Assignment -8

Q1. 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 ~~withdraw~~ deposit (int money) to substract money from the balance. This ~~shout~~ method should return a string that describes the withdraw transaction, i.e. "withdrawing $money". Note that, if there is insuffient balance to successfully withdraw the desired amount, then the balance should not be adjusted and the returned string should be "withdrawing $money (Insufficient Balance)". Int getBalance() to return the account-balance.
2. The transaction class has two data members Account and list transtions. The class should implement the following three methods.

the withdraw method in the Account class. This should add the transaction the transaction 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 transactions.

```
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)";
}
```

```
else {
balance -= money;
return "withdraw $" + money;}
}
public int getBalance() {
return balance; }
}
class Transaction {
Account account = new Account();
List<String> transactions = new Array-
List<>();
public Transaction (Account account){
this.account = account;
}
public void deposit (int money){
transactions.add(account.deposit
(money));}
public void withdraw (int money){
transactions.add(account.withdraw
(money));}
public list<String> getTransaction()
{
return transactions;
}
}
```

Page No.
Date

Q2. Write a program of producer and consumer.

```
import java.util.LinkedList;
public class Threadexample
{
public static void main(String[] args)
throws InterruptedException
{
Final PC pc = new PC();
Thread t1 = new Thread(new Runnable
() {
public void run()
{
try
{
pc.produce();
}
catch (InterruptedException e)
{
e.printStackTrace();
}
Thread t2 = new Thread(new Runnable
() {
public void run()
{
try
{
```

```
            pc.consume();
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
    }
});
t1.start();
t2.start();
t1.join();
t2.join();
}
public static class pc
{
LinkedList<Integer> list = new LinkedList
        ();
    int capacity = 2;
public void produce() throws Interru-
 ptedException
    {
    int value = 0;
    while(true)
        {
Synchronized (this)
    {
```

```
while (list.size() == capacity)
    wait();
System.out.println("producer produced
    -" + value);
    list.add(value++);
    notify();
    Thread.sleep(1000); }
    }
    }
public void consume() throws Inte-
rruptedException
    {
while (true) {
Synchronized (this)
    {
while (list.size() == 0)
    wait();
int val = list.removeFirst();
System.out.println("consumer consumed-
            +val);
        notify();
Thread.sleep(1000); }
        }
        }
```

output:- producer produced-0 consumer consumed-1
producer produced-1 producer produced-2
consumer consumed-0

Q.4 you are required to compute the power of a 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 an exception which says "n and p should not be negative". 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.lang.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
{
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.nextInt();
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(e.getMessage()); }
}
```

Q5. Java program showing Execution of multiple Tasks with a single thre

```
class Test+Multitasking5
{
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();
}
}
```

Output:

task one
task two

96. Java program showing two threads working simultaneously upon two object.

```
class prime extends Thread
  {
public void run()
  {
For (int i=1; i<=20; i++)
  {
if((i==2)||(i==3)) {
System.out.println("prime:"+i);}
elseif(!(i%2==0)||(i%3==0))
  {
System.out.println("prime:"+i);}
  }}
}
class Nonprime extends Thread
  {
public void run()
  {
For(int i=1; i<=20; i++)
  {
if((i==2)||(i==3)) { }
  }
elseif((i%2==0)||(i%3==0)){
```

```
System.out.println(" Non-prime : " +i);}
        }
    }
}
class Primethread
{
public static void main (String args[])
{
Thread p = new prime();
Thread np = new Nonprime();
    p.start();
    np.Start();
    }
    }
```

Page No.
Date

Q7. Java program showing Two Threads 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 prev=1;
while (prev<1111) {
    int prev=1;
p.printone(prev);
prev=(int)
(prev+math.pow(10,i));
    i=i+1; }
  }
}
public class run() Two extends Thread
  {
  int i=1;
printNumbers 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. First Thread = new one (b);
  Two Second Thread = new Two (b);
First Thread.setName ("First :") ; First
  Thread.start(); }
  }
```

output:

First : 1

second : 2

First : 11

Second : 22

First : 111

Q8. Java program with 2 Threads which prints Alternatively

```
public class Test {
  Static int count =0;
public static void main (String [] args)
  throws InterruptedException
    {
  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 current
Thread() . getName();
  try {
  lock.wait();
  lock.notify();
}catch (InterruptedException e) {
  e.printStackTrace(); }
  } }
  }
}
}
```

```
Thread t2 = new Thread(new Runna-
    ble() {
    @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();
    }
  }
```

99. 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 TestThread-
        Twice1();
t1.start();
t1.start();
}
}
```

output:

```
running
Exception in thread "main".
Java.lang.IllegalThreadStateException.
```

910. Java program to check currentThread in multi Threading concept.

