R. P. Shaha University
- an institution of Kumudini Welfare Trust of Bengal (BD) Ltd.

Course Code: CSE 226

Course Title: Visual and Net-Based Programming Lab

Assignment 3

Instructor: Agnik Saha

Building a partial banking application

In Assignment 3, you will use threads to allow multiple account holders to process simultaneous withdrawals from a single checking account.
Implement the following classes:

1) CreditCard Class

a) Must have the following attribute:
- account balance: double, private, credit card balance
- initialized with 5,000.00

b) Must have a getBalance method, which
- takes in no parameters
- returns the amount within the balance attribute

c) Must have a withdraw method, which
- takes in a parameter of type double
- returns no value
- substracts the parameter value from the account balance attribute

2) CardHolder Class

a) Must inherit Runnable abstract class

b) Must have the following attribute
- card: of type CreditCard, private, customer's credit card

c) Must have an overloaded constructor, which
- takes in a parameter, of type CreditCard
- assigns parameter value to the card attribute

d) Must override the run method, which
- creates a for loop that iterates one (1) to multiple times (i.e. one to six times)
- during each iteration of the loop
• calls the makeWithdrawal method, passing a withdrawal amount (i.e. 500.00)
• after calling the makeWithdrawal method, checks the account balance
• if the account balance is less than zero (0), prints an appropriate error message to the screen

e) Must have a makeWithdrawal method, which
- uses the private method modifier (i.e. the method is a private method)
- synchronizes the method {i.e. Synchronized (Java) }

A. to synchronize the method in Java, include the keyword "Synchronized" in the makeWithdrawal method header
• create a lock object, i.e. "private static object padlock = new object();"

• place the body of the makeWithdrawal method inside the begin/end
brackets of a lock statement, i.e. “lock(padlock) {…}”
- takes in a parameter (withdrawal amount) of type double
- returns no value
- checks the account balance
A. if the account balance is less than the withdrawal amount, prints an error message that contains the thread name, withdraw amount, and account balance (i.e. “Not enough in: thread 1 to withdraw: $500.00, Balance: $200.00”)
B. if the account balance is greater than or equal to the withdrawal amount:
• prints a message that contains the thread name, withdrawal amount, and account balance (i.e. “thread 1, before withdrawing $500.00, Balance: $4000.00”)
• within a try/catch block, has the thread sleep for a “little bit” (i.e. thread.sleep(6000))
• after leaving the try/catch block, invokes the withdraw method (i.e. the class method within the CreditCard class), passing in the withdraw amount (i.e. 500.00)
• after returning from the withdraw method, prints a message that contains the thread name, withdrawal amount, and account balance (i.e. “thread 1, after withdrawing $500.00, Balance: $3500.00”)

3) Driver Class

a) using the CreditCard class, create a credit card object
b) using the CardHolder class, create a card holder object, passing the credit card object as a parameter
c) create a thread that takes the CardHolder object as a parameter
d) give this thread a name, i.e. Siam
e) create another thread that takes the CardHolder object as a parameter
f) give this thread a different name, i.e. Alif
g) start each individual thread