**Q1. WAP in java to display an object.**

**Source code:**

class A

{

int a;

public void input()

{

a=5;

}

public void display()

{

System.out.println("Value of a = "+a);

}

}

class B

{

public static void main(String args[])

{

A a1=new A();

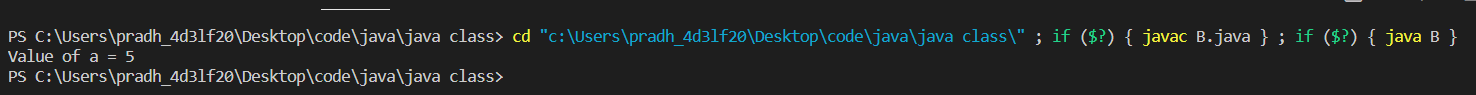
a1.input();

a1.display();

}

}

**Output:**



**Q2. WAP in java to print the factorial of a number.**

**source code:**

import java.util.\*;

class factorial

{

public static void main(String args[])

{

int f=1,n;

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number");

n=sc.nextInt();

while(n>0)

{

f=f\*n;

n--;

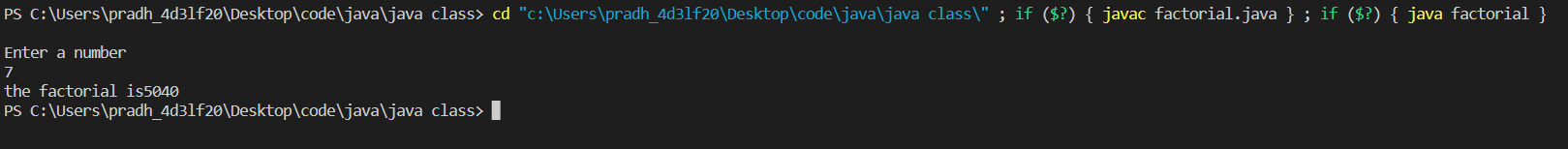
}

System.out.println("the factorial is"+f);

}

}

**Output:**



**Q3. WAP in java to print the Fibonacci series.**

**Source code:**

import java.util.\*;

class Fibonacci

{

public static void main(String args[])

{

int a = 0, b = 1, c, n;

Scanner sc = new Scanner(System.in);

System.out.println("enter the no. of terms");

n = sc.nextInt();

c = a+b;

System.out.print(a+"\t"+b+"\t"+c);

for(;n>3;n--)

{

a = b;

b = c;

c = a+b;

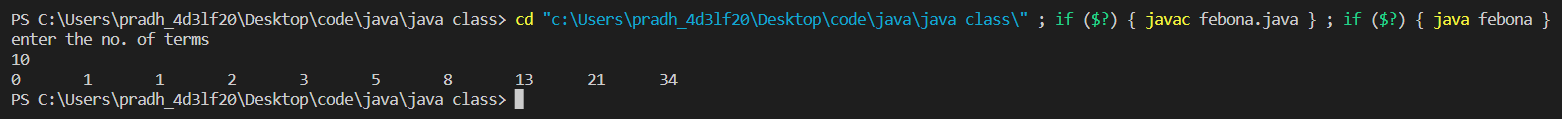
System.out.print("\t"+c);

}

}

}

**Output:**



**DATE:**

**Q4. WAP in java to create a ‘complex’ class and add two complex numbers.**

**Source code:**

import java.util.\*;

class complex

{

int real,imagin;

void input(int x,int y)

{

real=x;

imagin=y;

}

void display()

{

System.out.println(real+"+"+imagin+"i");

}

complex add(complex a,complex b)

{

complex result=new complex();

result.real=a.real+b.real;

result.imagin=a.imagin+b.imagin;

return result;

}

}

class cp

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter real and imaginary part of Complex Number 1:\n");

int a=sc.nextInt();

int b=sc.nextInt();

complex obj1=new complex();

obj1.input(a,b);

obj1.display();

System.out.println("Enter real and imaginary part of complex number 2:\n");

int c=sc.nextInt();

int d=sc.nextInt();

complex obj2=new complex();

obj2.input(c,d);

obj2.display();

complex obj3=new complex();

obj3=obj3.add(obj1,obj2);

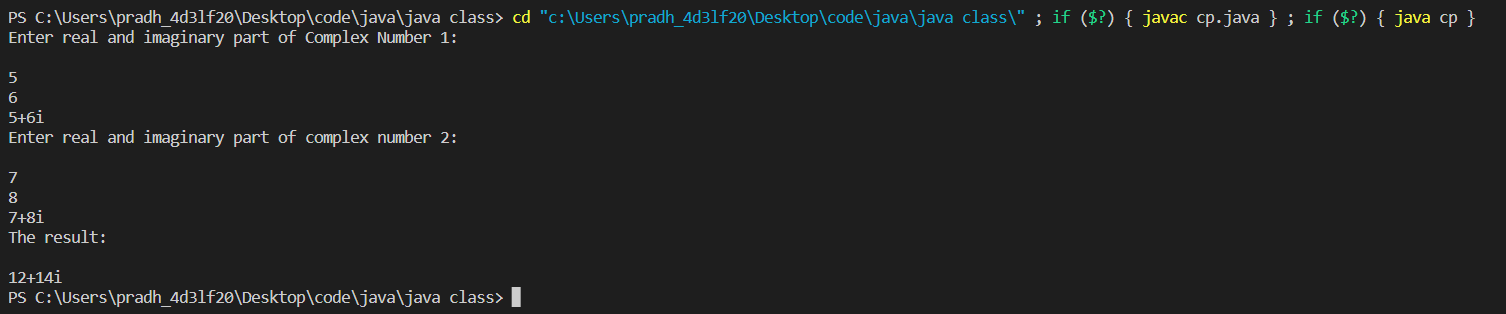
System.out.println("The result:\n");

obj3.display();

}

}

**Output:**



**Q5. WAP in java to create a triangle class and calculate the perimeter and area of all three types of triangles(Isoscealer,Equilateral,scaler).**

**Source code:**

import java.util.\*;

import java.lang.\*;

class triangle

{

double a,b,c;

double area,peri;

void input(double x,double y,double z)

{

a=x;b=y;c=z;}

void area()

{

if(a==c&&b==c)

area=(Math.sqrt(3)/4)\*a\*a;

else if(a==b&&b!=c)

area=(1.0/4)\*c\*(Math.sqrt((4\*a\*a)-(c\*c)));

else if(b==c&&c!=a)

area=(1.0/4)\*a\*(Math.sqrt((4\*b\*b)-(a\*a)));

else if(a==c&&a!=b)

area=(1.0/4)\*b\*(Math.sqrt((4\*a\*a)-(b\*b)));

else

{

double s=(a+b+c)/2.0;

area=Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));

}

}

void perimeter()

{

peri=a+b+c;

}

void display()

{

System.out.println("Area = "+area);

System.out.println("Perimeter = "+peri);

}

}

class tri

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.print("Enter the sides:\n");

double a=sc.nextDouble();

double b=sc.nextDouble();

double c=sc.nextDouble();

triangle t=new triangle();

t.input(a,b,c);

t.area();

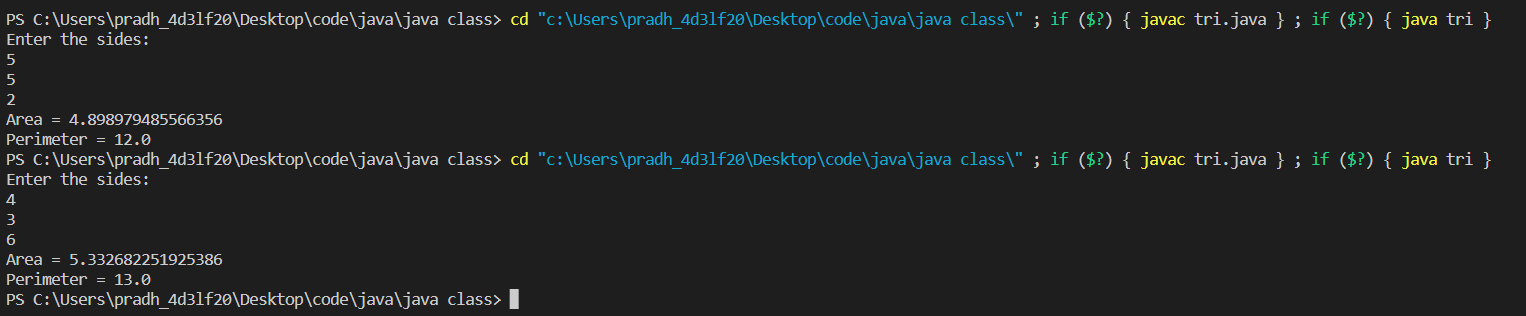
t.perimeter();

t.display();

