MOHIT
Page No.:
Date:

Assignment 8

) In this challenge, simulate a banking system.
. create the account and transaction classes.
1. The account class has a data member int balance, initially assigned to zero. The class should implement the following three methods:
String deposit(int money) to add money to the balance. This method should return a string that describes the deposit transaction, i.e. "depositing $money".

String withdraw(int money) to substract money from the balance. This method should return a string that describes the withdraw transaction, i.e "withdrawing $money". Note that, if there is insufficient balance to successfully withdraw the desired amount, then the balance should not be adjusted and

the returned string should be "withdrawing $m (Insufficient Balance)". int getBalance() to retu the account balance.

. The transaction class has two data memb Account and List transaction. The class

Should implement the following three methods:

Void deposit (int money) to invoke the deposit method in the Account class. This should add the transaction message to the transactions list.

Void withdraw (int money) to invoke the withdraw method in the Account class. This should add the transaction message to the transaction list. list getTransaction() to return the transaction.

→

```
import java.util.*;
class Account {
int balance = 0;
public string deposit (int money){
 balance += money;
return "Depositing $" + money;
}
Public string withdraw (int money){
   if ( balance < money ){
return "withdraw $" + money" (Insufficient
    Balance");
```

MOHIT
Page No.:
Date:

```
}
else{
 balance = money;
return "withdraw $"+money;
 }
 }
 }
 Public int getbalance (){
  return balance;
 }
 }
 }
 class Transaction{
 Account account = new Account ();
 List <String> transactions= new AnoyList <>();
 Public Transaction (Account account ){
   this.account = account;
  }
  Public void deposit (int money){
 transactions.add (account.deposit (money));
 }
 Public void withdraw (int money ){
 transactions.addCaccount.withdraw (money))
 }
 Public List<String> getTransaction (){
    return transactions;
  }
```

2.2) write a Program of Producer and consume

→

```
Public class threadexample {
Public static void main (string args[])
throws InterruptedException
{
// object of a class that has both produce()
// and consume() methods
final Pc pc = new Pc();
// create producer thread
Thread t1 = new Thread (new Runnable
{
 override
Public Void run()
 {
try {
   Pc. consume();
}
catch (InterruptedException e){
  e.printStackTrace();
}
}
});
// start both threads
```

MOHIT
Page No.:
Date:

```
        t1.start();
        t2.start();
    //t1 finishes before t2
        t1.join();
        t2.join();
    //This class has a list shared by producer and
                consumer
    // size of list is 2.
    Lindlist<Integer> list = new LinkedList<>();
       int capacity = 2;
    // Function called by Producer thread
    Public void produce() throws Interruptedexception
    {
     int value=0;
     while (true) {
     synchronized (thus)
      {
     // Producer thread wait while list
     //  is full
    while (list.size() == capacity)
        wait();
    System.out.println(" Producer Produced-" +value
    // to insert the jobs in the list
       list.add(value++);
```

```
// notifies the consumer thread that
// now it can start consuming
 notify ();
// makes the working of program easier
// to understand
Thread.sleep (1000);
}
}
} // Function called by consumer thread
Public Void consume () throws Interrupted
 Exception
 {
 while (true){
 synchronized (this)
{
// consumer thread waits while list
 // is empty
 while (list.size()==0)
 Wait ();
// to retrive the first
 int val = list.removeFirst ();
System.out.println("Consumer consumed -"+val
// wake up producer thread
    notify ();
   // and sleep
```

MOHIT
Page No. :
Date:

```
        Thread.sleep(1000);
    }
}
}
}
}
```

4) You are required to compute the power of number by implementing a calculator. Create a class Mycalculator which consists of a single method long power (int, int). This method takes two integers, n and p, as parameters and finds. if either p or n is negative, then the method must throw and exception which says "n and p should not be negative, ~~then the~~ Also, if both and are zero, then the method must throw an exception which says "n and p should not be zero."
For example, -4 and -5 would result in java. long exception : n or p should not be negative
Complete the function power in class Mycalculator and return the appropriate result after the power operation or an appropriate exception as detailed above.

→
```
import java.io.*;
import java.util.*;
import java.io*;
// write your code here
class calculator
{
```

MOHIT
Page No.:
Date:

```
Public int Power (int n, int p) throws exception
{
 if (n>=0 && p>=0)
 {
 return (int) Math.pow(n,p);
}
else {
throw new exception ("n and p should be non-negative");

}}}
class Solution {
Public static void main (String args[])
Scanner in = new Scanner (System.in);
    int t = in.nexint ();
     while (t-->0) {
  int n= in.nextInt ();
  int p= in.nextInt ();
Calculator mycalculator = new calculator ();
 try {
 Int ans= mycalculator.Power (n,p);
 System.out.println (ans);
 }
 catch (Exception e) {
System.out.println (e.getmessage ());
```

```
}
in.close();
}
}
```

MOHIT
Page No.:
Date:

Q.5] Java Program showing execution of multiple tasks with a single thread.

