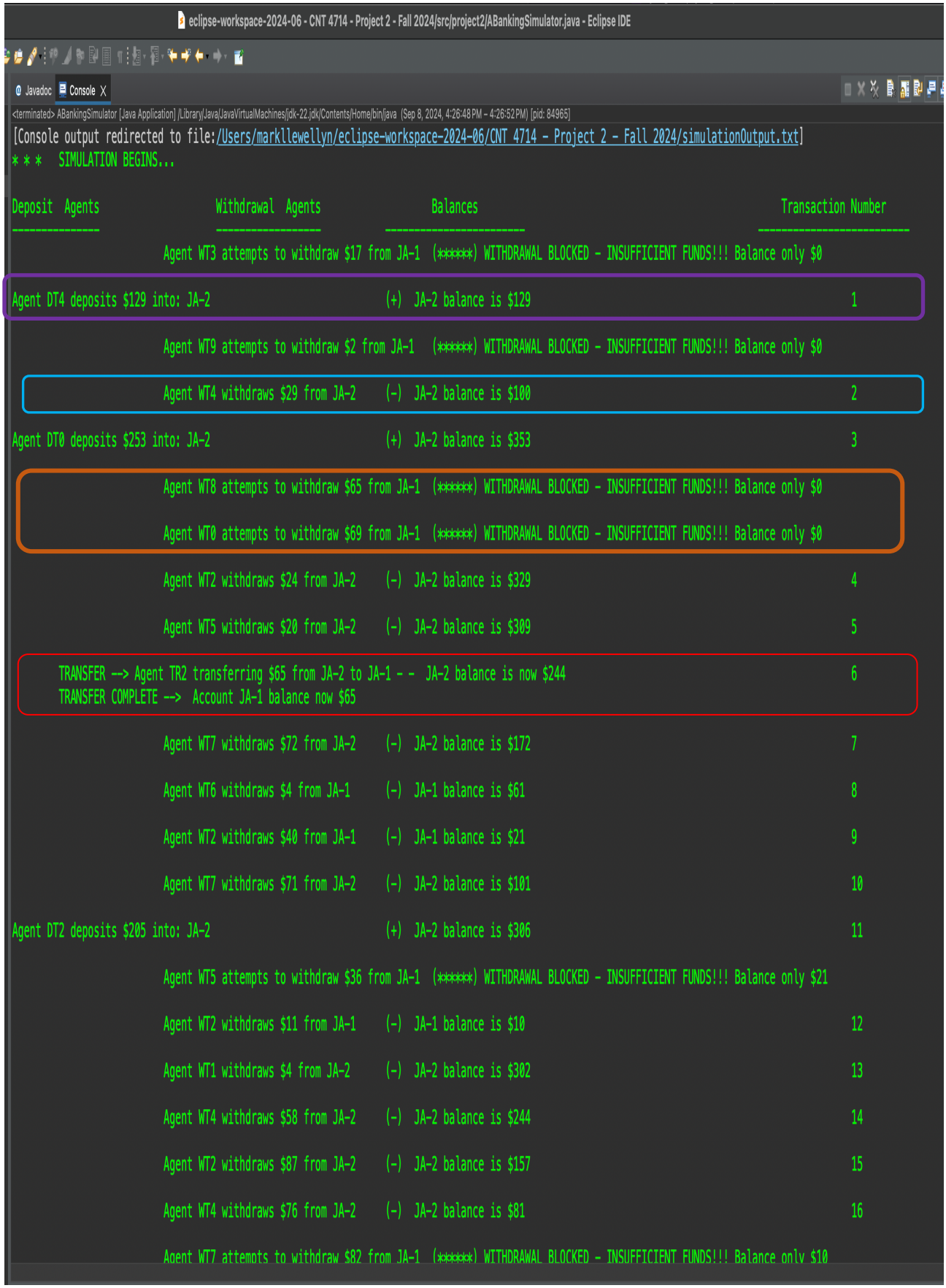
(2) Include a sufficient number of screen shots that illustrate the execution of your synchronized threaded application (the simulation output). See below for some representative examples. Your screenshots should illustrate all the different types of output we expect to see. Include as many screen shots as necessary. Label all screenshots clearly.