}

}

**Output:**



**Q6. WAP in java to copy the details of one object to another.**

**Source code:**

class copy

{

int a,b;

copy()

{

a=b=1;

}

copy(int x,int y)

{

a=x;

b=y;

}

copy(copy x)

{

a=x.a;

b=x.b;

}

void display()

{

System.out.println("value of:\na= "+a+"\nb="+b);

}

public static void main(String args[])

{

copy c1=new copy();

copy c2=new copy(5,6);

copy c3=new copy(c1);

System.out.println("for 1st object:");

c1.display();

System.out.println("for 2nd object:");

c2.display();

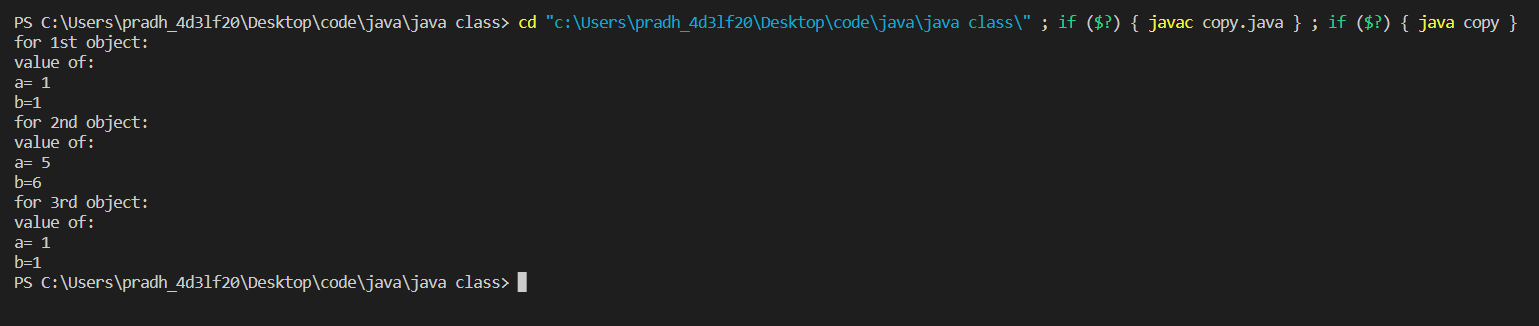
System.out.println("for 3rd object:");

c3.display();

}

}

**Output:**

****

**Q7. WAP in java to create a class and overload a method.**

**Source code:**

class sum

{

int add(int a,int b)

{

return(a+b);

}

int add(int a,int b,int c)

{

return(a+b+c);

}

double add(double a,double b)

{

return(a+b);

}

void add(int a,int b,String s)

{

System.out.println(s+""+(a+b));

}

}

class output

{

public static void main(String arg[])

{

sum s=new sum();

System.out.println(s.add(1,6));

System.out.println(s.add(7,9,2));

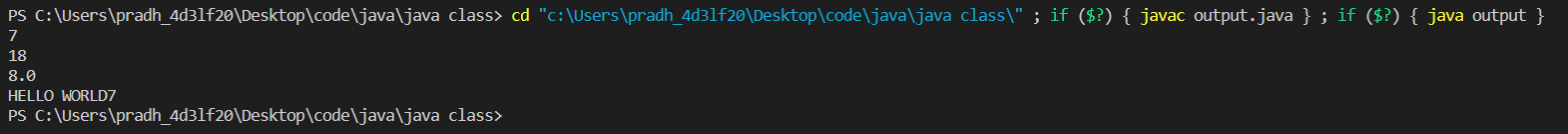
System.out.println(s.add(5.23,2.77));

s.add(3,4,"HELLO WORLD");

}

}

**Output:**



**Q8. WAP in java to create a ‘figure’ class and overload the method create() to compute the area of a square, area of a rectangle and area of a circle.**

**Source code:**

class figure

{

double create(double a)

{

return(a\*a);

}

double create(double a,double b)

{

return(a\*b);

}

double create(double r,String s)

{

return(3.14\*r\*r);

}

}

class out

{

public static void main(String args[])

{

figure f=new figure();

System.out.println("Area of Square : "+f.create(5));

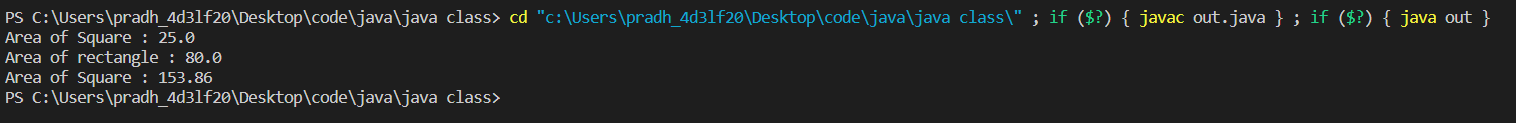
System.out.println("Area of rectangle : "+f.create(10.0,8.0));

System.out.println("Area of Square : "+f.create(7.0,"c"));

}

}

**Output:**



**Q9. WAP in Java to create complex class and add two complex numbers [Using constructor initialization].**

**Source code:**

import java.util.\*;

class Complex

{

int real,imaginary;

//static int count;

Complex()

{

real = 0;

imaginary=0;

//count++;

}

Complex(int x,int y)

{

real = x;

imaginary = y;

//count++;

}

void display()

{

System.out.println(real+"+i"+imaginary);

//System.out.println(count);

}

Complex add(Complex a, Complex b)

{

Complex sum = new Complex();

sum.real = a.real+b.real;

sum.imaginary = a.imaginary+b.imaginary;

return sum;

}

}

class test

{

public static void main(String arg[])

{

Scanner sc= new Scanner(System.in);

System.out.println("Enter 4 number ");

int input1= sc.nextInt();

int input2= sc.nextInt();

int input3= sc.nextInt();

int input4= sc.nextInt();

Complex c1 = new Complex(input1,input2);

Complex c2= new Complex(input3,input4);

Complex c3 = new Complex();

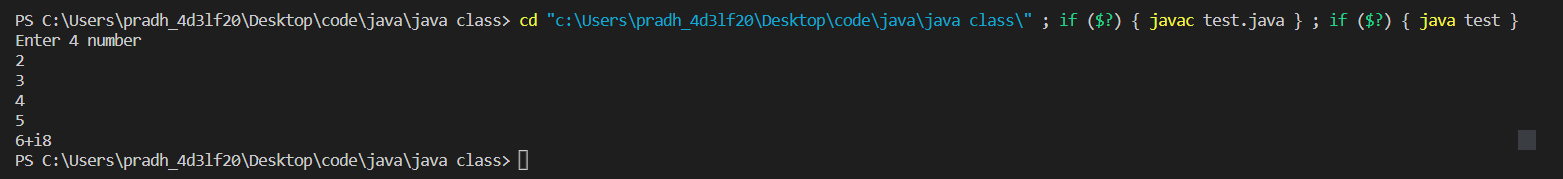
c3=c3.add(c1,c2);

c3.display();

}

}

**Output:**



**Q10. WAP in Java to overload methods to display different types of triangle by different inputs .**

**Source code:**

import java.util.\*;

class tria

{

int side1;

int side2;

int side3;

tria(int x, int y, int z)

{

side1 = x;

side2 = y;

side3 = z;

System.out.println("Triangle is Scaler, and sides are " + side1 + "," + side2 +"and" + side3);

}

tria(int x, int y)

{

side1 = side2 = x;

side3 = y;

System.out.println("Triangle is Isosceles, and sides are " + side1 + "," +side2

+ " and " + side3);

}

tria(int x)

{

side1 = side2 = side3 = x;

System.out.println("Triangle is Equilateral, and sides are " + side1 + "," +

side2 + " and " + side3);

}

}

class Tree

{

public static void main(String args[])

{

Scanner sc= new Scanner(System.in);

System.out.println("Enter 3 sides ");

int input1= sc.nextInt();

int input2= sc.nextInt();

int input3= sc.nextInt();

tria t1 = new tria(input1);

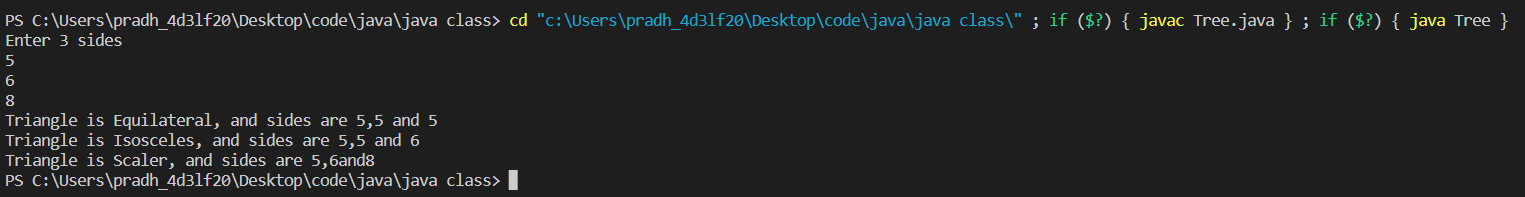
tria t2 = new tria(input1, input2);

tria t3 = new tria(input1, input2, input3);

