

```

1 package com;
2
3 import java.util.*;
4
5 public class Main {
6
7     public static void main(String[] args) {
8
9         new Main();
10
11     } //main loop
12
13     Main() {
14         Scanner keyVal = new Scanner(System.in);
15         BankAccount commBA = null;
16         ArrayList<double[]> userTL = new ArrayList<>();
17
18
19         double[] transaction_1 = {50, 10, -20, 10, -20, 20, 10, 50, -10, 10, -10, 50};
20         userTL.add(transaction_1);
21
22         double[] transaction_2 = {20, 20, -20, 50, -20, 10, 50, 50, -20, 10, 10};
23         userTL.add(transaction_2);
24
25         double[] transaction_3 = {50, 10, 10, -10, -10, 50, 20, -10, -20};
26         userTL.add(transaction_3);
27
28         double[] transaction_4 = {50, 10, -20, 20, 10, -20};
29         userTL.add(transaction_4);
30
31         ArrayList<User> userAL = new ArrayList<>();
32
33         while (true) {
34             System.out.println("\n1. Create Bank Account\n2. Create User\n3. Run SyncSim\n4. Run UnSyncSim\n5.
Reset bank account to 1980\n6. Exit");
35             switch (keyVal.nextInt()) {
36                 case 1:
37                     //create new bank account
38                     commBA = new BankAccount(9876543210L, 1980);
39                     System.out.println("Created new bank account, Acc No: " + commBA.getAccountNo() + " with
initial balance of: " + commBA.getAccountBalance());
40                     break;
41
42                 case 2:
43
44                     try {
45                         userAL.add(new User("Saul", "Goodman", commBA, userTL.get(0)));
46                         userAL.add(new User("Walter", "White", commBA, userTL.get(1)));
47                         userAL.add(new User("Jessie", "Pinkman", commBA, userTL.get(2)));
48                         userAL.add(new User("Hank", "Schrader", commBA, userTL.get(3)));
49
50                         for (int i = 0; i < userAL.size(); i++) {
51                             System.out.println("User: " + userAL.get(i).getUName() +
52                                 " Bank Account: " + userAL.get(i).getBA().getAccountNo() +
53                                 " Available Money: " + userAL.get(i).getBA().getAccountBalance());
54                         }
55
56                     } catch (NullPointerException e) {
57                         System.out.println("No Bank account Created!!");
58                     }
59                     break;
60
61                 case 3:
62                     try {
63                         for (int i = 0; i < userTL.size(); i++) {
64                             SyncSim syncUser = new SyncSim(userAL.get(i));
65                             syncUser.start();
66                             //uncommenting this line may jeopardize the result.
67                             //syncUser.join();
68                         }
69                     } catch (Exception e) {
70                         System.out.println("No bank account or user accounts created!!");
71                         e.printStackTrace();
72                     }
73                     break;
74
75                 case 4:
76                     try {
77                         for (int i = 0; i < userTL.size(); i++) {
78                             UnSyncSim unsyncUser = new UnSyncSim(userAL.get(i));
79                             unsyncUser.start();
80                             //uncomment this line prints menu after threads finish

```

```

81         //but doesn't show the problem of wrong acc value.
82         //unsyncUser.join();
83     }
84     } catch (Exception e) {
85         System.out.println("No bank account or user accounts created!");
86         e.printStackTrace();
87     }
88     break;
89
90     case 5:
91         try {
92             commBA.setAccountBalance(1980);
93         } catch (NullPointerException e) {
94             System.out.println("No Bank account Created!!");
95         }
96
97         break;
98     case 6:
99         System.exit(0);
100        break;
101    default:
102        System.out.println("Wrong input value!");
103        break;
104    }
105 }
106 }
107 }