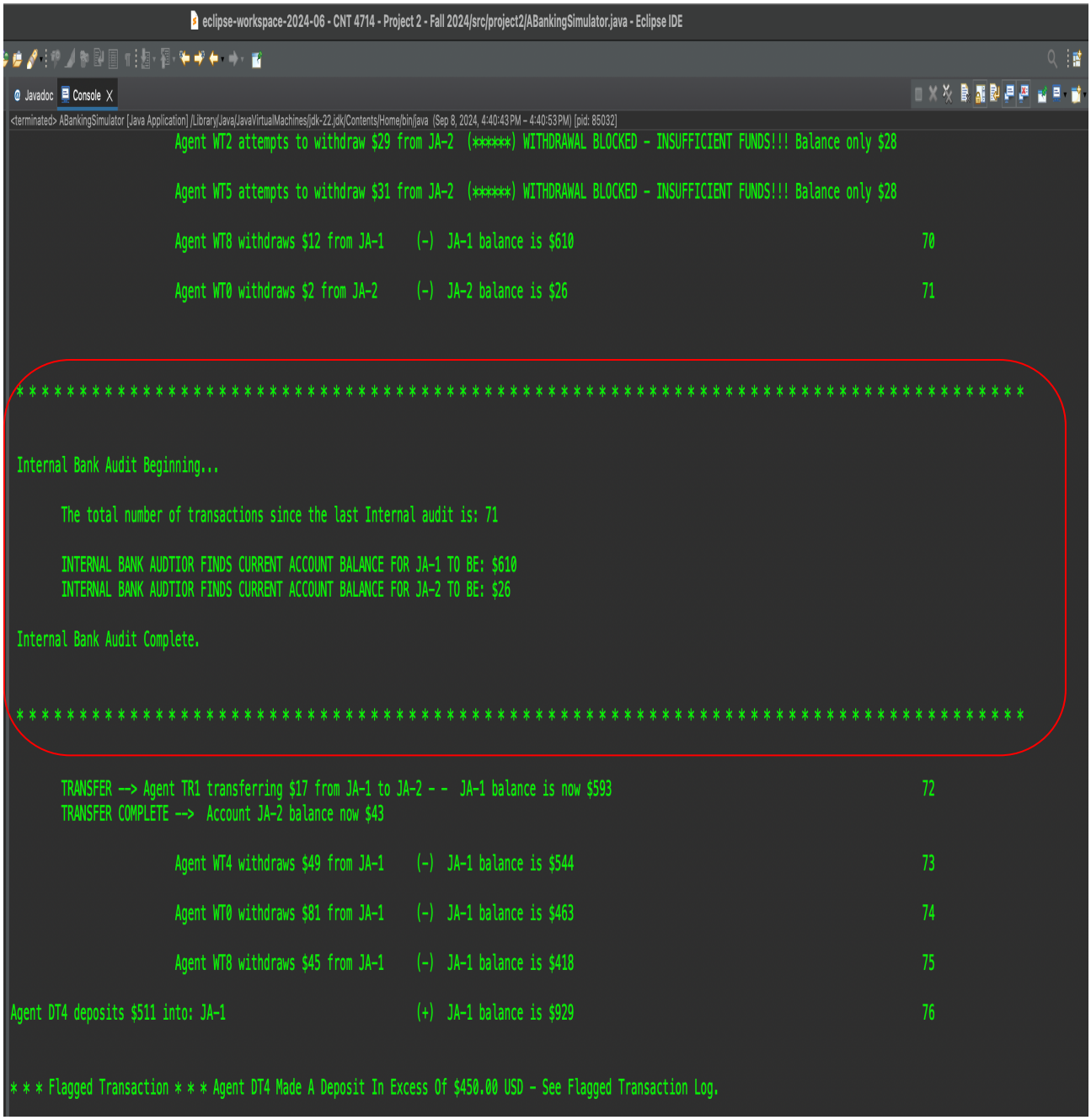
(3) Redirect the console output to an output file. Include a copy of this output file from a run of your banking simulation. This file output should be of sufficient length to include **at least** 750 transactions.

