Assignment ~ 8

1) 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. Following three methods
string deposit (int money) to substract 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 $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)
 {
int 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-1 }
  }
   class Transaction {
 Account account = new Account();
List<string> transactions = new Arraylist<>();
public transaction (Account account ;1
    this.account = account;
   }
   public void deposit(int money){
 transactions add(account.withdraw(money)) ;
  }
public list<string>get Transaction()
   {
   return transactions;
   }}
```

2] W.a.p. of procedure 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(InterruptedExcptione)
    {
    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 InterruptedEx-
                    ception
{
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 Interrupted-
                    Exception
{
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-0
producer produced-1 producer produced-2
consumer consumed-0.

public void consume() throws

You are required to compute the power of a number by implementing a calculator. Create a class MyCalculator. Create a class 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 & return the appropriate result after the power operation or an appropriate exception as detailed above.

```
import java.io.*;
import java.util.*;
import java.io.*;

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());
}
 }
```

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();
   }
    }
```

Output:-

task one
task two

6) Java programs showing Two Threads Working simultaneously upon Two objects

```
class Prime extends Thread
{
 public void run()
  {
  for(int i = 1; i<= 20; i++)
  {
 if((i == 2) || (i == 3))
  {
System.out.println(" Prime:" +i);
else if (!(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)) {
   }
    }
  else if((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();
}
}
```

7) Java program showing Two Threads Actin 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 = +;
while (Prev < 1111(){
P. print1one(prev);
prev = (int)
(prev + Math.pow(10, i));
i = i + 1;
}
}
public class Tuo 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 printNumber();
  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
Second : 222
```

8) 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 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();
}
  }
```

Java program to start one thread more than once

No. After starting a thread, it can be 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 TestThreadTwiset extends Thread{
  public void run()
  {
 system.out.println("running....");
  }
public static void main(string args[])
  {
Test ThreadTwiset t1 = new Test Thread-
                       Twice1();
      t1. start();
      t1. start();
    }}
```

Output:- running
Exception in thread "main"
Java.lang.IllegalThreadstateException.

10) Java program to check CurrentThread in multithreading concept.

```
public class Test extends Thread
  {
public static void main(string [] args)
  {
Thread t = Thread.currentThread();
   t. setName("Geeks");
system.out.println("After name change:"
                   ++ getName());
System.out.println("Main thread priority:"
          ++ 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");
  }
   {
    }
```

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());
DataOutputStream dos = new DataOutput-
    stream(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 Simple Date
    Format ("YYY (mm/dd");
```

```
DateFormat fortime = new SimpleDate
  Format("hh:mm:ss");
Final DataInputstream dis;
Final DataOutputstream dos;
Final socket s;
public clientHandler(socket s, DataInput
stream dis, DataOutputstream dos)
 {
   this.s =s;
this.dis = dis;
this.dos = dos;
}
@Override
public void run()
 {
string received;
String toreturn;
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.s+
      "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(); }
       }
  }
```

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)
Inetaddress ip = InetAddress.getBy Name
        ("localhost");
Socket s = new socket(ip, serverport);
DataInputstream dis = new DataInput
  -Stream(s.getInputstream());
  DataOutputstream dos = new DataOutputStre-
  am(s.getoutputstream());
Thread sendMessage = new. Thread new
                    Runnable()
  {
  @ override
public void run(){
 while(true)
 {
  String msg = scn.nextLine();
  try {
 dos.writeUTF(msg);
  } catch(IO Exception 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();
 }
  }
```