Que1.	Read a content from file: calculate number of sentences, words and characters from the file.

Code:-

```
importjava.io.*;

publicclass First {
publicstaticvoidmain(String[] args) throwsIOException{
        File file = newFile("C:\\Users\\Akshat\\Desktop\\Java Assignment-
3\\q1.txt");
FileInputStreamfileInputStream = newFileInputStream(file);
InputStreamReaderinputStreamReader = newInputStreamReader(fileInputStream);
BufferedReaderbufferedReader = newBufferedReader(inputStreamReader);

        String line;
intwordCount = 0;
intcharacterCount = 0;
intsentenceCount = 0;

while ((line = bufferedReader.readLine()) != null) {
if (line.equals(""))
            {
paraCount = paraCount + 1;
            }
else
            {
characterCount = characterCount + line.length();

String[] words = line.split("\\s+");
wordCount = wordCount + words.length;

String[] sentences = line.split("[!?.:]+");
sentenceCount = sentenceCount + sentences.length;
            }
        }
bufferedReader.close();

if (sentenceCount>= 1)
        {
paraCount++;
        }
System.out.println("Total number of characters = "+ characterCount);
System.out.println("Total number of words = "+ wordCount);
System.out.println("Total number of sentences = "+ sentenceCount);

    }
}
```

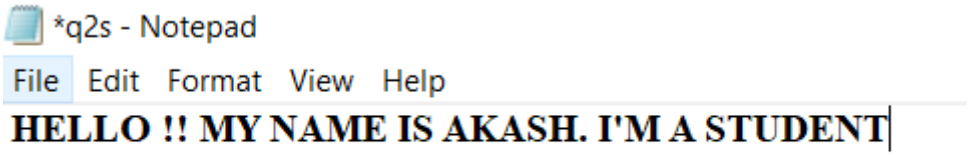
Output:-


```
Total number of characters = 40
Total number of words = 8
Total number of sentences = 3
```

2.	Read from a file convert it to uppercase and save it into another file.
Code:-	

```
importjava.io.*;
publicclass file_Upp {
publicstaticvoidmain(String[] args) throwsIOException
    {
        File file = newFile("C:\\Users\\Akshat\\Desktop\\Java Assignment-
3\\q2.txt");
FileReaderfr = newFileReader(file);
FileWriterfw = newFileWriter("C:\\Users\\Akshat\\Desktop\\Java Assignment-
3\\q2s.txt");
intupperchar; intch;

while((ch = fr.read()) != -1)
        {
upperchar = Character.toUpperCase(ch);
fw.write(upperchar);
        }
fr.close(); fw.close();
    }
}
```