```

```
1 package com;
2
3 public class User
4 {
5     private String name;
6     private String surname;
7     private BankAccount bankAccount;
8     private double[] transactionList;
9
10    public User(String name, String surname, BankAccount bA, double[] tL)
11    {
12        this.name = name;
13        this.surname = surname;
14        this.bankAccount = bA;
15        this.transactionList = tL;
16    }
17
18    public String getUName(){
19        return this.name;
20    }
21
22    public BankAccount getBA(){
23        return this.bankAccount;
24    }
25
26    //get user TranList
27    public double[] getTransactionList(){
28        return this.transactionList;
29    }
30
31    //set user TranList
32    public void setTransactionList(double[] list){
33        this.transactionList = list;
34    }
35
36    public double getTransaction(int index){
37        return this.transactionList[index];
38    }
39
40    public int lengthTransactionList(){
41        return this.transactionList.length;
42    }
43 }
```

```
1 package com;
2
3 public class SyncSim extends Thread{
4
5     private User user;
6     private double[] transactions;
7     private double value;
8     private BankAccount ba;
9
10    public SyncSim(User anyUser)
11    {
12        super("name");
13
14        this.user = anyUser;
15        this.transactions = anyUser.getTransactionList();
16        this.ba = user.getBA();
17    }
18
19    public void run(){
20        try{
21            for(int i = 0; i < transactions.length; i++) {
22                value = transactions[i];
23                if (value > 0) {
24                    ba.sync_deposit(value, user);
25                }
26                if (value < 0) {
27                    ba.sync_withdraw(value, user);
28                }
29                if (value == 0) {
30                    System.out.println("No Transaction!\n");
31                }
32                sleep(5);
33            }
34        }
35        catch(Exception e) {
36            System.out.println("No account has been created!\n");
37        }
38    }
39
40 }
41
```

```
1 package com;
2
3 public class UnSyncSim extends Thread{
4
5     private User user;
6     private double[] transactions;
7     private double value;
8     private BankAccount ba;
9
10    public UnSyncSim(User anyUser)
11    {
12        super("name");
13
14        this.user = anyUser;
15        this.transactions = anyUser.getTransactionList();
16        this.ba = user.getBA();
17    }
18
19    public void run(){
20        try{
21            for(int i = 0; i < transactions.length; i++) {
22                value = transactions[i];
23                if (value > 0) {
24                    ba.deposit(value, user);
25                }
26                if (value < 0) {
27                    ba.withdraw(value, user);
28                }
29                if (value == 0) {
30                    System.out.println("No Transaction!\n");
31                }
32                sleep(5);
33            }
34        }
35        catch(Exception e) {
36            System.out.println("No account has been created!\n");
37        }
38    }
39
40 }
41
```

```

1 package com;
2
3 public class BankAccount
4 {
5     private long accountNo;
6     private double accountBalance;
7
8
9     public BankAccount(long accountNo, double accountBalance)
10    {
11        this.accountNo = accountNo;
12        this.accountBalance = accountBalance;
13    }
14
15    public long getAccountNo()
16    {
17        return this.accountNo;
18    }
19
20    public double getAccountBalance()
21    {
22        return this.accountBalance;
23    }
24
25    public void setAccountBalance(double newBalance)
26    {
27        this.accountBalance = newBalance;
28        System.out.println("Account balance value: " + this.accountBalance);
29    }
30
31    public synchronized void sync_deposit(double value, User u)
32    {
33        accountBalance += value;
34        notifyAll();
35        System.out.println("Transaction value: " + value + "\tBalance after transaction: " + accountBalance + "\t"
Client: " + u.getUserName());
36    }
37
38    public synchronized void sync_withdraw(double value, User u)
39    {
40        while(Math.abs(value) > accountBalance){
41            try{
42                wait();
43            }catch (InterruptedException e){
44                System.out.println("Thread interrupted");
45            }
46        }
47        accountBalance += value;
48        System.out.println("Transaction value: " + value + "\tBalance after transaction: " + accountBalance + "\t"
Client: " + u.getUserName());
49    }
50
51    }
52
53    public void deposit(double value, User u)
54    {
55        accountBalance += value;
56        System.out.println("Transaction value: " + value + "\tBalance after transaction: " + accountBalance + "\t"
Client: " + u.getUserName());
57    }
58
59    public void withdraw(double value, User u)
60    {
61        accountBalance += value;
62        System.out.println("Transaction value: " + value + "\tBalance after transaction: " + accountBalance + "\t"
Client: " + u.getUserName());
63    }
64    }
65
66 }

```