→

```
class TestMultitasking5{
Public static void main (String args[])
{
Runnable r1 = new runnable (){
 Public void run (){
System.out.println ("task one");
}}
Runnable r2 = new runnable (){
 Public void run (){
System.out.println ("task two");
}}
Thread t1 = new Thread (r1);
Thread t2 = new Thread (r2);
 t1.start ();
 t2.start ();
 }
 }
```

MOHIT
Page No.:
Date:

Q.6] Java Program showing two treads working simultaneously upon two objects.

```
Class ThreadDemo extends Thread {
 Privale Thread t;
 Private string threadName;
ThreadDemo (string name) {
threadName = name;
System.out.Println ("creating" + threadName);
}
Public Void run () {
System.out.println ("Running" + threadName);
try {
for (int i=4; i>0; i-) {
System.out.println ("Thread :" + threadName + ", " + i
// let the thread sleep for a while
Thread.sleep (50);
}
}
catch (InterruptedException e)
{
System.out.Println ("Thread" + threadName + "exi
}
public Void start () {
```

MOHIT
Page No.:
Date:

```
System.out.Println("starting"+threadName);
  if (t == null) {
t = new Thread (this, threadName);
  t.start();
}
}}
Public Class Testthread {
Public static void main (String args [] ) {
ThreadDemo T1 = new ThreadDemo ("Thread
T1.start();
ThreadDemo T2 = new ThreadDemo ("Thread-
T2.start();
}
}
```

MOHIT
Page No.:
Date:

Q.7] Java Program showing two threads working acting upon a single object.

→
```
Public class one extends Thread {
PrintNumbers P;
 int i=1;
Public one (PrintNumbers P){
 this.P=P;
}
 @ override
Public void run (){
 int Pre=1;
while (Prev<1111){
 P.Printone(Prev);
 prev = (int)
(Prev + Math.pow (10,i));
 i=i+1;
}}}
Public class two extends thread{
 int i=1;
 PrintNumber P;
Public Two (PrintNumbers p)
{
 this.p=p;
```

```
}
@override
Public void run (){
 int prev = 2;
while (prev < 2222){
p.printTwo (prev);
prev = (int) (prev+2 * Math.pow (10,i));
   i = i+1;
}}}
Public class printtest{
Public static void main (string args[]) {
printNumbers b = new printNumbers ();
 one firstThread = new one (b);
 Two secondthread = new two (b);
firstThread.setName ("first :");
 firstthread.start ();
 secondthread.start ();
}
}
```

MOHIT
Page No.:
Date:

Q.8] Java Program with 2 threads which prints Alternatively.

→

```
Public class Test
{
Static int count = 0;
Public Static Void main (String args[])
throwsInterruptedException {
final object lock = new object ();
Thread t1 = new Thread (new runnable (){
  @ Override
  Public Void run (){
for (int i=0; i<10; i++){
Synchronized (lock){
count ++;
System.out.println ("count incremented to"+count+
"by"+ thread.currentthread ().getName ());
try{
  lock.Wait ();
  lock. notify ();
} catch (interruptedException e){
  e. PrintStackTrace ();
}
}}
```

```
});
Thread t2 = new thread (new Runnable(){
  @ oVerride
 Public Void run(){
for (int i=0; i<10 ; i++){
synchronized (lock){
 lock. notify();
  count ++;
System. out. println("count incremented to"+
count + " by" + Thread. currentThread(). getName
try{
    lock. wait(); }
catch (interruptedexception e){
 e. printstacktrace ();
}
}
});
 t1. start();
t2. start ();
t1. join();
t2. join();
}}
```

MOHIT
Page No.:
Date:

Q.9] java program to start one thread more than once

→ No, after starting a thread, it can never be started again. if you does so an IllegalthreadStateException is thrown. In such case, thread will run once but for second time, it will throw exception. Let's understand it by the example given below:

```
Public class Testthreadtwice1 extends thread {
  Public void run (){
System.out.println ("running...");
}
Public static void main (String args[]){
Testthreadtwice1 t1 = new Testthreadtwice1();
   t1.start();
   t1.start();
   }
   }
```

MOHIT
Page No.:
Date:

.10) Java Program to check currentThread in multi threading Concept.