}

}

**Output:**



**Q11. WAP in Java to print addition of two Strings,two whole numbers and two imaginary numbers.**

**Source code:**

import java.util.\*;

class sun

{

int real;

int imaginary;

void input(int x, int y)

{

real = x;

imaginary = y;

}

void display()

{

System.out.println("Complex = "+real+"+i"+imaginary);

}

sun add(sun a, sun b)

{

sun ans = new sun();

ans.real = a.real+b.real;

ans.imaginary = a.imaginary+b.imaginary;

return ans;

}

void add(int a, int b)

{

System.out.println("Sum of two no. = "+(a+b));

}

void add(String m, String n)

{

System.out.println("Sum of two string = "+m+n);

}

}

class sumoo

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println(“Enter 4 no:”);

int inp1 =sc.nextInt();

int inp2 =sc.nextInt();

int inp3 =sc.nextInt();

int inp4 =sc.nextInt();

sun c1 = new sun();

sun c2 = new sun();

sun c3 = new sun();

c1.input(inp1,inp2);

c2.input(inp3,inp4);

c3=c3.add(c1,c2);

c3.display();

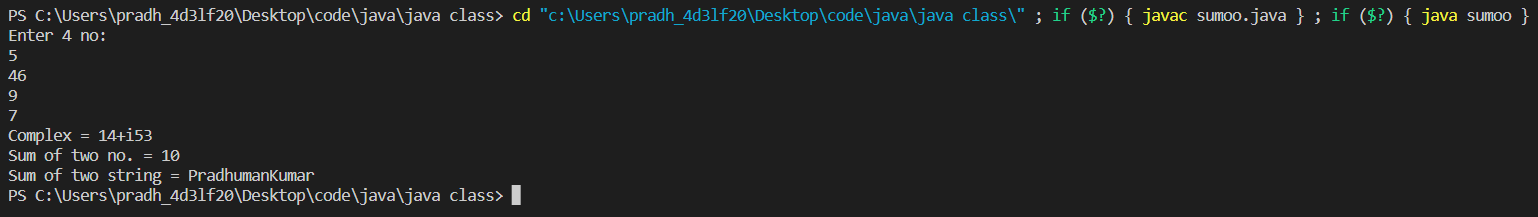
c3.add(4,6);

c3.add("Abir","Nath");

}

}

**Output:**



**Q12. WAP in java to add imaginary number using constructor and multiple objects**

**Source code:**

import java.util.\*;

class Complex

{

int real,imaginary;

//static int count;

Complex()

{

real=0;

imaginary=0;

}

Complex(int x,int y)

{

real=x;

imaginary=y;

//count++;

}

Complex(Complex a)

{

real = a.real;

imaginary = a.imaginary;

}

void display()

{

System.out.println("SUM ="+real+"+i"+imaginary);

//System.out.println(count);

}

Complex add(Complex a, Complex b)

{

Complex sum = new Complex();

sum.real = a.real+b.real;

sum.imaginary = a.imaginary+b.imaginary;

return sum;

}

}

class Testioe2

{

public static void main(String arg[])

{

Scanner sc= new Scanner(System.in);

System.out.println("Enter 4 number ");

int input1= sc.nextInt();

int input2= sc.nextInt();

int input3= sc.nextInt();

int input4= sc.nextInt();

Complex c1 = new Complex(input1,input2);

Complex c2= new Complex(input3,input4);

Complex c3 = new Complex(c1);

Complex c4 = new Complex();

c1.display();

c2.display();

c3.display();

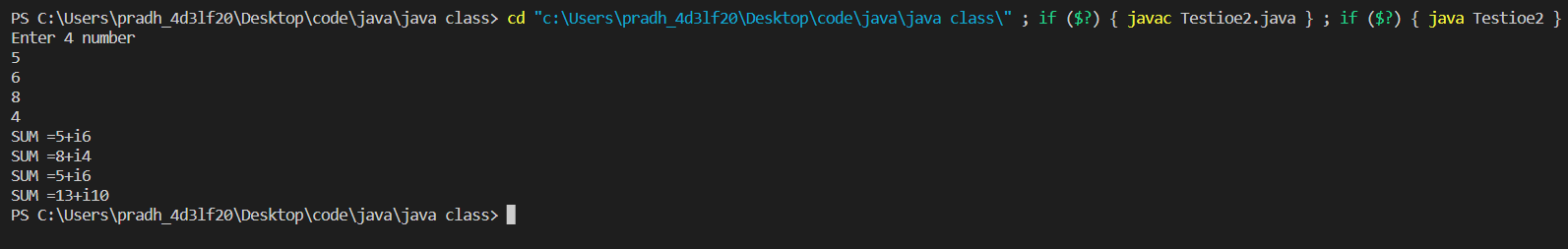
c4=c4.add(c1,c2);

c4.display();

}

}

**Output:**



**Q13. WAP in java to create a room class and room area class where room area extend room class.**

**Source code:**

import java.util.\*;

import java.lang.\*;

class Room

{

double l,b;

Room()

{

l=0.0;

b=0.0;

}

Room(double x,double y)

{

l=x;

b=y;

}

}

class RoomArea extends Room

{

RoomArea(double x,double y)

{

super(x,y);

}

double countTiles(double a,double b)

{

System.out.println("room Area= "+(super.l\*super.b));

return (Math.ceil((super.l\*super.b)/(a\*b)));

}

}

class result

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the Length and Breadth of Room");

int a=sc.nextInt();

int b=sc.nextInt();

System.out.println("Enter the Length and Breadth of Tiles");

int c=sc.nextInt();

int d=sc.nextInt();

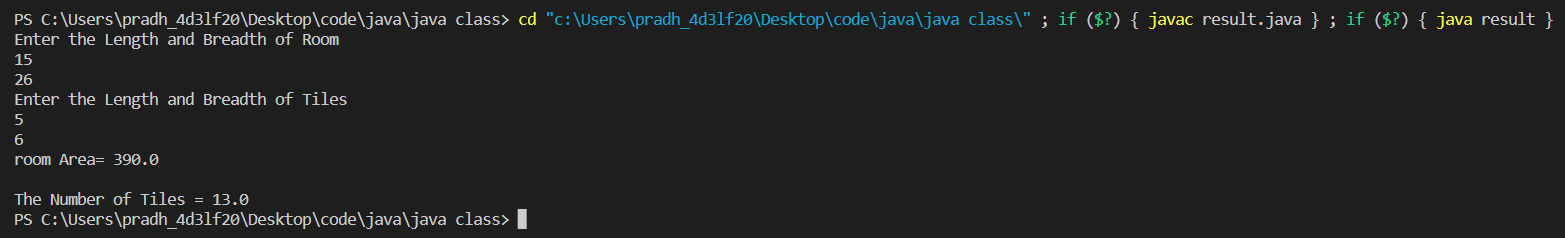
RoomArea r=new RoomArea(a,b);

System.out.println("\nThe Number of Tiles = "+r.countTiles(c,d));

}

}

**Output:**

****

**Q14. WAP in java to create a queue and do insert() and delete() in queue.**

**Source code:**

**class queue{**

**int array[];**

**int front;**

**int rear;**

**queue(int size){**

**array = new int [size];**

**front = -1;**

**rear = -1;**

**}**

**void insert(int item)**

**{**

**if((front == 0)&&(rear == array.length -1)){**

**System.out.println("queue is full");**

**}**

**else**

**{**

**if(front<0)**

**front = 0;**

**rear++;**

**array[rear] = item;**

**}**

**}**

**int delete()**

**{**

**int element ;**

**if(front<0)**

**{**

**System.out.println("queue is empty");**

**return 0;**

**}**

**else{**

**element = array[front];**

**if(front>=rear)**

**{**

**front = -1;**

**rear = -1;**

**}**

**else{**

**front++;**

**}**

**return element;**

**}**

**}**

**}**

**class testqueue{**

**public static void main(String args[])**

**{**

**queue q1 = new queue(5);**

**queue q2 = new queue(10);**

**q1.insert(32);**

**q1.insert(44);**

**q1.insert(21);**

**q1.insert(55);**

**q1.insert(12);**

**for( int i=0;i<8;i++){**

**q2.insert(i);**

**}**

**System.out.println(q1.delete());**

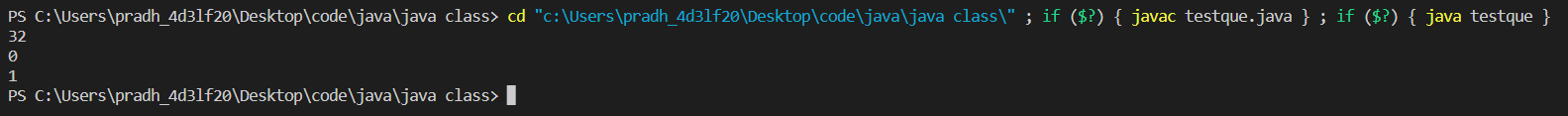
**System.out.println(q2.delete());**

**System.out.println(q2.delete());**

**}**

**}**

**Output:**

****

**Q15. WAP in java to create a stack and use push() and pop() function in stack.**

**Source code:**

class Stack{

int array[];

int top;