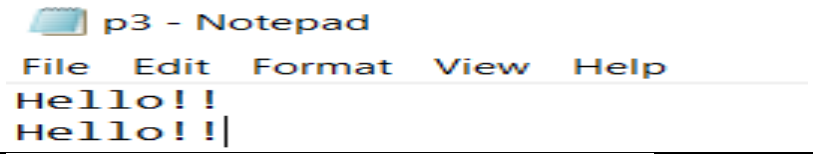
Output:-

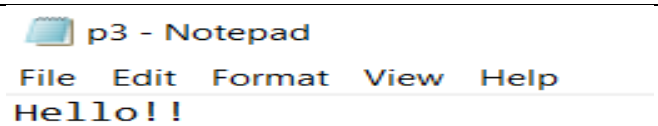


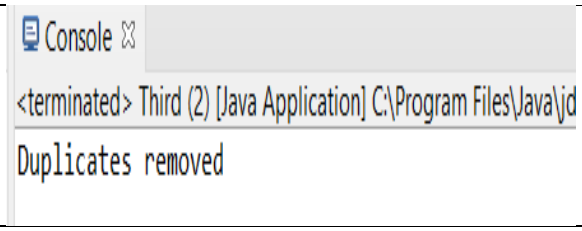
3.	Remove duplicate lines from a File.
Code:-	

```
importjava.io.*;
importjava.util.*;
publicclass Third {
publicstaticvoidmain(String[] args) throwsIOException {
        File file = newFile("C:\\Users\\Akshat\\Desktop\\Java Assignment-
3\\p3.txt");
FileReaderfr = newFileReader(file);
BufferedReaderbr = newBufferedReader(fr);

        Set<String>lines = new HashSet<String>(100);
        String line;
while ((line = br.readLine()) != null) {
lines.add(line);
        }
br.close();

FileWriterfw = newFileWriter(file);
BufferedWriterbw = newBufferedWriter(fw);
for (String finalline :lines) {
bw.write(finalline);
bw.newLine();
        }
bw.close();
System.out.println("Duplicates removed");
    }
}
```

Output:-







4.	Create a class called Student. Write a student manager program to manipulate the student information from files by using FileInputStream and FileOutputStream

Code:-

```
importjava.io.*;
importjava.util.Scanner;
publicclass Fourth {
static Scanner sc = new Scanner(System.in);
static File file = new File("C:\\Users\\Akshat \\Desktop\\Java Assignment-
3\\p4.txt");
static int i;
public static void main(String[] args) throwsIOException{
int selection;
Boolean check = true;

do
        {
System.out.println("\nEnter a choice. \n1: Add info \n2: Read info \n3:
Exit");
selection = sc.nextInt();
switch(selection)
            {
case 1:
                {
addinfo(); break;
                }
case 2:
                {
getinfo(); break;
                }
case 3:
                {
check = false; break;
                }
default:
                {
System.out.println("Please enter valid input");
                }
            }
}while(check);
    }
Public static void getinfo() throws IOException
    {
FileInputStreaminStream = new FileInputStream(file);
while ((i = inStream.read()) != -1)
        {
System.out.print((char)i);
        }
inStream.close();
    }

Public static void addinfo() throws IOException
    {

FileOutputStreamoutStream = new FileOutputStream(file, true);

sc.nextLine();
System.out.println("Enter name of student");
        String name = sc.nextLine();

System.out.println("Enter age of student");
```

```java
Int age = sc.nextInt(); sc.nextLine();

System.out.println("Enter college name of Student");
        String college = sc.nextLine();

System.out.println("Enter branch of Student");
        String branch = sc.nextLine();

System.out.println("Enter batch of Student");
        String batch = sc.nextLine();

        String str = "\nName: "+ name +", Age: "+ age +", College: "+
college + ", Branch: "+ branch + ", Batch: "+ batch;
Byte strTObyte[] = str.getBytes();
outStream.write(strTObyte);
outStream.close();
    }
}
```

Output:-

```
Enter a choice.
1: Add info
2: Read info
3: Exit
1
Enter name of student
Manav
Enter age of student
18
Enter college name of Student
PDEU
Enter branch of Student
ICT
Enter batch of Student
2020-24

Enter a choice.
1: Add info
2: Read info
3: Exit
2

Name: Manav, Age: 18, College: PDEU, Branch: ICT, Batch: 2020-24
Enter a choice.
1: Add info
2: Read info
3: Exit
3
```

p4 - Notepad

File Edit Format View Help

```
Name: Manav, Age: 18, College: PDEU, Branch: ICT, Batch: 2020-24
```

5.	Refine the student manager program to manipulate the student information from files by using the BufferedReader and BufferedWriter

