Laboratory 5

Title of the Laboratory Exercise: ATM application

1. Introduction and Purpose of Experiment

Aim and Objectives

Aim

- To develop programs to maintain state consistency

2. Experimental Procedure

i. Analyse the problem statement

ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code

iii. Implement the algorithm in Java language

iv. Compile the Java program

v. Test the implemented program

vi. Document the Results

vii. Analyse and discuss the outcomes of your experiment

3. Question

Create a multithreaded Java program (min 3 threads) with the following operations

- Balance Enquiry
- Deposit(int x)
- Withdrawal(int y)

4. Computations/Algorithms

A. Withdraw:

Take the amount user desires to withdraw as input.

If the balance amount greater than or equal to the withdrawal amount then Perform the transaction and give the user the desired amount.

Else print Insufficient Funds message.

B. Deposit:

Take the amount user desires to deposit as input.

Add the received input from the user to balance and update its value.

balance = balance + deposit.

Print a message on screen stating deposit transaction has been successful.

C. Check Balance:

Print a message on screen showing the value of balance amount.

D. Exit:

Exit the current Transaction mode and return the user to the home page or initial screen.

5. Source Code

ThreadATM.java

```
1  package lab5;
2   //@author Deepak R
3  public class ThreadATMDemo {
4          public static void main(String[] args) {
5                  // TODO Auto-generated method stub
6                  Account account = new Account();
7                   // Set the user name;
8                  account.setName("Deepak R");
9                   // Initialize the balance;
10                 account.setValue(0);
11
12                  //Save in 100;
13                 new SaveMoney(7500,account).start();
14
15                  //Save in 200;
16                 new SaveMoney(4400, account).start();
17
18                  // Take out 500;
19                 new FetchMoney(2400, account).start();
20                 new FetchMoney(4800, account).start();
21         }
22 }
```

Savemoney.java

```java
package lab5;

import java.io.Serializable;

/**
   * deposit thread class;
 * @author Deepak R
 *
 */
public class SaveMoney extends Thread implements Serializable{

	private static final long serialVersionUID = 3095120546560212724L;

	Private account account; // account object;
	 int money; //amount;

	public SaveMoney(){
		super();
	}

	public SaveMoney(int money,Account account){
		this.account = account;
		this.money = money;
	}

	@Override
	public void run() {
		// TODO Auto-generated method stub
		 int currMonery = account.search(); //Query account balance;

		 // Synchronize objects, only allow single-threaded operation;
		synchronized (account) {
			try {
				 sleep(5); //When the query is made, the time is paid;
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			 account.putMonery(this.money); // insert amount;
			 // Output deposit information;
			 System.out.println("Dear "+account.getName()+" User Hello!"+" The current balance is: "+currMonery +" Rupee. "+" "
					 + "Deposit balance: "+this.money+ "Yuan." + "Account balance is: "+account.search()+" ");
		}
	}
}
```

Fetchmoney.java

```java
package lab5;

import java.io.Serializable;
/**
   * withdrawal thread class;
 * @author Deepak R
 *
 */
public class FetchMoney extends Thread implements Serializable{

	private static final long serialVersionUID = -5059599151558445815L;

	Private account account; // account object;
	 int monery; // balance;

	public FetchMoney(){
		super();
	}

	public FetchMoney(int monery,Account account){
		this.account = account;
		this.monery = monery;
	}

	@Override
	public void run() {
		// TODO Auto-generated method stub
		 int currMonery = account.search(); //current balance;
		synchronized (account) {
			try {
				 sleep(5); //Time for withdrawal of payment;
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			 // Take out the balance;
			int getMonery = account.getMonery(monery);
			System.out.println(+account.getName()+" User Hello!"+" The current balance is: "+currMonery +" yuan. "+""
					+ " Take the balance as: "+getMonery+". The "+" account balance is: "+account.search()+"yuan");
		}
	}
}
```

-

Account.java

```java
package lab5;
//@author Deepak R
public class Account {
        String name; //username;
        int value; // account balance;
        int Monery;
        public void putMonery(int monery){
                this.value = this.value + monery;
        }
        public int getMonery(int monery){

                // Determine whether the account balance is greater than the money to be taken;
                if(this.value > monery ){
                        this.value = this.value - monery;
                }else{
                        Monery = this.value; //If the account balance is not enough, then take out the amount of all account balances.
                        this.value = 0;
                }
                //Return the money that was taken out;
                return monery;
        }
        /**
                * Check balances;
         * @return returns the account balance;
         */
        public int search(){
                return this.value;
        }

        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public int getValue() {
                return value;
        }
        public void setValue(int value) {
                this.value = value;
        }
}
```

Output

```
Output - Lab5 (run)
run:
Deepak R The Current Balance is : 00 Rupee
deposit balance is:7500 Rupee The Account balance is :7500 Rupee

Deepak R The Current Balance is : 7500 Rupee
Withdrawn balance is: 4400 Rupee The Account balance is: 3100 Rupee

Deepak R The Current Balance is : 3100 Rupee
deposit balance is:2400 Rupee The Account balance is :5500 Rupee

Deepak R The Current Balance is : 5500 Rupee
Withdrawn balance is: 4800 Rupee The Account balance is: 700 Rupee


BUILD SUCCESSFUL (total time: 0 seconds)
```

6. Analysis and Discussions

1. Limitations of Experiments

Program was Complex

2. Limitations of Results

Tested only for small data. Result may vary if tried for large data.

3. Learning happened

Learnt how to implement ATM functions using threads

4. Recommendations

None