Stack(int size){

array = new int [size];

top = -1;

}

void push(int item)

{

if(top>array.length-1){

System.out.println("STACK is full");

}

else

{

top++;

array[top] = item;

}

}

int pop()

{

if(top<0)

{

System.out.println("STACK is empty");

return 0;

}

else{

return array[top--];

//top--;

}

}

}

class testStack{

public static void main(String args[])

{

Stack s1 = new Stack(5);

Stack s2 = new Stack(10);

s1.push(32);

s1.push(44);

s1.push(21);

s1.push(55);

s1.push(12);

for( int i=0;i<8;i++){

s2.push(i);

}

System.out.println(s1.pop());

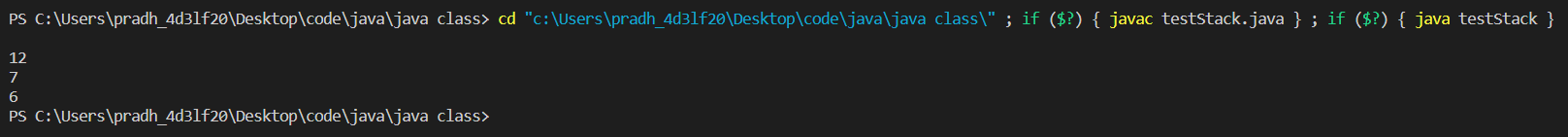
System.out.println(s2.pop());

System.out.println(s2.pop());

}

}

**Output:**



**Q16. Write a program in java to create a number class, a binary class and an octal class where both binary and octal classes extends number class. Write the method conversion() in both of the subclasses to convert the given decimal no. Into binary and octal representation.**

**Source code:**

import java.util.\*;

class Number

{

int n;

Number()

{

n=0;

}

Number(int x)

{

n=x;

}

}

class Binary extends Number

{

Binary()

{

super();

}

Binary(int x)

{

super(x);

}

void conversion()

{

int n=super.n;

if(n==0)

{

System.out.println("0");

return;

}

int i=0;

String s[]=new String[10];

while(n>0)

{

s[i]=Integer.toString(n%2);

n/=2;

i++;

}

System.out.print("\nThe Binary Equivalent:\t");

for(int j=i-1;j>=0;j--)

{

System.out.print(s[j]);

}

}

}

class Octal extends Number

{

Octal()

{

super();

}

Octal(int x)

{

super(x);

}

void conversion()

{

int n=super.n;

if(n==0)

{

System.out.println("0");

return;

}

int i=0;

String s[]=new String[10];

while(n>0)

{

s[i]=Integer.toString(n%8);

n/=8;

i++;

}

System.out.print("\n\nThe Octal Equivalent:\t");

for(int j=i-1;j>=0;j--)

{

System.out.print(s[j]);

}

}

}

class convo

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.print("Enter the Number:\t");

int a=sc.nextInt();

Binary b=new Binary(a);

Octal k=new Octal(a);

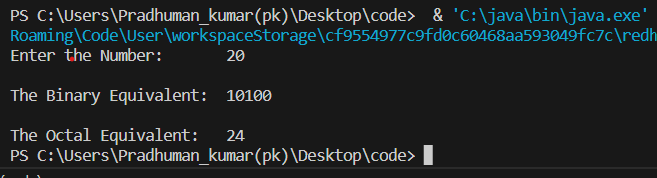
b.conversion();

k.conversion();

}

}

**Output:**



**Q17. WAP in java to implement inheritance.**

**Source code:**

class A

{

private int a;

protected int b;

public int c;

A()

{

a=10;

b=11;

c=12;

}

A(int x)

{

x=a;

}

A(int y , int z)

{

y=b;

z=c;

}

void display()

{

System.out.println(a+","+b+","+c);

}

}

class B extends A

{

int b;

B()

{

super();

b=5;

}

B(int i)

{

super.b=i;

}

B(int k , int l, int m)

{

super(k,l);

m=b;

}

}

class Main

{

public static void main(String[] args)

{

A a1 = new A();

a1.display();

System.out.println(a1.c);

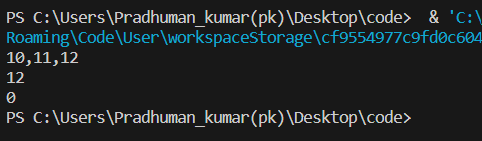
B b1 = new B(5);

System.out.println(b1.b);

}

}

**Output:**



**Q18. WAP in JAVA to create two packages p1 and p2. Create a class prime and method “checkPrime(int x)” in p1.Call the method “checkPrime(int x)” in package p2 to print all the Prime numbers within range 10 to 90.**

**Source code:-**

***In package p1:***

package p1;

public class prime

{

public static boolean checkPrime(int x)

{

for(int i = 2; i <= x/i; ++i)

{

if(x % i == 0)

{

return false;

}

}

return true;

}

}

***In package p2:***

package p2;

class main

{

public static void main(String args[])

{

int min = 10;

int max = 90;

System.out.println("Prime numbers between 10 to 90:");

for(int x=min;x<=max;x++)

{

if(p1.prime.checkPrime(x))

{

System.out.print(x+" ");

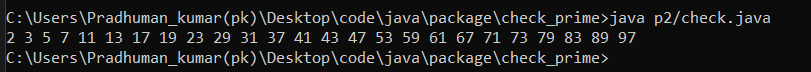
}

}

}

}

Output:-



**Q19. WAP in JAVA to create two packages p1 and p2. In p1 create a class ‘Time’ and write a method “def Time(Time t1, Time t2)” to compute the difference between two Times t1 & t2. Compute the duration of a task if the task starts at 07 hr 35 min 57 sec and ends at 10 hr 07min 09sec.**

**Source code:-**

***In package p1:-***

package p1;

public class time

{

int h,m,s;

public time(int hours, int minutes, int seconds)

{

h = hours;

m = minutes;

s = seconds;

}

public int differenceTime(time t1, time t2)

{

int ts1 = t1.h \* 3600 + t1.m \* 60 + t1.s;

int ts2 = t2.h \* 3600 + t2.m \* 60 + t2.s;

return Math.abs(ts2 - ts1);

}

}

***In package p2:-***

package p2;

public class compute

{

public static void main(String[] args)

{

p1.time startTime = new p1.time(7, 35, 45);

p1.time endTime = new p1.time(11, 5, 7);

int timeDuration = endTime.differenceTime(startTime, endTime);

System.out.println("Time duration: " + timeDuration + " seconds");

int n=timeDuration;

int a=n/3600;

int b=n%3600;

int c=b/60;

int d=b%60;

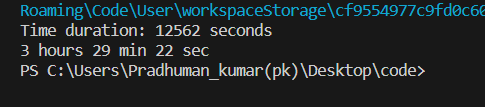
int e=d%60;

System.out.println(a+" hours "+c+" min "+e+" sec ");

}

}

**Output:-**



**Q20.WAP in JAVA to create two thread which will print all the even number between 10 to 100 and other thread will print all the odd numbers between the range 10 to 100.**

**SOURCE CODE:-**

class A extends Thread

{

public void run()

{

try

{

Thread.sleep(10);

for(int i = 0;i<=100;i++)

{

if(i%2==0)

{

System.out.print("Even = "+i+" ");

}

}

}

catch(InterruptedException e)

{

System.out.println("Child Thread Is Interrupted");

}

System.out.println("\n");

}

}

class B

{

public static void main(String args[])

{

A even = new A();

even.start();

try

{

Thread.sleep(1000);

for(int i = 0;i<=100;i++)

{

if(i%2!=0)

{

System.out.print("Odd ="+i+" ");

}

}

}

catch(InterruptedException e)