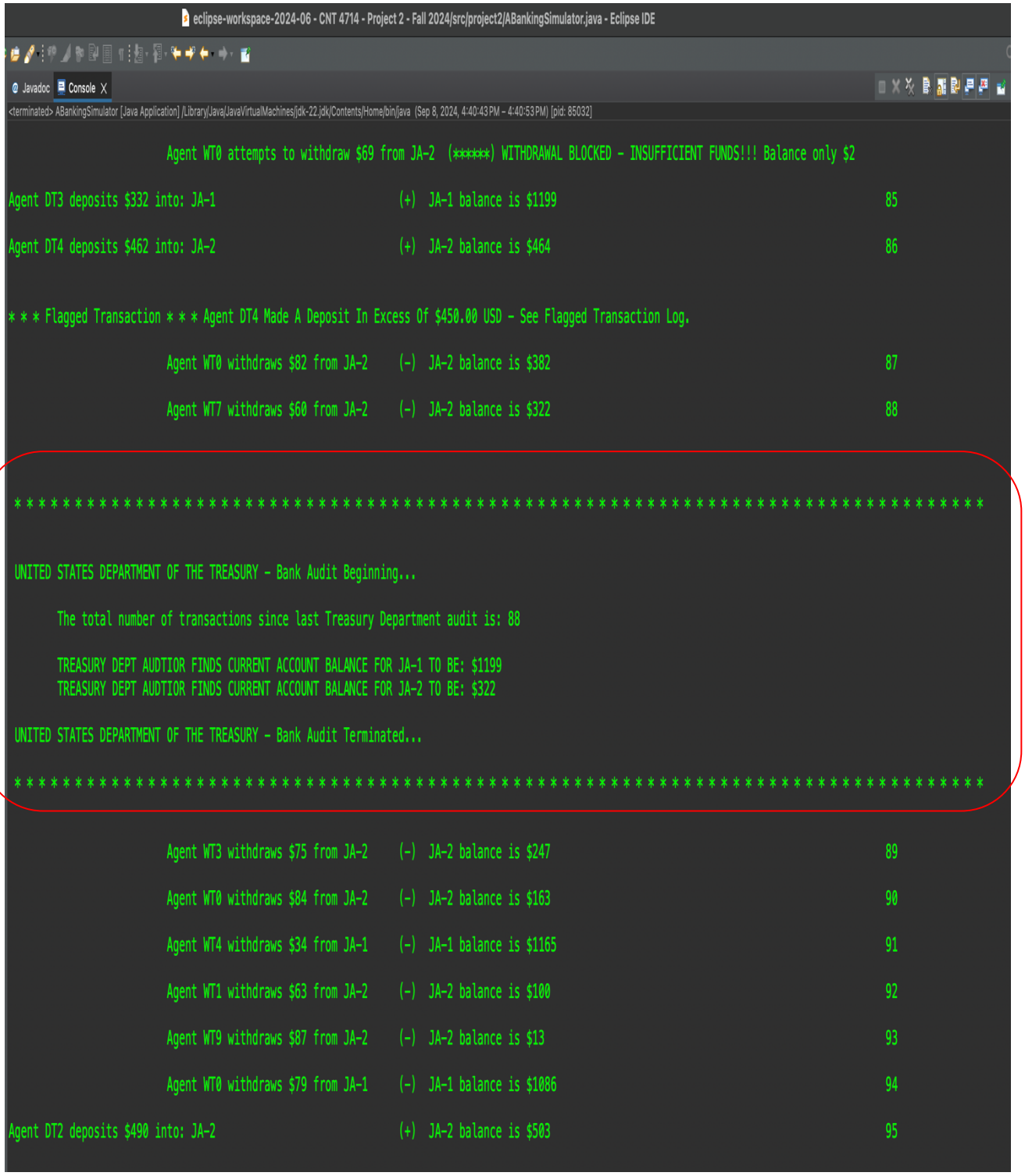
(4) Include a copy of your transaction.csv log file that matches the simulation output from (3) above. (see below for explanation).

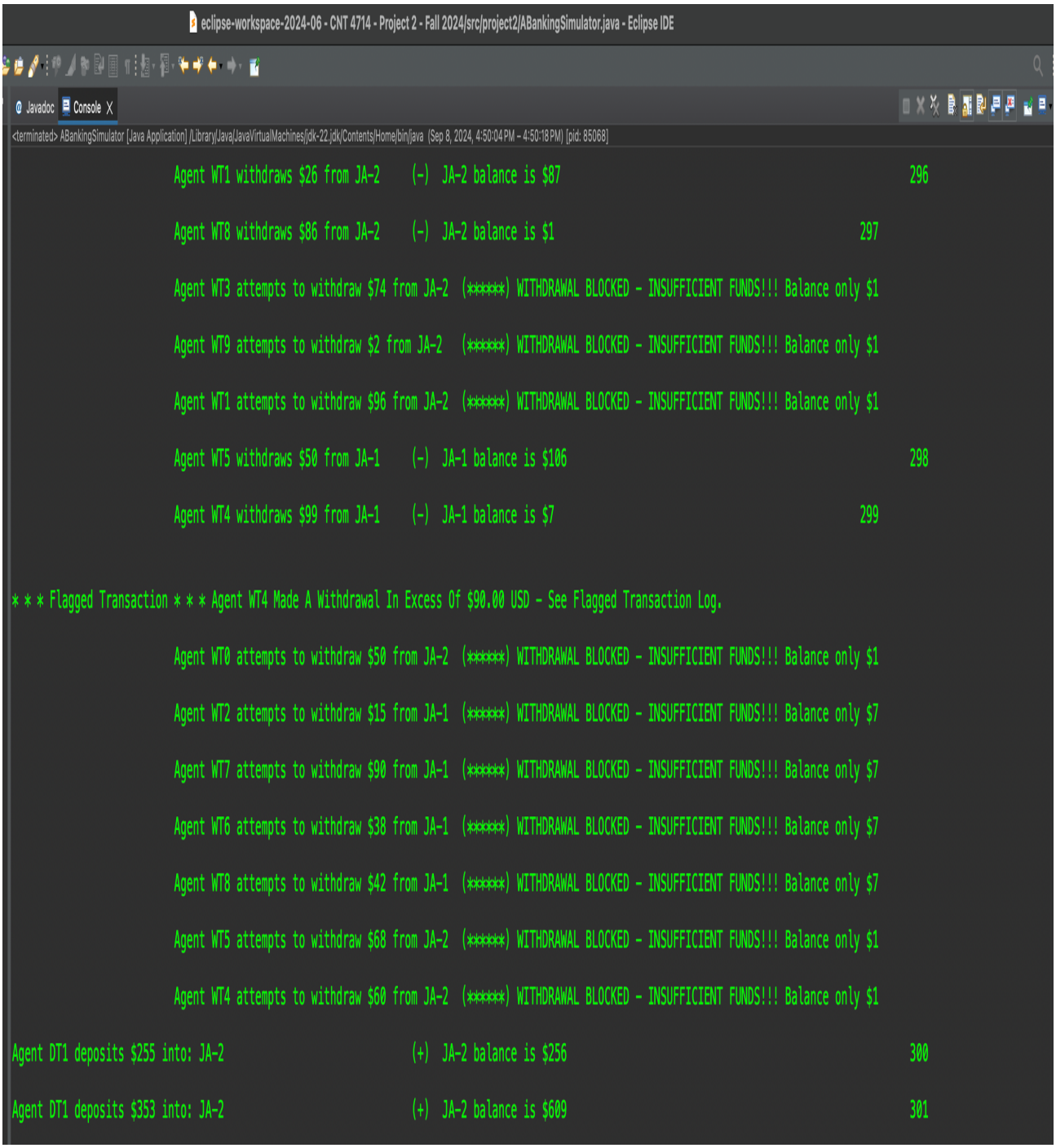
Additional Information:

