

EXERCISE-1

(BASICS)

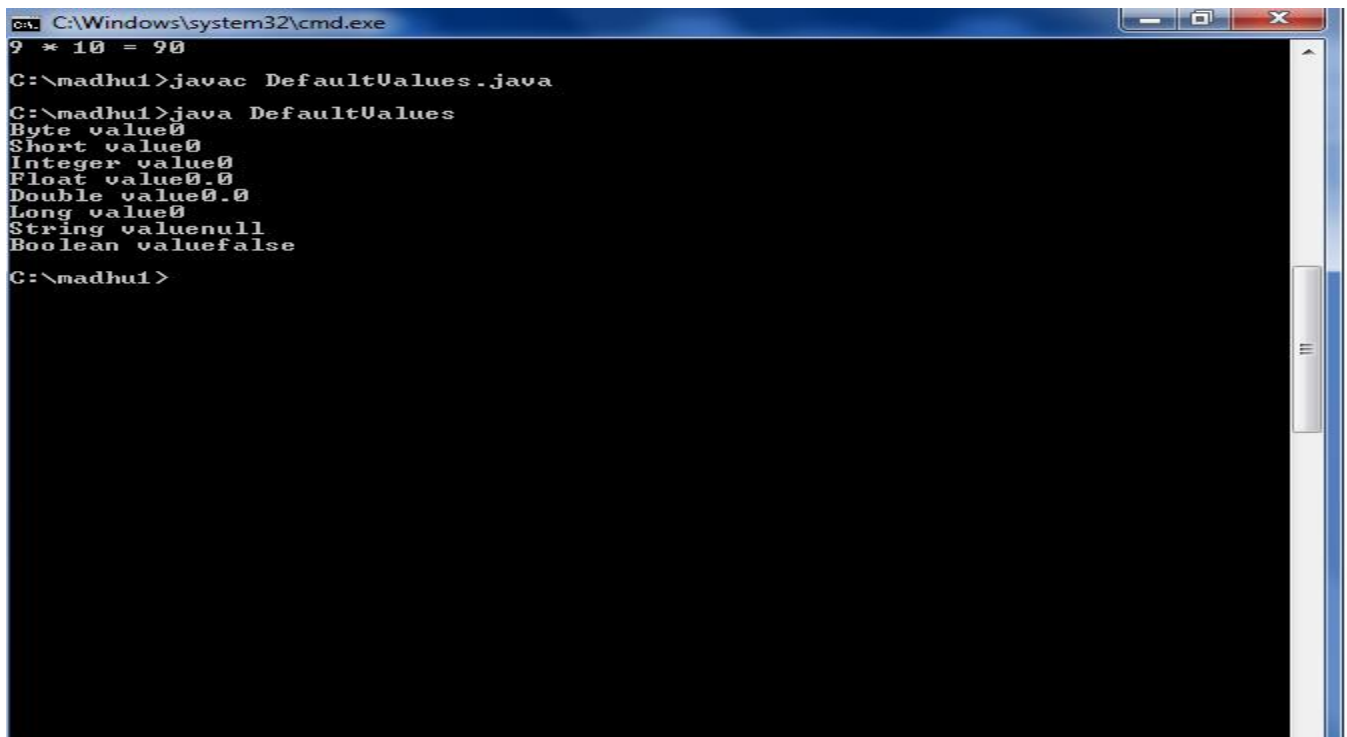
1. Write a java program to display default values of all primitive datatypes of java.

Description: primitive datatypes are byte, short, int, float, double, long, string, boolean etc.

Source Code:

```
public class DefaultValues{
byte val1;
short val2;
int val3;
float val4;
double val5;
long val6;
String val7;
boolean val8;
public static void main(String args[]){
DefaultValues df=new DefaultValues();
System.out.println("Byte value"+df.val1);
System.out.println("Short value"+df.val2);
System.out.println("Integer value"+df.val3);
System.out.println("Float value"+df.val4);
System.out.println("Double value"+df.val5);
System.out.println("Long value"+df.val6);
System.out.println("String value"+df.val7);
System.out.println("Boolean value"+df.val8);
}}
```

Output:



```
C:\Windows\system32\cmd.exe
9 * 10 = 90

C:\madhu1>javac DefaultValues.java

C:\madhu1>java DefaultValues
Byte value0
Short value0
Integer value0
Float value0.0
Double value0.0
Long value0
String valuenull
Boolean valuefalse

C:\madhu1>
```

2. Write a java program that displays the root of a quadratic equation

$ax^2+bx+c=0$. calculate the discriminate D and basing on value of D, describe the nature of roots(Finding root values).