{

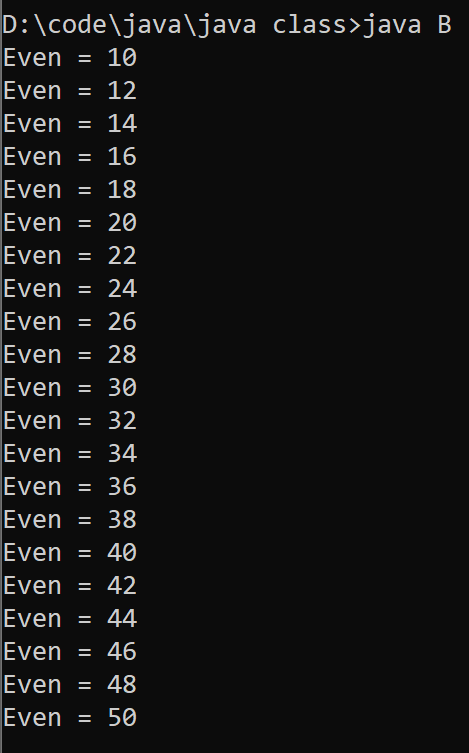
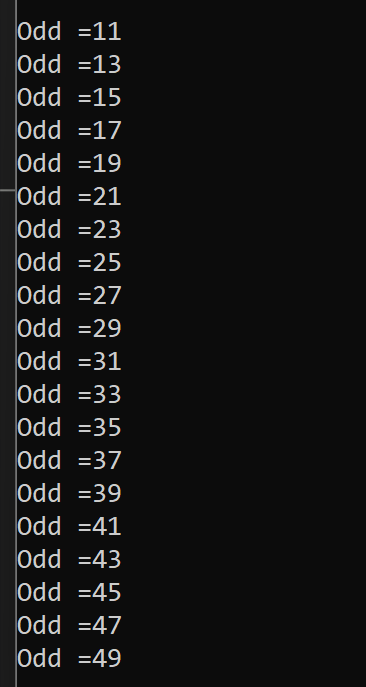
System.out.println("Main Thread Is Interrupted");

}

}

}

**OUTPUT:-**

**Q21.WAP in JAVA to create two thread which will print all multiple of 5 within the range 10 to 100 and other thread will print all multiple of 7 between the range 10 to 100.check the priority of the threads and also re-execute the threads by changing their priority.**

**SOURCE CODE:-**

class A1 extends Thread

{

public void run()

{

try

{

Thread.sleep(10);

for(int i = 0;i<=100;i++)

{

if(i%5==0)

{

System.out.print("Multiple of 5 = "+i+" ");

}

}

}

catch(InterruptedException e)

{

System.out.println("Child Thread Is Interrupted");

}

System.out.println("\n");

}

}

class B2

{

public static void main(String args[])

{

A1 mul = new A1();

mul.start();

mul.setPriority(6);

try

{

Thread.sleep(1000);

System.out.println(mul.getPriority());

Thread.currentThread().setPriority(10);

System.out.println(Thread.currentThread().getPriority());

System.out.println("\n");

for(int i = 0;i<=100;i++)

{

if(i%7==0)

{

System.out.print("Multiple of 7 ="+i+" ");

}

}

}

catch(InterruptedException e)

{

System.out.println("Main Thread Is Interrupted");