→
```
Public class Test extends Thread
{
Public static void main (String args[]) {
// getting reference to main thread
Thread t = Thred.currentThread();
// getting name of main thread system.out.println
("current thread:" + t.getName());
// changing the name of main thread
t.setname ("Geeks");
System.out.println("After name change:"+t.getname());
// getting priority of main thread
System.out.println(" Main thread priority:"t.
getpriority());
// setting priority of main thread to MAx(10)
t.setpriority (MAx_PRIORITY);
System.out.println(" Main thread new priority:"
+ getpriority());
for (int i =0; i<2 ;i++){
System.out.println (" Main thread"); }
// Main thread creating a child thread
childthread ct = new childthread();
```

```
// getting Priority of child thread
// which will be inherited from main thread
// as it is created by main thread
System.out.println(" child thread priority:"+ct.
  getPriority());
// Setting Priority of main thread to MIN(1)
ct.setPriority(MIN_PRIORITY);
System.out.println(" child thread new priority:"
  ct.getPriority.());
  // Starting child thread
  ct.start();}
} // child thread class
Class childThread extends Thread{
  Public void run(){
for (int i =0; i<2; i++){
System.out.println(" child thread");}}
}
```

MOHIT
Page No.:
Date:

Q.11) Java program to create a server with 2 threads to communicate with serval clients.

→
```
import java.io*;
import java.text.*;
import java.Util.*;
import java.net.*; //server class
Public class server {
Public static void main (String args[])
  throwsIoException {
// server is listening on port 5056
serversocket ss = new serversocket (5056);
// running infinite loop for getting client request
while (true) {
Socket s = null;
try {
// Socket object to receive incoming client requests
  s = s.s.accept ();
System.out.println ("A new client is connected:"+s)
// obtaining input and out streams
DataInputStream dis = new DataInputStream(s.get-
InputStream ()); DataInputStream dos = new
DataOutputstream (s.getoutput.Stream ());
System.out.println ("Assigning new thread
    for this client");
```

```
// create a new thread object
Thread t = new ClientHandler (s, dis, dos);
// invoking the start() method
t.start();
}
catch (Exception e){
 s.close();
e.printStackTrace();
}}}
} // clientHandler class
class ClientHandler extends Thread{
DataFormat fordate = new SimpleDateFormat
  ("yyyy/MM/dd");
DateFormat fortime=new SimpleDateFormat
     ("hh:mm:ss");
final DataInputStream dis;
final DataOutputstream dos;
 final Socket s;
  // constructor
public ClientHandler (Socket s, DataInput.
stream dis, DataOutputstream dos)
{
this.s=s;
this.dis= dis;
this.dos = dos;}
```

```
@ oVerrid
Public void run (){
 String received;
 String toreturn;
 while (true){ // Ask user what he want
dos.writeUTF ("What do you want? [Date|T
\n" + " TYpe Exit to terminate connectio
// receive the answer from client
received = dis.readUTF ();
 if (received.equals ("Exit")){
System.out.println ("client"+this.s+" sends
System.out.println ("closing this connection"
  this.s.close ();
System.out.println ("connection closed");
  break; }
// creating data object
Data date = new Date();
// write on output stream based on t
 // answer from the client
 switch (received ){
  case "Date":
toreturn = fordate.format(date); dos.writ
break;
 Case "Time":
toreturn = fortime.format(date); dos.write
                                (toreturn
```

```
break;
default:
dos.writeUTF("Invalid input");
break;
}} catch (IOException e){
 e.printStackTrace();
}} try { // closing resources
this.dis.close();
this.dos.close();
} catch (IOException e){
 e.printStackTrace();}
 }}
```

MOHIT
Page No.:
Date:

Q.12) Java program to create a client that receive message from the server.

→
```
import java.io.*;
import java.net.*;
import java.util.scanner;
Public class client {
final static int serverPort = 1234;
Public static void main(String args[] throws UnknownHostException, IoException){
Scanner scn = new scanner (System.in);
// getting localhost ip
InetAddress ip = inetaddress.getbyname("localhost");
// establish the connection
Socket s = new Socket(ip, serverPort);
// obtaining input and out streams
DataInputStream dis = new datainputstream(s.getinputstream()); dataoutputstream dos = new data.outputstream(s.getoutputstream());
// sendmessage thread
Thread sendMessage = new Thread(new Runnable()
{
@ override
public void run(){
while (true) { // read the message to deliver.
```

```
String msg = scn.nextline();
try { // write on the output stream
   dos.writeUTF(msg);
} catch (IoException e){
 e.printStackTrace();
}}}}; //read message thread
Thread readMessage = new Thread (new Runnable
{
  @override
public void run(){
 while (true){
 try{
// read the message sent to this client
String msg = dis.readUTF();
System.out.println(msg);
} catch (IO Exception e){
 e.printStackTrace();
}}}  }};
sendMessage.start();
readMessage.start();
}}
```