```
public class Test extends Thread
  {
public static void main (String[] args)
  {
  Thread t = Thread.currentThread();
  Thread
    t.setName("Geeks");
System.out.println("After name change:"
          +t.getName());
System.out.println("main thread priori-
  ty:" + t.getpriority());
   t.setpriority(MAX_PRIORITY);
System.out.println("main thread new
  priority:" +t.getpriority());
   for(int i=0; i<2; i++){

System.out.println("main thread");
  }
childThread ct = new childThread();
System.out.println("child thread priority
  :"+ct.getpriority());
   ct.setpriority(MIN_PRIORITY);
System.out.println("child thread new
  priority:" +ct.getpriority());
```

```
        ct.start();
        }
    }
class childThread extends Thread
    {
public void run()
    {
for(int i=0; i<2; i++)
    {
System.out.println("child thread");
        }
    }
}
```

Q11. Java program to Create a server with 2 Threads to communicate with several clients.

```
import java.io.*;
import java.text.*;
import java.util.*;
import java.net.*;
public class Server {
public static void main (String args[])
throws IOException {

ServerSocket ss = new ServerSocket(5056);
    while (true)
        {
    Socket s = null;
    try
    {
  s = ss.accept();
System.out.println("A new client is
connected :" +s);
DataInputStream dis = new DataInput-
Stream(s.getInputStream()); DataOutput
Stream dos = new DataOutputStream(s.
getOutputStream());

System.out.println("Assigning new
```

```
thread for this client ");
Thread t = new clientHandler (s, dis,
            dos);
    t.start ();
    }
  catch (Exception e)
    {
    s.close();
    e.printStackTrace(); }
    }
  }
}
class clientHandler extends Thread
  {
DateFormat Fordate = new SimpleDate-
format ("yyyy/MM/dd");
DateFormat Fortime = new simpleDate
Format ("hh:mm:ss");
  Final DataInputStream dis;
  Final DataOutputStream dos;
  Final Socket s;

public clientHandler (Socket s, Data-
InputStream dis, DataOutputStream dos)
  {
    this.s = s;
```

```
this.dis=dis;
    this.dos = dos;
      }
    @override
public void run()
    {
String received;
String toreturn;
String while (true) {
        try {
dos.writeUTF ("What do you want?
[Date | Time]....\n" + "Type Exit to
terminate connection.");
  received = dis.readUTF ();

IF (received.equals ("Exit")).
      {
System.out.println ("client " + this.st
"Sends exit...");
System.out.println ("closing this connection
    this.s.close ();
System.out.println ("connection closed");
  break;
      }
  Date date = new Date ();
```

```
switch (received)
    {
    case "Date":
toreturn = fordate.format(date);
    dos.writeUTF(toreturn);
    break;
    case "Time":
toreturn = fortime.format(date);
    dos.writeUTF(toreturn);
    break;
    default;
dos.writeUTF("Invalid input");
    break; }
    }
catch (IOException e)
    {
e.printStackTrace(); }
    }
try {
this.dis.close();
this.dos.close();
} catch (IOException e) {
    e.printStackTrace(); }
    }
    }
```

Q12. Java program to create a client That Receive message from the server

```
import java.io.*;
import java.net.*;
import java.util.~~net~~ Scanner;
public class client
{
~~Final class client~~
Final static int serverport = 1234;
public static void main (String args[])
throws UnknownHostException, IOEx-
ception
{
Scanner scn = new Scanner (System.in);
InetAddress ip = InetAddress.getBynam-
e("localhost");
Socket s = new Socket (ip, Serverport);

~~Socket~~ s
DataInputStream dis = new DataInput-
Stream (s.getInputstream());
DataoutputStream dos = new data-
outputStream (s.getOutputstream());

Thread SendMessage = new Thread (new
Runnable()
```

```
{
    @override
public void run() {
while (true)
    {
String msg = scn.nextLine();
    try {
dos.writeUTF(msg);
    } catch (IOException e)
    {
e.printStackTrace(); } }
    }
    }
    }
Thread readmessage = new Thread
        (new Runnable()
        {
    @override
public void run()
    {
while(true){
    try{
String msg = dis.readUTF();
System.out.println(msg);
    } catch (IOException)
    {
```

```
            e.printStackTrace(); }
                }
            } }
        };
sendmessage.start();
    readmessage.start(); }
            }
```