Shown below are some examples 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.

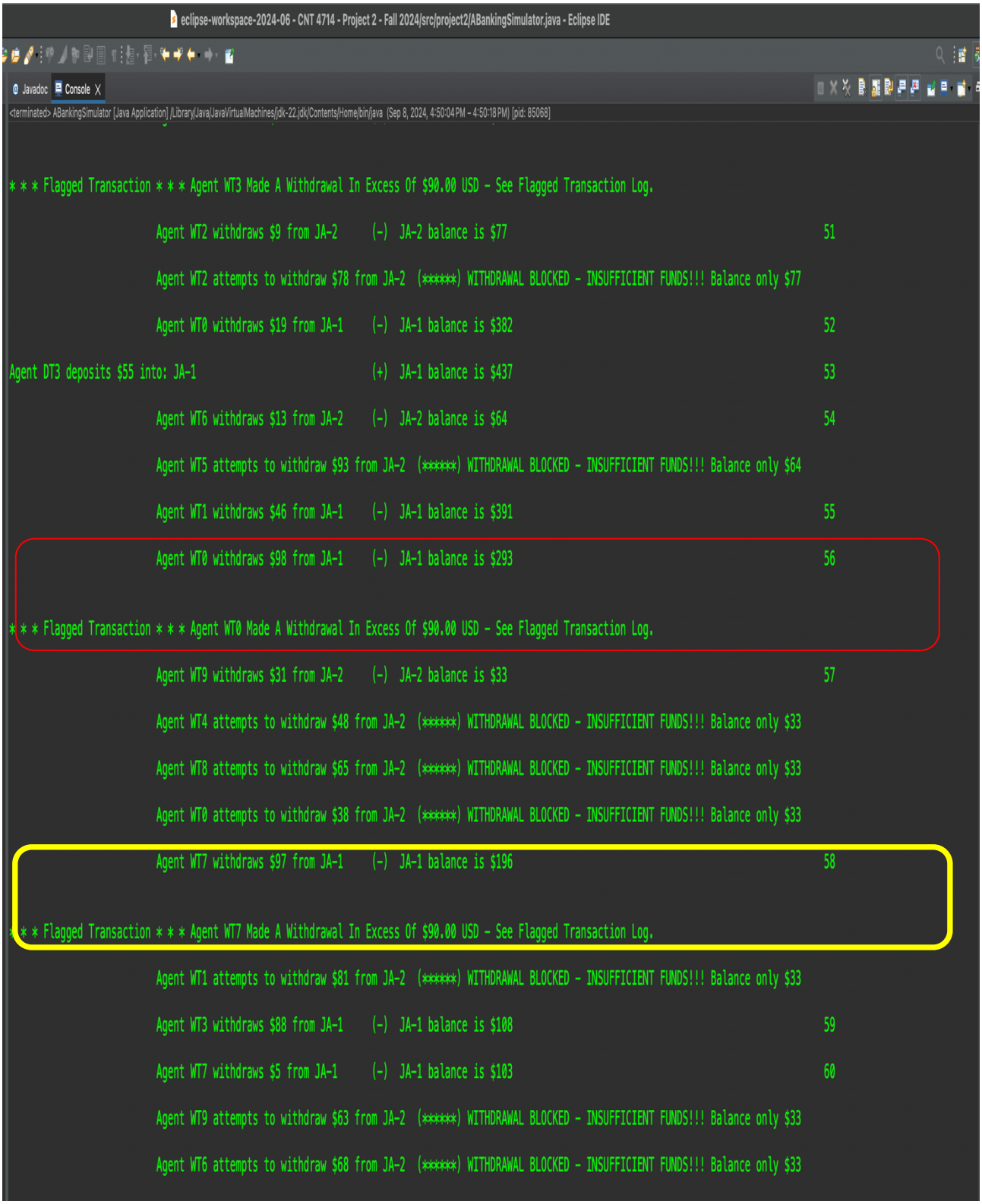

eclipse-workspace-2024-06 - CNT 4714 - Project 2 - Fall 2024/src/project2/ABankingSimulator.java - Eclipse IDE
Javadoc
Console
<terminated> ABankingSimulator [Java Application] /Library/Java/JavaVirtualMachines/jdk-22.jdk/Contents/Home/bin/java (Sep 8, 2024, 4:26:48 PM – 4:26:52 PM) [pid: 84965]
[Console output redirected to file:/Users/markllewellyn/eclipse-workspace-2024-06/CNT 4714 - Project 2 - Fall 2024/simulationOutput.txt]
* * * SIMULATION BEGINS...
Deposit Agents
Withdrawal Agents
Balances
Transaction Number
Agent WT3 attempts to withdraw $17 from JA-1 (*****) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $0
Agent DT4 deposits $129 into: JA-2 (+) JA-2 balance is $129 1
Agent WT9 attempts to withdraw $2 from JA-1 (*****) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $0
Agent WT4 withdraws $29 from JA-2 (-) JA-2 balance is $100 2
Agent DT0 deposits $253 into: JA-2 (+) JA-2 balance is $353 3
Agent WT8 attempts to withdraw $65 from JA-1 (*****) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $0
Agent WT0 attempts to withdraw $69 from JA-1 (*****) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $0
Agent WT2 withdraws $24 from JA-2 (-) JA-2 balance is $329 4
Agent WT5 withdraws $20 from JA-2 (-) JA-2 balance is $309 5
TRANSFER --> Agent TR2 transferring $65 from JA-2 to JA-1 - - JA-2 balance is now $244 6
TRANSFER COMPLETE --> Account JA-1 balance now $65
Agent WT7 withdraws $72 from JA-2 (-) JA-2 balance is $172 7
Agent WT6 withdraws $4 from JA-1 (-) JA-1 balance is $61 8
Agent WT2 withdraws $40 from JA-1 (-) JA-1 balance is $21 9
Agent WT7 withdraws $71 from JA-2 (-) JA-2 balance is $101 10
Agent DT2 deposits $205 into: JA-2 (+) JA-2 balance is $306 11
Agent WT5 attempts to withdraw $36 from JA-1 (*****) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $21