Code:-

```
Import java.io.*;
Import java.util.Scanner;
Public class student {
static Scanner sc = new Scanner(System.in);
static File file = new File("C:\\Users\\Akshat \\Desktop\\Java Assignment-
3\\p5.txt");
static int i;
public static void main(String[] args) throws IOException{
int selection;
Boolean check = true;

do
        {
System.out.println("\nEnter a choice. \n1: Add info \n2: Read info \n3:
Exit");
selection = sc.nextInt();
switch(selection)
            {
case 1:
                {
addinfo(); break;
                }
case 2:
                {
getinfo(); break;
                }
case 3:
                {
check = false; break;
                }
default:
                {
System.out.println("Please enter valid input");
                }
            }
}while(check);
    }
Public static void getinfo() throws IOException
    {
FileReaderfr = new FileReader(file);
BufferedReaderbuffRead = new BufferedReader(fr);

while ((i = buffRead.read()) != -1)
        {
System.out.print((char)i);
        }
buffRead.close();
    }

Public static void addinfo() throws IOException
    {

FileWriter fw = new FileWriter(file, true);
BufferedWriter buffWrite = new BufferedWriter(fw);
sc.nextLine();
System.out.println("Enter name of student");
        String name = sc.nextLine();
```

```
System.out.println("Enter age of student");
Int age = sc.nextInt(); sc.nextLine();

System.out.println("Enter college name of Student");
        String college = sc.nextLine();

System.out.println("Enter branch of Student");
        String branch = sc.nextLine();

System.out.println("Enter batch of Student");
        String batch = sc.nextLine();

        String str = "\nName: "+ name +", Age: "+ age +", College: "+
college + ", Branch: "+ branch + ", Batch: "+ batch;
buffWrite.write(str);
buffWrite.close();
    }

}
```

Output:-

```
Enter a choice.
1: Add info
2: Read info
3: Exit
1
Enter name of student
XYZ
Enter age of student
00
Enter college name of Student
ABCD
Enter branch of Student
PQR
Enter batch of Student
1234-56

Enter a choice.
1: Add info
2: Read info
3: Exit
2

Name: XYZ, Age: 0, College: ABCD, Branch: PQR, Batch: 1234-56
Enter a choice.
1: Add info
2: Read info
3: Exit
3
```

p5 - Notepad

File Edit Format View Help

```
Name: XYZ, Age: 0, College: ABCD, Branch: PQR, Batch: 1234-56
```

6.	Write a program to manipulate the information from files by using the Reader and Writer class. Assume suitable data.
File Writer	
Code:-	

```
Import java.io.File;
Import java.io.IOException;
Public class rd_wr {
Public static void main(String[] args) {

        File file = newFile("C:\\Users\\Akshat \\Desktop\\Java Assignment-
3\\p1.txt");
try {
Boolean createFile = file.createNewFile();
if (createFile) {
System.out.println("New File is created.");
}else {
System.out.println("File already exists.");
            }
        } catch (IOExceptione) {
e.printStackTrace();
        }
     }
}
```

Output:-

```
File already exists.
```

File Reader

Code:-

```
Import java.io.FileReader;
Public class read {
Public static void main(String args[])throws Exception{
FileReaderfilereadObj=newFileReader("C:\\Users\\Akshat\\Desktop\\Java
Assignment-3\\p1.txt");
Int iterator;
while((iterator=filereadObj.read())!=-1)
System.out.print((char)iterator);
filereadObj.close();
    }
}
```

Output:-

Hello!! I am Akash

7.	Write a program “DivideByZero” that takes two numbers a and b as input, computes a/b, and invokes Arithmetic Exception to generate a message when the denominator is zero.
Code:-	

```
importjava.util.Scanner;
public class div_by_zero {
public static void main(String[] args)

 {

Int x,y;

   Scanner input=new Scanner(System.in);

try {

System.out.print("Enter first integer : ");

x=input.nextInt();

System.out.print("Enter second integer : ");
y=input.nextInt();

System.out.println(x + " / " + y +" = " + (x/y));

 }

catch(ArithmeticExceptione)
  {
       System.out.println("Denominator Cannot be Zero while Integer
Division");
  }
 }
}
```

Output:-

```
Enter first integer :  5
Enter second integer :  0
Denominator Cannot be Zero while Integer Division
```

8.	Write a program to show the use of nested try statements that emphasizes the sequence of checking for catch handler statements.
Code:-	

```
Public class nest_try {
Public static void main(String [] args)
{
try
{
int[] a =newint[10];
// displaying element at index 12

System.out.println(a[12]);
// another try block
try {
System.out.println("Division");
Int res = 100/ 0;
            }
catch (ArithmeticExceptionex2) {
System.out.println("Sorry! Division by zero isn't feasible!");
            }
        }
catch (ArrayIndexOutOfBoundsExceptionex1) {
System.out.println("ArrayIndexOutOfBoundsException");
        }
    }
}
```

