**Exercise 6**

1. Complete class WithdrawThread. Add variables and methods as needed

class WithdrawThread extends Thread {

private Account account;

private int totalWithdraw;

public WithdrawThread(String n, Account a) { super(n); account = a; }

public void run()

{

/\* Add a loop that

1. Random the amount of money requested by thread. You may set the

range of randomed values, e.g. 1-10

2. Call account.update(…) to withdraw the money & update balance

3. Stop when the account’s balance is 0

\*/

}

} // end WithdrawThread

**Hint** : use shared (or static) variable for loop condition, so that all threads

work & stop upon the same condition

2. Complete class Account. Modify it as needed. You can add more variables and methods,

but do not change the visibility of existing ones

class Account {

private int balance;

private PrintWriter out;

public Account(int b) {

balance = b;

try { out = new PrintWriter(new File("balance.txt")); }

catch(Exception e) { out = new PrintWriter(System.out); }

}

public int update(int request)

{

/\* Update the account

1. If request <= balance, the thread can get money as it requests

2. If request > balance, the thread can only get available money

3. Print thread’s name, money requested, money withdrawn, current balance

to file balance.txt

4. You may return some value to WithdrawThread

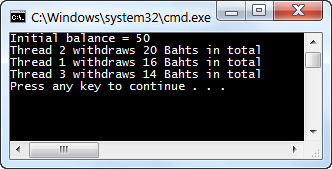
\*/

}

} // end Account

3. Write another class that acts as main class. In main method

* Get initial balance from user. Create an Account with this initial balance
* Create 3 WithdrawThreads and let them compete to withdraw money from this Account
* All threads sees the same Account object and write to the same output file
* After all threads finish their jobs, report the total money withdrawn by each thread. Sort the output in either increasing or decreasing order of total money



**balance.txt**

Thread 1 >> request = 2, get = 2, balance = 48, total withdraw = 2

Thread 3 >> request = 3, get = 3, balance = 45, total withdraw = 3

Thread 2 >> request = 7, get = 7, balance = 38, total withdraw = 7

Thread 1 >> request = 7, get = 7, balance = 31, total withdraw = 9

Thread 2 >> request = 4, get = 4, balance = 27, total withdraw = 11

Thread 3 >> request = 5, get = 5, balance = 22, total withdraw = 8

Thread 1 >> request = 4, get = 4, balance = 18, total withdraw = 13

**Thread 3** >> request = 6, get = 6, balance = 12, **total withdraw = 14**

Thread 2 >> request = 4, get = 4, balance = 8, total withdraw = 15

**Thread 1** >> request = 3, get = 3, balance = 5, **total withdraw = 16**

**Thread 2** >> request = 6, get = 5, **balance = 0**, **total withdraw = 20**

Balance

update

must be

correct