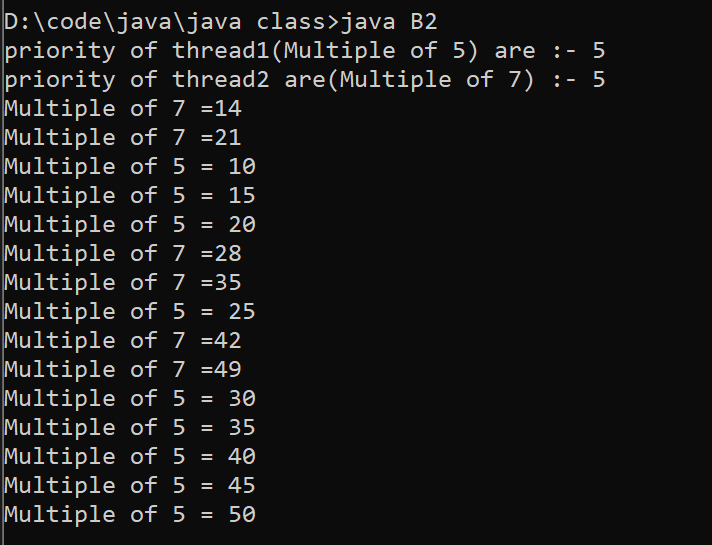
}

}

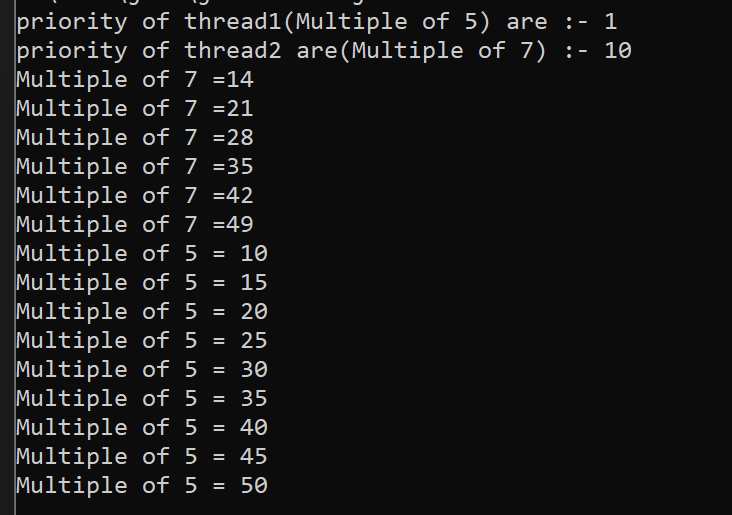
}

**OUTPUT:-**

**(without set priority)**



**(with set priority)**

****

**Q22.Implement producer consumer problem using JAVA so that both the threads can be synchronized.**

**SOURCE CODE:-**

**// with wait() and notify() method**

**class Q**

**{**

**int x;**

**boolean value = false;**

**synchronized int get()**

**{**

**if(!value)**

**{**

**try**

**{**

**wait();**

**}**

**catch(Exception e)**

**{**

**System.out.println(e);**

**}**

**}**

**System.out.println("consumed: "+ x);**

**value=false;**

**notify();**

**return x;**

**}**

**synchronized void put(int a)**

**{**

**if(value)**

**{**

**try**

**{**

**wait();**

**}**

**catch(Exception e)**

**{**

**System.out.println(e);**

**}**

**}**

**x=a;**

**System.out.println("Produced: "+ x);**

**value = true;**

**notify();**

**}**

**}**

**class Producer implements Runnable**

**{**

**Q q;**

**Thread t;**

**Producer(Q q1)**

**{**

**q = q1;**

**t = new Thread(this, "Producer");**

**t.start();**

**}**

**public void run()**

**{**

**int i=10;**

**while(i<15)**

**{**

**q.put(i++);**

**}**

**}**

**}**

**class Consumer implements Runnable**

**{**

**Q q;**

**Thread t;**

**Consumer(Q q1)**

**{**

**q = q1;**

**t = new Thread(this, "Consumer");**

**t.start();**

**}**

**public void run()**

**{**

**int i=10;**

**while(i<15)**

**{**

**q.get();**

**i++;**

**}**

**}**

**}**

**class syncroniz1**

**{**

**public static void main(String arg[])**

**{**

**Q q = new Q();**

**new Producer(q);**

**new Consumer(q);**

**}**

**}**

**// without wait() and notify() method**

**class Q extends Thread**

**{**

**int n;**

**synchronized int get()**

**{**

**try**

**{**

**Thread.sleep(500);**

**}**

**catch(Exception e)**

**{**

**System.out.println(e);**

**}**

**return n;**

**}**

**synchronized void put(int x)**

**{**

**try**

**{**

**n = x;**

**System.out.println( "Item is produced"+n);**

**Thread.sleep(500);**

**}**

**catch(Exception e)**

**{**

**System.out.println(e);**

**}**

**}**

**}**

**class Producer implements Runnable**

**{**

**Q q;**

**Thread t;**

**Producer(Q q)**

**{**

**this.q = q;**

**t = new Thread(this, "Producer");**

**t.start();**

**}**

**public void run()**

**{**

**int i=10;**

**while(i<15)**

**{**

**q.put(i++);**

**}**

**}**

**}**

**class Consumer implements Runnable**

**{**

**Q q;**

**Thread t;**

**Consumer(Q q)**

**{**

**this.q = q;**

**t = new Thread(this, "Consumer");**

**t.start();**

**}**

**public void run()**

**{**

**while(true)**

**{**

**System.out.println("consumer: "+q.get());**

**}**

**}**

**}**

**class PC**

**{**

**public static void main(String arg[])**

**{**

**Q q = new Q();**

**System.out.println("Press Ctrl+c to stop");**

**new Producer(q);**

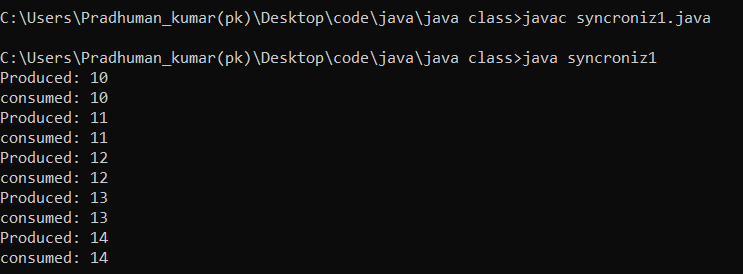
**new Consumer(q);**

**}**

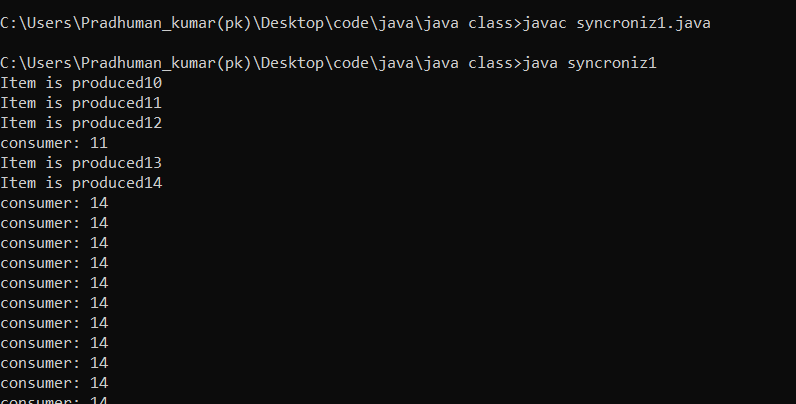
**}**

**OUTPUT:-**

With wait() and notify()



Without wait() and notify()



**Q23. Implement producer consumer problem using JAVA so that both the threads can be synchronized. In which producer produce a number in integer format (19) but consumer got number in words format (nineteen).**

**SOURCE CODE:-**

**import java.util.Dictionary;**

**import java.util.Hashtable;**

**class synchroniz2**

**{**

**public static void main(String[] args)**

**{**

**Q q=new Q();**

**new producer(q);**

**new consumer(q);**

**}**

**}**

**class Q**

**{**

**int n;**

**boolean value=false;**

**synchronized void put(int a)**

**{**

**if(value)**

**{**

**try**

**{**

**wait();**

**}**

**catch(Exception e)**

**{**

**System.out.println(e);**

**}**

**}**

**n=a;**

**System.out.println("Produced: "+ n);**

**value = true;**

**notify();**

**}**

**synchronized int get()**

**{**

**if(!value)**

**{**

**try**

**{**

**wait();**

**}**

**catch(Exception e)**

**{**

**System.out.println(e);**

**}**

**}**

**Dictionary<Integer, String> no= new Hashtable<>();**

**no.put(0,"zero");**

**no.put(1,"one");**

**no.put(2,"two");**

**no.put(3,"three");**

**no.put(4,"four");**

**no.put(5,"five");**

**no.put(6,"six");**

**no.put(7,"seven");**

**no.put(8,"eight");**

**no.put(9,"nine");**

**no.put(10,"ten");**

**no.put(11,"eleven");**

**no.put(12,"twelve");**

**no.put(13,"thirteen");**

**no.put(14,"fourteen");**

**no.put(15,"fifteen");**

**no.put(16,"sixteen");**

**no.put(17,"seventeen");**

**no.put(18,"eighteen");**

**no.put(19,"nineteen");**

**no.put(20,"twenty");**

**no.put(30,"thirty");**

**no.put(40,"fourty");**

**no.put(50,"fifty");**

**no.put(60,"sixty");**

**no.put(70,"seventy");**

**no.put(80,"eighty");**

**no.put(90,"ninety");**

**no.put(100,"hundred");**

**if (n<=20 | n==100)**

**{**

**System.out.println("consumed "+no.get(n));**

**}**

**else if(n<100)**

**{**

**int m=n%10;**

**int o=n-m;**

**if(m==0)**

**{**

**System.out.println("consumed "+no.get(o));**

**}**

**else**

**{**

**System.out.println("consumed "+no.get(o)+" "+no.get(m));**

**}**

**}**

**value=false;**

**notify();**

**return n;**

**}**

**}**

**class producer implements Runnable**

**{**

**Q q;**

**Thread t;**

**producer(Q q1)**

**{**

**q=q1;**

**t=new Thread(this,"producer");**

**t.start();**

**}**

**public void run()**

**{**

**int i=1;**

**while(i<101)**

**{**

**q.put(i++);**

**}**

**}**

**}**

**class consumer implements Runnable**

**{**

**Q q;**

**Thread t;**

**consumer(Q q1)**

**{**

**q=q1;**

**t=new Thread(this,"consumer");**

**t.start();**

**}**

**public void run()**

**{**

**int i=1;**

**while(i<101)**

**{**

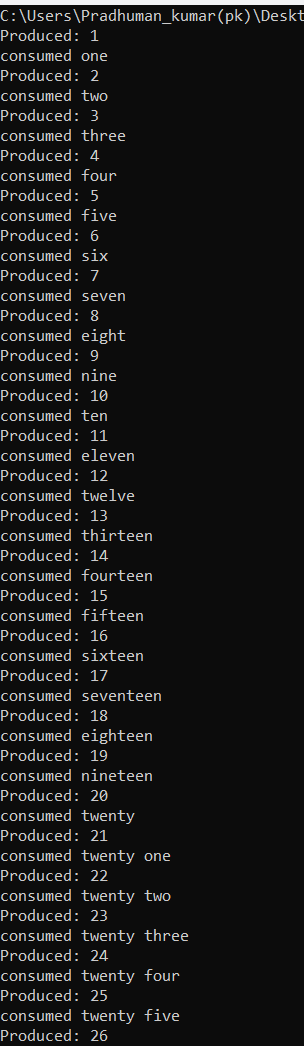
**q.get();**

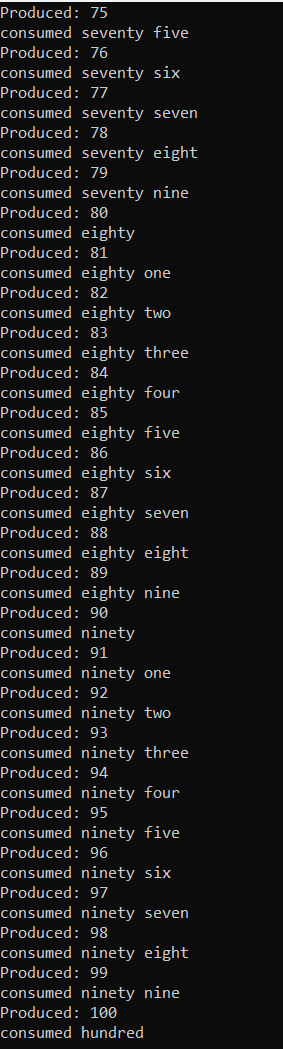
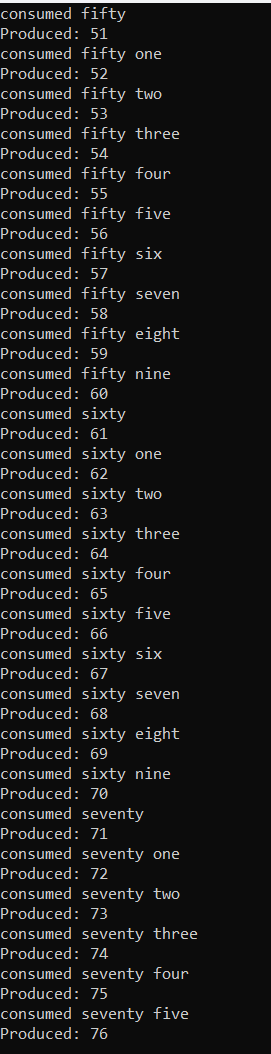
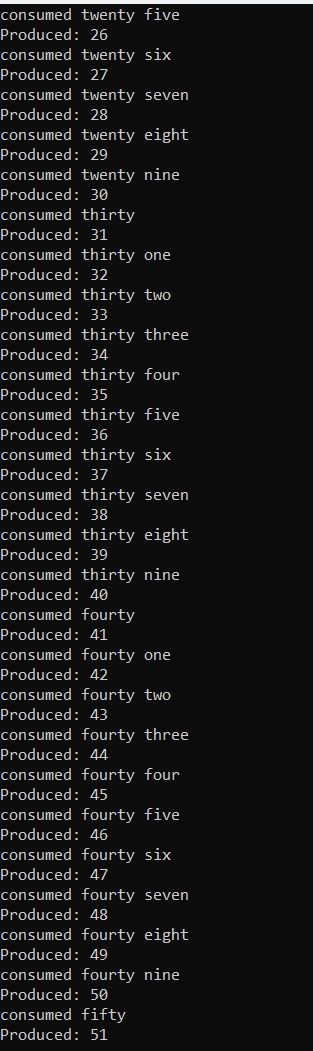
**i++;**