Output:-

```
ArrayIndexOutOfBoundsException
```

9.	Write a program to create your own exception types to handle situation specific to your application (Hint: Define a subclass of Exception which itself is a subclass of Throwable).
Code:-	

```
Public class myexception {
Public static void sum(int a,int b) throws MyException{
if(a<0)
        {
Throw new MyException(a);
        }
else
        {
System.out.println(a+b);
        }
    }

publicstaticvoidmain(String[] args)
    {
try
        {
sum(-10, 10);
        }
catch(MyExceptionme)
        {
System.out.println(me);
        }
    }
}

classMyExceptionextends Exception
{
privateintex;
MyException(inta)
    {
ex = a;
    }
public String toString()
    {
return"MyException[" + ex +"] is less than zero";
    }
}
```

Output:-	MyException[-10] is less than zero

10.	Write a small application in Java to develop Banking Application in which user deposits the amount Rs 1000.00 and then start withdrawing of Rs 400.00, Rs 300.00 and it throws exception “Not Sufficient Fund” when user withdraws Rs. 500 thereafter.

Code:-

```
Import java.util.Scanner;
Public class bank {
static Scanner sc = new Scanner(System.in);
    String name, account;
Int bal;
    bank ()
    {
name = "(Customer name)";
account = "(account value)";
bal = 1000;
    }
publicstaticvoidmain(String[] args)
    {
        bank object1 = new bank();

Int selection; boolean check = true;
Int amount;
do
        {
System.out.println("\nEnter a choice \n1: Withdraw \n2: Deposit \n3: Exit");
selection = sc.nextInt();
switch(selection)
            {
case 1:
                {
System.out.println("Please enter amount to be withdrawn");
amount = sc.nextInt();
if(checkBal(object1, amount))
                    {
                        (object1).bal = (object1).bal - amount;
System.out.println("Amount withdrawn: "+ amount +", New Balance: "+
(object1).bal);
                    }
else
                    {
System.out.println("Insufficient balance to withdraw given amount");
                    }
break;
                }

case 2:
                {
System.out.println("Please enter amount to be deposited");
amount = sc.nextInt();
                    (object1).bal = (object1).bal + amount;
System.out.println("Amount deposited: "+ amount +", New Balance: "+
(object1).bal);
break;
                }

case 3:
                {
check = false; break;
                }

default:
```

```
                    {
System.out.println("Enter valid input");
                    }
                }
}while(check);
    }

Public static Boolean checkBal(bank p1, int withdrawing)
    {
if(p1.bal>= withdrawing)
        {
Return true;
        }
else
        {
Return false;
        }

    }
}
```

Output:-

```
Enter a choice
1: Withdraw
2: Deposit
3: Exit
2
Please enter amount to be deposited
420
Amount deposited: 420, New Balance: 1420

Enter a choice
1: Withdraw
2: Deposit
3: Exit
1
Please enter amount to be withdrawn
1500
Insufficient balance to withdraw given amount

Enter a choice
1: Withdraw
2: Deposit
3: Exit
```

11.	Write a program to handle ArrayIndexOutOfBounds exception for binary search.
Code:-	

```
Public class bsearch {
Public static void main(String[] args) {
String[] arr = {"ABC","EFG","PQR","XYZ"};


for(inti=0;i<=arr.length;i++) {


System.out.println(arr[i]);

        }

    }
}
```

Output:-

```
ABC
EFG
PQR
XYZ
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4
```

12.	Write a Java Program that demonstrates thread class and few methods.
Code:-	

```
Public class thread_demo {

Public static voidmain(String[] args) {
            Thread t = new Thread("My first thread");

t.start();
            String str = t.getName();
System.out.println(str);

        }
    }
```