Agent WT2 withdraws $11 from JA-1 (-) JA-1 balance is $10 12
Agent WT1 withdraws $4 from JA-2 (-) JA-2 balance is $302 13
Agent WT4 withdraws $58 from JA-2 (-) JA-2 balance is $244 14
Agent WT2 withdraws $87 from JA-2 (-) JA-2 balance is $157 15
Agent WT4 withdraws $76 from JA-2 (-) JA-2 balance is $81 16
Agent WT7 attempts to withdraw $82 from JA-1 (*****) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $10


eclipse-workspace-2024-06 - CNT 4714 - Project 2 - Fall 2024/src/project2/ABankingSimulator.java - Eclipse IDE
Javadoc
Console X
<terminated> ABankingSimulator [Java Application] /Library/Java/JavaVirtualMachines/jdk-22.jdk/Contents/Home/bin/java (Sep 8, 2024, 4:40:43 PM – 4:40:53 PM) [pid: 85032]
Agent WT2 attempts to withdraw $29 from JA-2 (******) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $28
Agent WT5 attempts to withdraw $31 from JA-2 (******) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $28
Agent WT8 withdraws $12 from JA-1 (-) JA-1 balance is $610 70
Agent WT0 withdraws $2 from JA-2 (-) JA-2 balance is $26 71
****
Internal Bank Audit Beginning...
The total number of transactions since the last Internal audit is: 71
INTERNAL BANK AUDTIOR FINDS CURRENT ACCOUNT BALANCE FOR JA-1 TO BE: $610
INTERNAL BANK AUDTIOR FINDS CURRENT ACCOUNT BALANCE FOR JA-2 TO BE: $26
Internal Bank Audit Complete.
****
TRANSFER --> Agent TR1 transferring $17 from JA-1 to JA-2 - - JA-1 balance is now $593 72
TRANSFER COMPLETE --> Account JA-2 balance now $43
Agent WT4 withdraws $49 from JA-1 (-) JA-1 balance is $544 73
Agent WT0 withdraws $81 from JA-1 (-) JA-1 balance is $463 74
Agent WT8 withdraws $45 from JA-1 (-) JA-1 balance is $418 75
Agent DT4 deposits $511 into: JA-1 (+) JA-1 balance is $929 76
* * * Flagged Transaction * * * Agent DT4 Made A Deposit In Excess Of $450.00 USD - See Flagged Transaction Log.

