Lab Assignment - 6

Copyright Policy

Academic Integrity

What is allowed:

- Looking up **syntax** related to Java
- You can refer the code and classwork created in this course

What is **NOT** allowed:

- Searching for partial or full solutions of the main problem description
- Communication with others, either inside or outside the class
- Sharing of resources, including but not limited to code, files, links, computers, etc.

Instructions:

- This assignment is to be done individually.
- Besides implementing the required functionality submissions are required to use the correct coding conventions used in class, professional organization of the code, alignment, clarity of names is all going to be part of the evaluation.

Submission Checklist:

- Once you are done, write your name and student number at the top of each **.java** file.
- Compress your entire project folder into a .zip file. Your .zip file name must be **YourFirstName_Section#_Lab6.zip** such as John_17_Lab6. **Please don't submit .rar or .7zip file.**
- Take a **screenshot** of the output and upload in the submission folder.
- Your submission folder should have two individual files: 1) the zip file, 2) a screenshot of output

Task:

In IntelliJ Idea, create a new project (Java Application) named **YourFirstName_Section#_Lab6** (where YourFirstName is your first name, such as John_17_Lab6).

The application will demonstrate multithreading concepts to perform transactions on account. Perform the following operations:

TransactionType enumeration:

Create an enumeration named as TransactionType having values DEPOSIT and WITHDRAW.

Account class:

- The account class contains owner name and balance properties. All the accounts by default get 1000$ as balance.

- The constructor of the Account class will allow to initialize only the owner's name.
- Create a toString function to display account details.
- Create a synchronized function named as performTransaction(). This function will accept the transaction type and amount for the transaction as parameter. The function is supposed to withdraw or deposit the amount based on the transaction type.

Transaction class:

This is a thread (implements Runnable) that will use a shared account as a resource to perform transactions.

- The class will have three properties; account, transaction type and amount. Initialize all the three properties through constructor.
- In the execution of this thread (run() method), call the performTransaction() method of the account class and then display the account details.

TransactionTest class:

This class will implement the main() method to test the operations of multithreading. Use the following code to create instances of Account and Transaction classes.

```
Account sharedAccount = new Account("John");

Transaction t1 = new Transaction(sharedAccount, TransactionType.WITHDRAW, 200);
Transaction t2 = new Transaction(sharedAccount, TransactionType.DEPOSIT, 1000);
Transaction t3 = new Transaction(sharedAccount, TransactionType.DEPOSIT, 500);
Transaction t4 = new Transaction(sharedAccount, TransactionType.WITHDRAW, 100);
```

Execute all the four threads (t1, t2, t3 & t4) with the help of ExecutorService and show appropriate result. At the end of execution, show the sharedAccount details.

Sample output:

Note: Your output could be different depending on the sequence of thread execution.

```
Performing transactions
--------------------------------------------------------------------------------
Account{ownerName='John', balance=1000.0}
Trying to withdraw $200.0 from John's account
successfully withdrawn 200.0$ from John's account
--------------------------------------------------------------------------------
Account{ownerName='John', balance=800.0}
Trying to withdraw $100.0 from John's account
successfully withdrawn 100.0$ from John's account
--------------------------------------------------------------------------------
Account{ownerName='John', balance=700.0}
Trying to deposit $500.0 to John's account
successfully deposited 500.0$ to John's account
--------------------------------------------------------------------------------
Account{ownerName='John', balance=1200.0}
Trying to deposit $1000.0 to John's account
successfully deposited 1000.0$ to John's account
--------------------------------------------------------------------------------
Account{ownerName='John', balance=2200.0}
SharedAccount  : Account{ownerName='John', balance=2200.0}
```