DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING CONCORDIA UNIVERSITY

COMP 5461: Operating Systems

Winter 2019

Programming Assignment 1

Submitted to:

DHRUBAJYOTI GOSWAMI

Submitted by:

MD Tanveer Alamgir

ID: 40014877

Tasks

Task 1: Atomicity violation

Compile and run the Java app given to you as it is. Explain why the main requirement above (i.e. consistent state of the account array) is not met. Find the bug(s) that cause(s) it and in a few sentences explain how it can be fixed.

ANS:

The account array was not consistent because the critical section in both “deposit” and “withdraw” methods where the “Balance” variable is shared was not synchronized.

balance = balance + amount;

&

balance = balance - amount;

So both threads for same account is trying to access the “balance” variable at the same time that causes the wrong output of final balance.

Bug:

```
public void debosit(double amount){


        // Waste some time doing fake computations

        // do not remove or modify any of the following 3 statements

        double k = 999999999;

        for(int i=0;i<100;i++)

            k = k / 2;

       balance = balance + amount; //Not synchronized

        // Waste some time doing fake computations
```

```
                // do not remove or modify any of the following 3 statements
                k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;
}
```

And

```
public void withdraw(double amount){

                // Waste some time doing fake computations
                // do not remove or modify any of the following 3 statements
                double k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;
                    balance = balance - amount; //Not synchronized
                // Waste some time doing fake computations
                // do not remove or modify any of the following 3 statements
                k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;
```

```
}
```

How to fix:

It can be fixed 2 ways. One is to synchronize the both deposit and withdraw methods or synchronize the critical section of both methods which is updating “balance” variable. It’s recommended to only synchronize the critical section not the whole method.

Task 2: Starting Order

Explain in a few sentences what determines the starting order of the threads. Can the consistency of the accounts preserved by changing the starting order? Explain.

ANS:

Here the threads start with .start() call. But the thread order can’t be determined without synchronizing the critical section of the code.

No, the consistency of the accounts can’t be preserved by changing the starting order as a newly started thread joins the main thread right after its start and only the consistency be achieved by synchronization.

Task 3: Critical section

Identify the critical sections of the given code. You can “cut and paste” the relevant pieces of code to your answer.

ANS:

```
public void debosit(double amount){


        // Waste some time doing fake computations
```

```
                // do not remove or modify any of the following 3 statements
                double k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;
                    balance = balance + amount; //Critical section
                // Waste some time doing fake computations
                // do not remove or modify any of the following 3 statements
                k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;
}
```

And

```
public void withdraw(double amount){

                // Waste some time doing fake computations
                // do not remove or modify any of the following 3 statements
                double k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;
                    balance = balance - amount; //Critical section
```

```
        // Waste some time doing fake computations
        // do not remove or modify any of the following 3 statements
        k = 999999999;
        for(int i=0;i<100;i++)
            k = k / 2;
    }
```

Task 6: Synchronized block versus synchronized method

Considering the results of Task 4 and Task 5, what is(are) the advantage(s) of synchronized block over synchronized method, or vice versa? Explain.

ANS:

Synchronized block is always better than synchronized method. A method can have some codes that do not need to be synchronized and they are not shared. So it will affect the efficiency of the application.

Like in the account.java we have the below code in both deposit and withdraw:

```
double k = 999999999;
for(int i=0;i<100;i++)
k = k / 2;
```

Since this block of code is not shared by threads so it should not be synchronized as it will slower down the application.