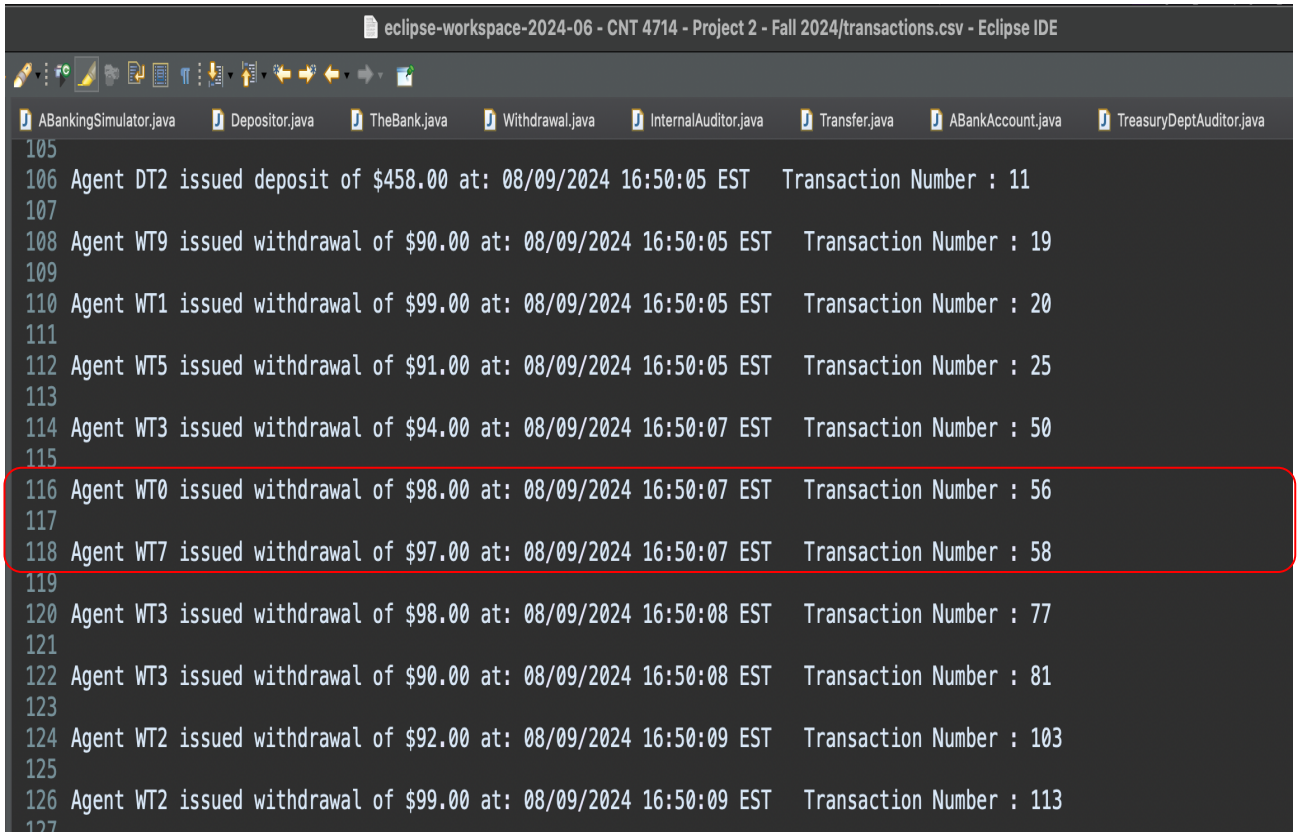

eclipse-workspace-2024-06 - CNT 4714 - Project 2 - Fall 2024/src/project2/ABankingSimulator.java - Eclipse IDE
Javadoc
Console
<terminated> ABankingSimulator [Java Application] /Library/Java/JavaVirtualMachines/jdk-22.jdk/Contents/Home/bin/java (Sep 8, 2024, 4:40:43 PM – 4:40:53 PM) [pid: 85032]
Agent WT0 attempts to withdraw $69 from JA-2 (******) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $2
Agent DT3 deposits $332 into: JA-1 (+) JA-1 balance is $1199 85
Agent DT4 deposits $462 into: JA-2 (+) JA-2 balance is $464 86
* * * Flagged Transaction * * * Agent DT4 Made A Deposit In Excess Of $450.00 USD - See Flagged Transaction Log.
Agent WT0 withdraws $82 from JA-2 (-) JA-2 balance is $382 87
Agent WT7 withdraws $60 from JA-2 (-) JA-2 balance is $322 88
UNITED STATES DEPARTMENT OF THE TREASURY - Bank Audit Beginning...
The total number of transactions since last Treasury Department audit is: 88