Output:-	

```
My first thread
```

13.	Write a program to demonstrate thread example by implementing runnable interface.
Code:-	

```
Public class thread_demo implements Runnable{
publicvoidrun()
    {
System.out.println("Now the thread is running ...");
    }

publicstaticvoidmain(String[] args)
    {

        Runnable r1 = new thread_demo();

        Thread th1 = newThread(r1, "My new thread");

th1.start();

        String str = th1.getName();
System.out.println(str);
    }
}
```

Output:-

```
My new thread
Now the thread is running ...
```

14.	Write a program to demonstrate priorities among multiple threads.
Code:-	

```
Import java.lang.*;
publicclass mult_th extends thread {

public void run()
    {
System.out.println("Inside run method");
    }

Public static void main(String[] args)
    {
        Multi_th t1 = new Multi_th();
        Multi_th t2 = new Multi_th();
        Multi_th t3 = new Multi_th();

System.out.println("t1 thread priority : "
                + t1.getPriority());

System.out.println("t2 thread priority : "
                + t2.getPriority());

System.out.println("t3 thread priority : "
                + t3.getPriority());

t1.setPriority(2);
t2.setPriority(5);
t3.setPriority(8);

System.out.println("t1 thread priority : "
                + t1.getPriority());

System.out.println("t2 thread priority : "
                + t2.getPriority());

System.out.println("t3 thread priority : "
                + t3.getPriority());


System.out.println(
"Currently Executing Thread : "
                        + Thread.currentThread().getName());

System.out.println(
"Main thread priority : "
                        + Thread.currentThread().getPriority());


Thread.currentThread().setPriority(10);

System.out.println(
"Main thread priority : "
                        + Thread.currentThread().getPriority());
    }
}
```

Output:-

```
t1 thread priority : 5
t2 thread priority : 5
t3 thread priority : 5
t1 thread priority : 2
t2 thread priority : 5
t3 thread priority : 8
Currently Executing Thread : main
Main thread priority : 5
Main thread priority : 10
```

15.	Write a program to demonstrate multithread communication by implementing synchronization among threads (Hint: you can implement a simple producer and consumer problem).

Code:-

```
Import java.util.*;
public class th_comm {
public static void main(String[] args)
throws InterruptedException
    {
final PC pc = new PC();

        Thread t1 = newThread(new Runnable() {
@Override
publicvoidrun()
            {
try {
pc.produce();
                }
catch (InterruptedExceptione) {
e.printStackTrace();
                }
            }
        });

        Thread t2 = new Thread(new Runnable() {
@Override
Public void run()
            {
try {
pc.consume();
                }
catch (InterruptedExceptione) {
e.printStackTrace();
                }
            }
        });

t1.start();
t2.start();

t1.join();
t2.join();
    }

Public static class PC {

        LinkedList<Integer>list = new LinkedList<>();
intcapacity = 2;

public void produce() throws InterruptedException
        {
Int value = 0;
while (true) {
synchronized (this)
                {
while (list.size() == capacity)
wait();

System.out.println("Producer produced-"
                            + value);
```

```
list.add(value++);

notify();

Thread.sleep(10);
                }
            }
        }
publicvoidconsume() throwsInterruptedException
        {
while (true) {
synchronized (this)
                {
while (list.size() == 0)
wait();

intval = list.removeFirst();

System.out.println("Consumer consumed-"
                            + val);
notify();

Thread.sleep(10);
                }
            }
        }
    }
}
```

Output:-

```
Producer produced-0
Producer produced-1
Consumer consumed-0
Consumer consumed-1
Producer produced-2
Producer produced-3
Consumer consumed-2
Consumer consumed-3
Producer produced-4
Producer produced-5
Consumer consumed-4
Consumer consumed-5
Producer produced-6
Producer produced-7
Consumer consumed-6
Consumer consumed-7
Producer produced-8
Producer produced-9
Consumer consumed-8
Consumer consumed-9
Producer produced-10
Producer produced-11
Consumer consumed-10
Consumer consumed-11
Producer produced-12
Consumer consumed-12
Producer produced-13
Producer produced-14
Consumer consumed-13
Consumer consumed-14
Producer produced-15
Producer produced-16
Consumer consumed-15
Consumer consumed-16
Producer produced-17
```