**}**

**}**

**}**

**OUTPUT:-**



**Q24. Implement producer consumer problem using JAVA so that both the threads can be synchronized. In which producer produce a date in integer format (10-5-2010) but consumer got the date in word format (ten-may-two thousand ten).**

**SOURCE CODE:-**

**import java.util.Dictionary;**

**import java.util.Hashtable;**

**class synchroniz3**

**{**

**public static void main(String[] args)**

**{**

**Q q=new Q();**

**new producer(q);**

**new consumer(q);**

**}**

**}**

**class Q**

**{**

**int m,n,o;**

**boolean value=false;**

**synchronized void put(int x,int y,int z)**

**{**

**if(value)**

**{**

**try**

**{**

**wait();**

**}**

**catch(Exception e)**

**{**

**System.out.println(e);**

**}**

**}**

**m=x;**

**n=y;**

**o=z;**

**System.out.println("produced "+m+"-"+n+"-"+o);**

**value = true;**

**notify();**

**}**

**synchronized int get()**

**{**

**if(!value)**

**{**

**try**

**{**

**wait();**

**}**

**catch(Exception e)**

**{**

**System.out.println(e);**

**}**

**}**

**Dictionary<Integer, String> date= new Hashtable<>();**

**date.put(1,"one");**

**date.put(2,"two");**

**date.put(3,"three");**

**date.put(4,"four");**

**date.put(5,"five");**

**date.put(6,"six");**

**date.put(7,"seven");**

**date.put(8,"eight");**

**date.put(9,"nine");**

**date.put(10,"ten");**

**date.put(11,"eleven");**

**date.put(12,"twelve");**

**date.put(13,"thirteen");**

**date.put(14,"fourteen");**

**date.put(15,"fifteen");**

**date.put(16,"sixteen");**

**date.put(17,"seventeen");**

**date.put(18,"eighteen");**

**date.put(19,"nineteen");**

**date.put(20,"twenty");**

**date.put(30,"thirty");**

**Dictionary<Integer, String> week= new Hashtable<>();**

**week.put(1,"january");**

**week.put(2,"february");**

**week.put(3,"march");**

**week.put(4,"april");**

**week.put(5,"may");**

**week.put(6,"jun");**

**week.put(7,"july");**

**week.put(8,"august");**

**week.put(9,"september");**

**week.put(10,"october");**

**week.put(11,"november");**

**week.put(12,"december");**

**if (m<=20)**

**{**

**int d=o/1000;**

**int c=o%1000;**

**System.out.println("consumed " + date.get(m) + " / " + week.get(n) + " / " + date.get(d) + " thousand " + date.get(c));**

**}**

**else if(m<31)**

**{**

**int d=o/1000;**

**int c=o%1000;**

**int a=m%10;**

**int b=m-a;**

**if(a==0)**

**{**

**System.out.println("consumed " + date.get(m) + " / " + week.get(n) + " / " + date.get(d) + " thousand " + date.get(c));**

**}**

**else**

**{**

**System.out.println("consumed " + date.get(b) + " " + date.get(a) + " / " + week.get(n) + " / " + date.get(d) + " thousand " + date.get(c));**

**}**

**}**

**value=false;**

**notify();**

**return n;**

**}**

**}**

**class producer implements Runnable**

**{**

**Q q;**

**Thread t;**

**producer(Q q1)**

**{**

**q=q1;**

**t=new Thread(this,"producer");**

**t.start();**

**}**

**public void run()**

**{**

**int i=1;**

**while(i<31)**

**{**

**int j=i%12+1;**

**int k=j+2008;**

**q.put(i++, j++, k++);**

**}**

**}**

**}**

**class consumer implements Runnable**

**{**

**Q q;**

**Thread t;**

**consumer(Q q1)**

**{**

**q=q1;**

**t=new Thread(this,"consumer");**

**t.start();**

**}**

**public void run()**

**{**

**int i=1;**

**while(i<31)**

**{**

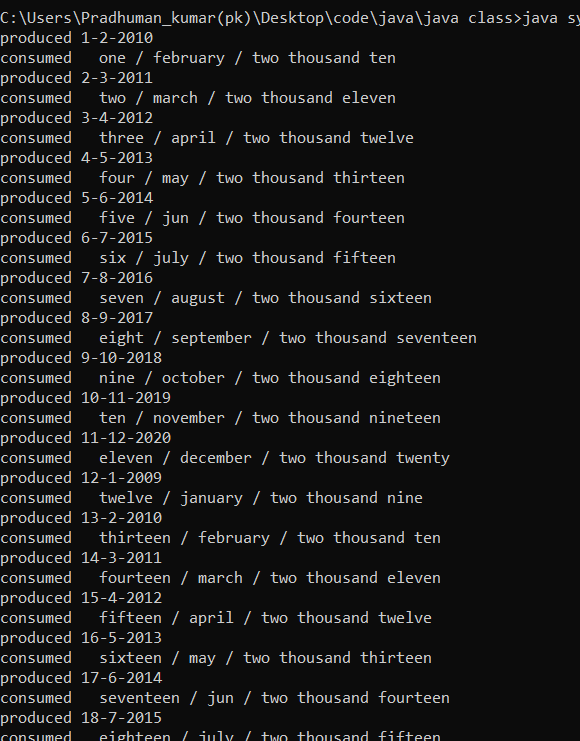
**q.get();**

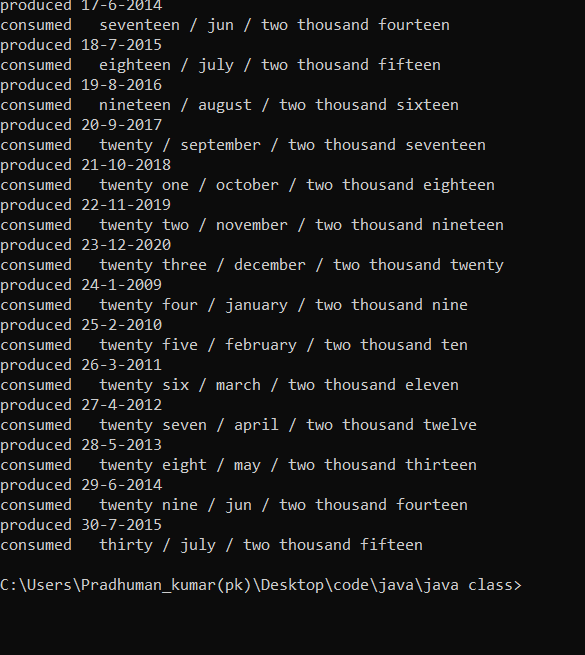
**i++;**

**}**

**}**

**}**

**OUTPUT:-**



**Q 25. WAP in java to create a package p1 and create three classes A, B and C in package p1. Also create a testpackage in package p2. Now instantiate three classes A, B and C of p1 in ‘main ()’ method of the class testpackage of package p2(Do not import p1 in p2). Any one class of A, B, C should consist of Private, Protected and Public members. List up the accessibility of these three types of members inside same and outside the package.**

**Source code:-**

**In package p1:**

**Parent class:**

package p1;

public class A {

private int a;

protected int b;

public int c;

int d;

void input(int x, int y, int z, int w) {

a = x;

b = y;

c = z;

d = w;

}

void display() {

System.out.println("a=" + a + " b=" + b + " c=" + c + " d=" + d);

}

}

**Sub-Class:**

package p1;

class C extends A {

void display() {

super.b = 10;

super.c = 11;

super.d = 12;

System.out.println("(In class C)" + " b=" + b + " c=" + c);

System.out.println("(In class C)" + " a=" + super.a);

System.out.println("(In class C)" + " d=" + d);

}

}

**Main Class:**

package p1;

class B {

public static void main(String args[]) {

A obj = new A();

obj.input(2, 3, 4, 5);

obj.display();

System.out.println("b=" + obj.b + " c=" + obj.c);

System.out.println("d=" + obj.d);