TREASURY DEPT AUDTIOR FINDS CURRENT ACCOUNT BALANCE FOR JA-1 TO BE: $1199
TREASURY DEPT AUDTIOR FINDS CURRENT ACCOUNT BALANCE FOR JA-2 TO BE: $322
UNITED STATES DEPARTMENT OF THE TREASURY - Bank Audit Terminated...
Agent WT3 withdraws $75 from JA-2 (-) JA-2 balance is $247 89
Agent WT0 withdraws $84 from JA-2 (-) JA-2 balance is $163 90
Agent WT4 withdraws $34 from JA-1 (-) JA-1 balance is $1165 91
Agent WT1 withdraws $63 from JA-2 (-) JA-2 balance is $100 92
Agent WT9 withdraws $87 from JA-2 (-) JA-2 balance is $13 93
Agent WT0 withdraws $79 from JA-1 (-) JA-1 balance is $1086 94
Agent DT2 deposits $490 into: JA-2 (+) JA-2 balance is $503 95



At the point in time the screenshot above was taken, all withdrawal agents (WT0 – WT9) are blocked.



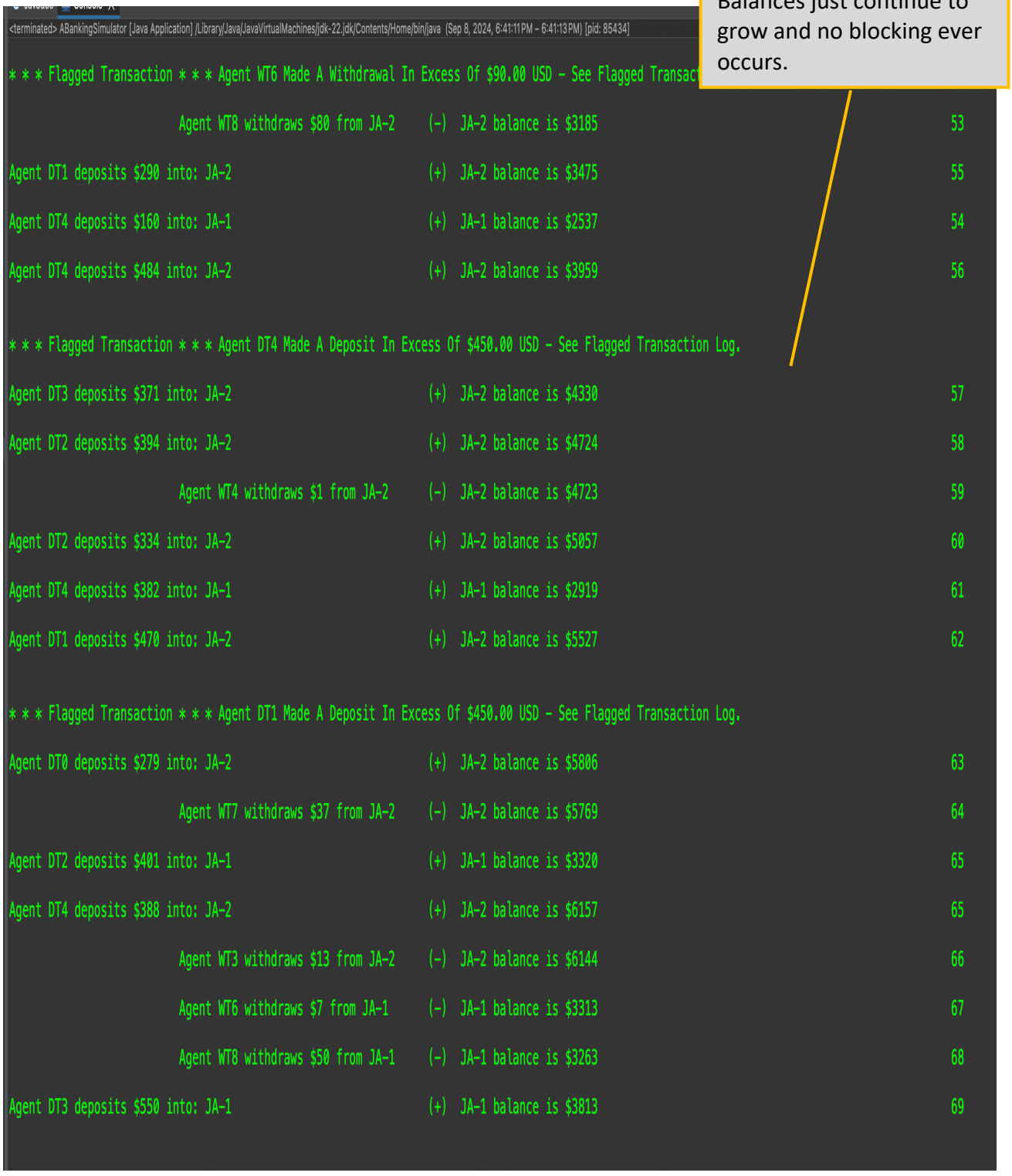
Note the flagged transactions. See next page for screenshot showing a portion of the transactions.csv file that illustrates the two flagged transactions shown above (transaction number 56 and 58).