List of labwork programs:

Que 1	Create single thread by extending class thread.
Code:	

```
class Message extends Thread{
    String message;
    int time;
    Message(String msg,int timeInt){
        message = msg;
        time = timeInt;
    }
    public void run(){
        try{
            Thread.sleep(this.time);
            System.out.println(message);
        }
        catch(InterruptedException e){
            System.out.println(e);
        }
    }
}

public class C_Thread{
    public static void main(String[] args) {
        Message m1 = new Message("Good Morning",500);
       // Message m2 = new Message("Good Evening",1000);
        //Message m3 = new Message("Good Night",1500);
        m1.start();
        //m2.start();
        //m3.start();
    }
}
```

Output:

```
Good Morning
```

Que 2	Check the details of current thread (id, name, priority, setName etc.)

Code:	public class Threads_id {
	public static void main(String[] args) {
	Thread obj = Thread.currentThread();
	System.out.println("Current thread: "+obj);
	
	obj.setName("Current thread");
	obj.setPriority(3);
	System.out.println("After changing the name:" + obj.getName());
	System.out.println("Priority of Thread:" + obj.getPriority());
	System.out.println("Id of Thread:" + obj.getId());
	}
	}

Output:	Current thread: Thread[main,5,main] After changing the name:Current thread Priority of Thread: 3 Id of Thread: 1

Que 3	Create a single thread by implementing runnable interface: a. Take thread object in main b. Take thread object in class that is implementing runnable

Code:	// a
	public class ThreadObjInMain implements Runnable{
	@Override
	public void run()
	{
	System.out.println("New thread ");
	}
	public static void main(String[] args)
	{
	System.out.println("Main thread running");
	ThreadObjInMain th = new ThreadObjInMain();
	Thread t = new Thread(th);
	t.start();
	}
	}

```
/* b
        class MultiThread implements Runnable{
            public void run(){
                Thread t = new Thread();
                System.out.println(t.currentThread().getName());
            }
        }

        public class RunnableInsideclass {
            public static void main(String[] args) {
                Thread t1 = new Thread(new MultiThread());
                t1.start();
            }
}
*/
```

Output:

```
Main thread running
New thread
```

```
Thread-0
```

Que 4	Create multiple threads by extending thread class and assign same task to all the threads.

Code:

```
        class mul_th extends Thread{
            int display = 10;
            public void run(){
                try{
                    System.out.println("Current thread which is running is
        "+Thread.currentThread().getName());
                    System.out.println(display);
                    Thread.sleep(1000);

                }
                catch(Exception e){
                    System.out.println(e);
                }
            }
        }

        public class MultiThread_sameTask {
            public static void main(String[] args) {
                int n = 4;
                for (int i = 1; i < n+1; i++) {
```

```
                ThreadSameTask tst = new ThreadSameTask();
                tst.start();
            }
        }
}
```

Output:

```
Current thread which is running is Thread-2
10
Current thread which is running is Thread-3
10
Current thread which is running is Thread-1
10
Current thread which is running is Thread-0
10

Process finished with exit code 0
```

Que 5	Create multiple threads by implementing runnable and assign different task to each thread (prime number, whether given number is Armstrong or not).

