Programming JAVA

You are required to write a Java program to simulate the operation of a bank ATM (cashpoint) system for payment and deposit on an account. In order to make the system fairly

simple there is only one bank account and there are 5 cards that can be used to deposit

or withdraw money from the account. The use of each card is to be simulated using a

single Java thread calling deposit or withdraw methods on the bank account object.

Clearly, the bank account must be simulated as a Java monitor. The monitor is to be

shared by all threads. Each thread should execute the code on the next page in a threadsafe manner. The initial amount of money in the account should be taken as the first

command line argument to the thread. The bank account must always have a positive

 balance, and the account is not allowed to become overdrawn, this means threads

must block if they call withdraw and insufficient funds are present. Additionally the

code must be modified in order to keep track of the amount of money deposited and

withdrawn by each thread in a thread-safe variable localBalance.

Your main thread should create and start the execution of all threads, then it should wait

until all threads have completed prior to printing out the message “Complete” along with

the final balance in the account, this means the bank account must also support the

method balance() to give the current balance in the account.

Remember: in concurrent programs, a program can be defective and appear to function

correctly, you need to know that you have developed your program correctly according to

the theoretical principles of concurrency. // account is a shared monitor object.

for(int i = 0; i < 20; i++) {

if(Math.random() > 0.5) {

account.withdraw((int)(Math.random()\*10));

}else {

account.deposit((int)(Math.random()\*10));

}

sleep(2000);

}

System.out.println(“THREAD “+ getId() + “ “ + localBalance);

Additionally, the program could deadlock under the circumstances that insufficient

balance is available to withdraw money, and all threads attempt to withdraw money

due to the values returned by Math.random() in the if-condition test. THIS IS A

FEATURE OF THE ASSIGNMENT DUE TO THE FACT THAT ALL THREADS COULD

ACT TO withdraw.