eclipse-workspace-2024-06 - CNT 4714 - Project 2 - Fall 2024/transactions.csv - Eclipse IDE
ABankingSimulator.java
Depositor.java
TheBank.java
Withdrawal.java
InternalAuditor.java
Transfer.java
ABankAccount.java
TreasuryDeptAuditor.java
105
106 Agent DT2 issued deposit of $458.00 at: 08/09/2024 16:50:05 EST Transaction Number : 11
107
108 Agent WT9 issued withdrawal of $90.00 at: 08/09/2024 16:50:05 EST Transaction Number : 19
109
110 Agent WT1 issued withdrawal of $99.00 at: 08/09/2024 16:50:05 EST Transaction Number : 20
111
112 Agent WT5 issued withdrawal of $91.00 at: 08/09/2024 16:50:05 EST Transaction Number : 25
113
114 Agent WT3 issued withdrawal of $94.00 at: 08/09/2024 16:50:07 EST Transaction Number : 50
115
116 Agent WT0 issued withdrawal of $98.00 at: 08/09/2024 16:50:07 EST Transaction Number : 56
117
118 Agent WT7 issued withdrawal of $97.00 at: 08/09/2024 16:50:07 EST Transaction Number : 58
119
120 Agent WT3 issued withdrawal of $98.00 at: 08/09/2024 16:50:08 EST Transaction Number : 77
121
122 Agent WT3 issued withdrawal of $90.00 at: 08/09/2024 16:50:08 EST Transaction Number : 81
123
124 Agent WT2 issued withdrawal of $92.00 at: 08/09/2024 16:50:09 EST Transaction Number : 103
125
126 Agent WT2 issued withdrawal of $99.00 at: 08/09/2024 16:50:09 EST Transaction Number : 113

Some additional things to watch for:

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.

Balances just continue to grow and no blocking ever occurs.



The Internal Bank auditor agent and the Treasury Dept auditor agents are not related and execute on different random intervals. They will typically see different account balances and a differing number of transaction executions between runs. This is shown below.

```
*********************************************************************************************

UNITED STATES DEPARTMENT OF THE TREASURY - Bank Audit Beginning...

      The total number of transactions since last Treasury Department audit is: 114

      TREASURY DEPT AUDTIOR FINDS CURRENT ACCOUNT BALANCE FOR JA-1 TO BE: $113
      TREASURY DEPT AUDTIOR FINDS CURRENT ACCOUNT BALANCE FOR JA-2 TO BE: $44

UNITED STATES DEPARTMENT OF THE TREASURY - Bank Audit Terminated...

*********************************************************************************************

                    Agent WT8 withdraws $6 from JA-2     (-)  JA-2 balance is $38                                    691

                    Agent WT7 attempts to withdraw $90 from JA-2  (******) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $38

                    Agent WT9 withdraws $24 from JA-1    (-)  JA-1 balance is $89                                    692

                    Agent WT5 attempts to withdraw $73 from JA-2  (******) WITHDRAWAL BLOCKED - INSUFFICIENT FUNDS!!! Balance only $38


*********************************************************************************************

Internal Bank Audit Beginning...

      The total number of transactions since the last Internal audit is: 192

      INTERNAL BANK AUDTIOR FINDS CURRENT ACCOUNT BALANCE FOR JA-1 TO BE: $89
      INTERNAL BANK AUDTIOR FINDS CURRENT ACCOUNT BALANCE FOR JA-2 TO BE: $38

Internal Bank Audit Complete.

*********************************************************************************************
```

Auditor agents (both types) hold locks on all accounts and thus block any/all other agents from executing while an audit is underway. You should never see any other agent running interleaved with an auditor agent.