Code:

```
class PrimeNumber extends Thread{
    int num;
    PrimeNumber(int num){
        this.num = num;
    }

    public void run(){
        boolean flag = false;
//          try {
//              Thread.sleep(1000);
//          } catch (InterruptedException e) {
//              throw new RuntimeException(e);
//          }
        for (int i = 2; i <= num / 2; ++i) {
            // condition for nonprime number
            if (num % i == 0) {
                flag = true;
                break;
            }
        }

        if (!flag) {
            System.out.println(num + " is a prime number.")
;
        }
        else {
            System.out.println(num + " is not a prime numbe
r.");
        }
```

```
    }
}

class armStrong extends Thread{
    int num;
    armStrong(int num){
        this.num = num;
    }

    public void run(){

        int originalNumber, remainder, result = 0;

        originalNumber = num;

        while (originalNumber != 0)
        {
            remainder = originalNumber % 10;
            result += Math.pow(remainder, 3);
            originalNumber /= 10;
        }

        if(result == num)
            System.out.println(num + " is an Armstrong numb
er.");
        else
            System.out.println(num + " is not an Armstrong
number.");

    }
}

public class DiffrentTask {
    public static void main(String[] args){
        PrimeNumber task1 = new PrimeNumber(23);
        PrimeNumber task3 = new PrimeNumber(235);
        armStrong task2 = new armStrong(372);
        task1.setPriority(7);
        task3.setPriority(6);
        task2.setPriority(6);
        task1.start();
//        task2.join();
        task2.start();
        task3.start();
    }
}
```

o/p:

```
23 is a prime number.
372 is not an Armstrong number.
235 is not a prime number.
```

Que 6 Use of synchronization

Code:

```
class syn{
    synchronized void Print_func(String name){
        for (int i = 0; i < 3; i++) {
            System.out.println(name +i);
            try {
                Thread.sleep(700);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }
}
class yug extends Thread{
    syn pn1;
    yug(syn pn1){
        this.pn1 = pn1;
    }
    public void run(){
        pn1.Print_func("Yug");
    }
}
class neel extends Thread{
    syn pn1;
    neel(syn pn1){
        this.pn1 = pn1;
    }

    public void run(){
        pn1.Print_func("Neel");
    }

}
public class PrintingName {
    int a;
    public static void main(String[] args) throws Interru
ptedException {
        PrintName v=new PrintName();
        neel n=new neel(v);
        neel n1=new neel(v);
        yug y=new yug(v);
//        yug y1=new yug(v);
```

```
        n.start();
        y.start();
        n1.start();
//        y1.start();
    }

}
```

Output:

```
Neel0
Neel1
Neel2
Yug0
Yug1
Yug2
Neel0
Neel1
Neel2
```

Que 7

Implement producer consumer problem (all four)
producer : p consumer : c
a. single p single c
b. single p multiple c
c. multiple p single c
d. multiple p multiple c

Code:

```
public class Consumer_Producer
{
    public static void main(String[] args)
    {
        Resourse resourse=new Resourse();
        Thread p=new Producer("P",resourse);
        Thread c=new Thread(new Consumer("C",resourse));
        p.start();
        c.start();
    }
}

class Resourse
{
    Boolean isproduced=false;
    int data;
    synchronized void put(int x) throws Exception
    {
        if(isproduced)
        {
            wait();
        }
```

```
            this.data = x;
            isproduced=true;
            notifyAll();

        }
        synchronized int get() throws Exception
        {
            if(!isproduced)
            {
                wait();
            }
            isproduced=false;
            notifyAll();
            return data;
        }
    }

    class Producer extends Thread
    {
        String name;
        Resourse res;
        Producer(String name, Resourse res)
        {
            this.name=name;
            this.res=res;
        }
        public void run() //not mandatory
        {
            try
            {
                for(int i=0;i<3;i++)
                {
                    res.put(i);
                    System.out.println("Produced = " + i);
                    Thread.sleep(1000);
                }
            }
            catch (Exception e)
            {
            }
            finally
            {
                System.out.println("job is done by producer..");
            }
        }
    }
```

```
class Consumer implements Runnable
{
    String name;
    Resourse res;
    Consumer(String name, Resourse res)
    {
        this.name=name;
        this.res=res;
    }

    public void run()   //Mandatory
    {
        try
        {
            for(int i=0;i<3;i++)
            {
                System.out.println("Consumed = " + res.get());
                Thread.sleep(1000);
            }
        }
        catch (Exception e)
        {
        }
        finally
        {
            System.out.println("job is done by consumer.");
        }
    }
}
```

Output:

```
Produced = 0
Consumed = 0
Produced = 1
Consumed = 1
Produced = 2
Consumed = 2
job is done by producer.
job is done by consumer.
```