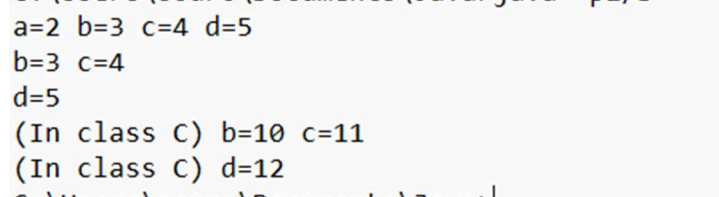
C obj1 = new C();

obj1.display();

}

}

**Output:-**



**In package p2:**

**Sub-Class**

package p2;

class other extends p1.A {

public void display() {

super.b = 10;

super.c = 11;

System.out.println("(In Sub-Class of p2 package)" + " b=" + b);

System.out.println("(In Sub-Class of p2 package)" + " c=" + c);

}

}

**In package p2:**

**Main class:**

package p2;

class testpackage {

public static void main(String args[])

{

p1.A obj2 = new p1.A();

System.out.println("parent class (Which is in p1) called in p2 package c= "+obj2.c+" (Default constructor value is 0)");

bj2.c = 13;

System.out.println("From main class after update from main class of p2 package c= "+obj2.c);

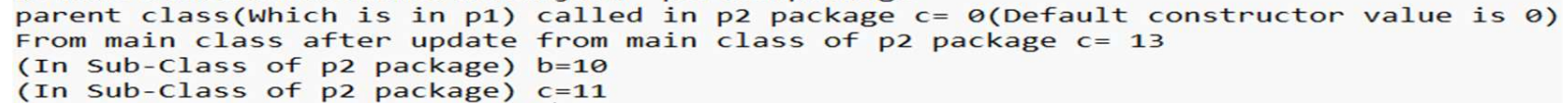
other obj3 = new other();

obj3.display();

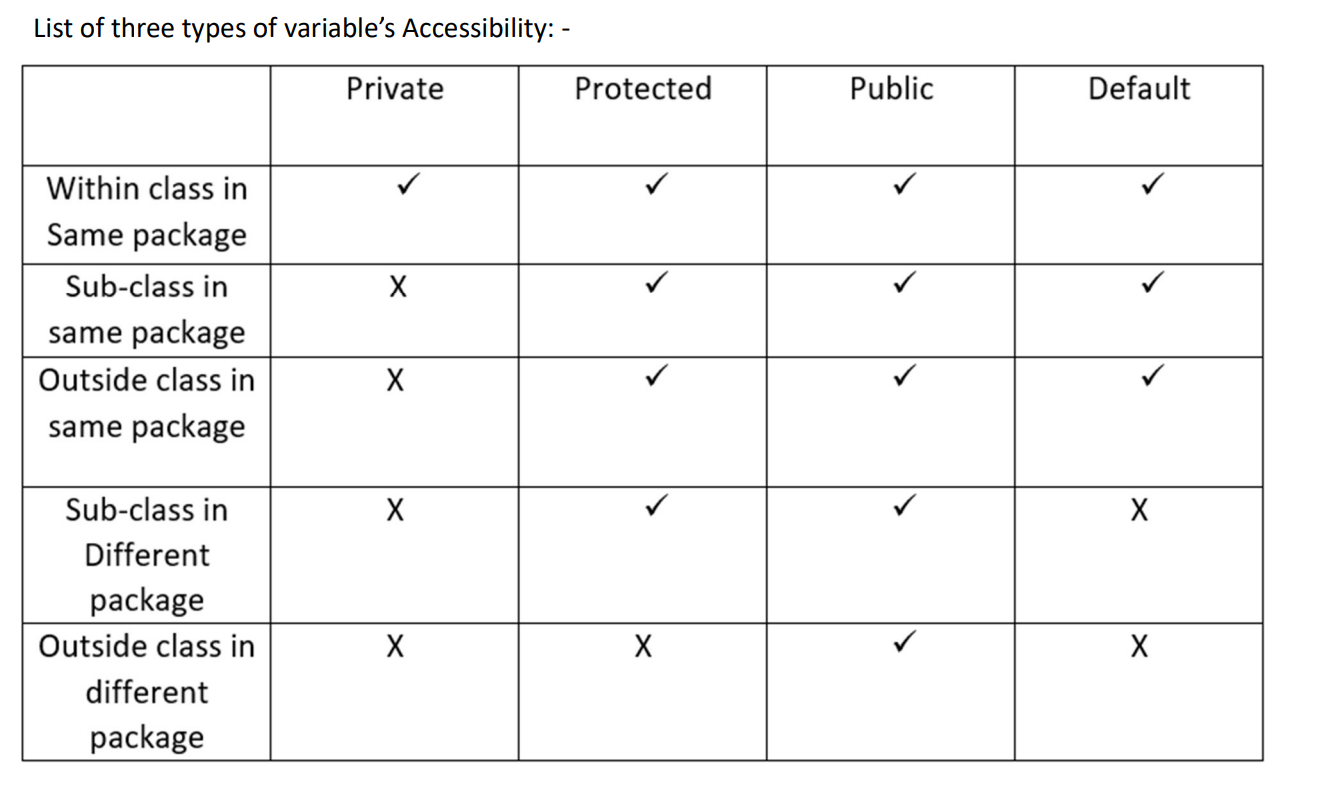
}

}

**Output:-**

****

**Conclusion:-**



**Q26.WAP in JAVA to implement a modulo-8 counter using inter-thread communication.**

**Source code:-**

class initiate

{

int a;

boolean val=true;

synchronized void pro(int x)

{

if(!val)

{

try

{

wait();

}

catch(Exception e)

{

System.out.println(e);

}

}

a=x;

System.out.println("\nClock: "+a);

val=false;

notify();

}

synchronized void con()

{

if(val)

{

try

{

wait();

}

catch(Exception e)

{

System.out.println(e);

}

}

System.out.println("\nCounter: "+bin(a));

val=true;

notify();

}

String bin(int a)

{

if(a%9==0)

return "000";

else if(a>9)

a=a%9;

else

{

if(a-1==0)

return "000";

else

a=a-1;

}

String s=" ";

while(a>0)

{

s= String.valueOf(a%2)+s;

a=a/2;

}

return s;

}

}

class producer implements Runnable

{

initiate p;

Thread t;

producer(initiate k)

{

p=k;

t = new Thread(this);

t.start();

}

public void run()

{

for(int i=1;i<=15;i++)

p.pro(i);

}

}

class consumer implements Runnable

{

initiate p;

Thread t;

consumer(initiate k)

{

p=k;

t = new Thread(this);

t.start();

}

public void run()

{

for(int i=1;i<=15;i++)

p.con();

}

}

public class manage

{

public static void main(String[] args)

{

initiate i = new initiate();

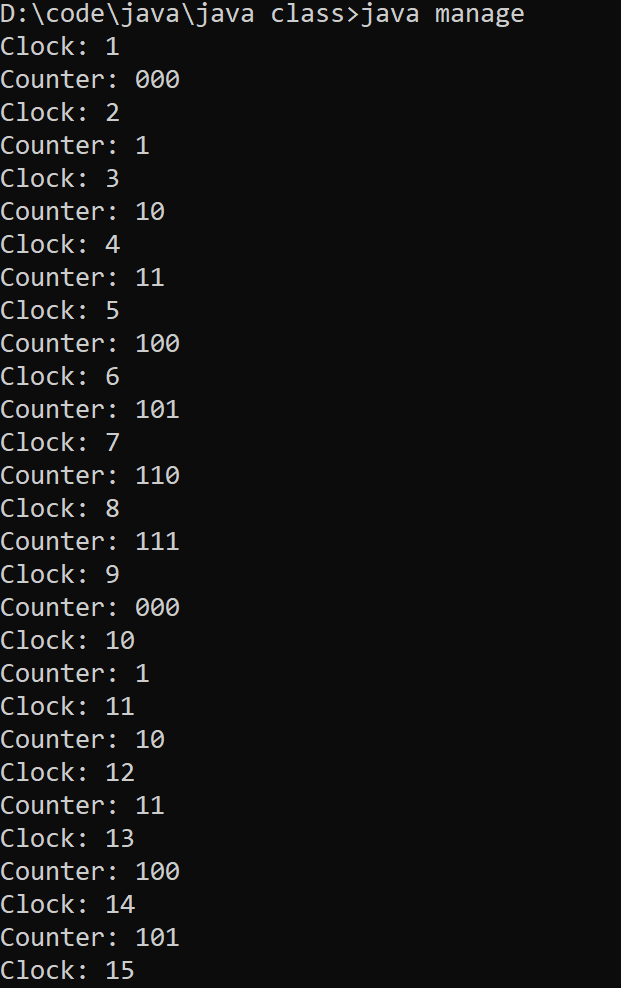
new producer(i);

new consumer(i);

}

}

**Output:-**

****