Description:

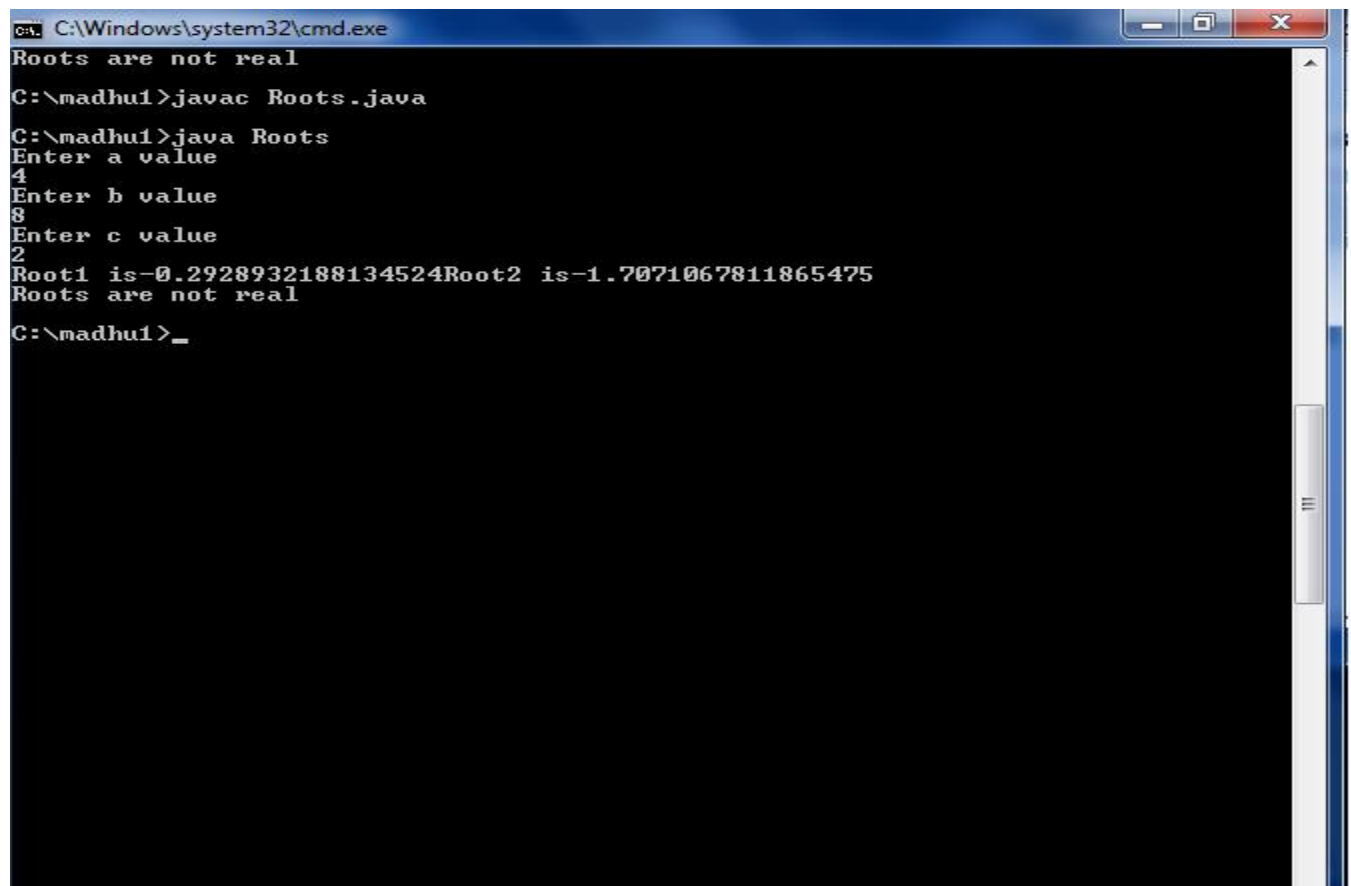
Calculate the roots. Based on the roots, if the root1 is greater than 0 then check the next condition that if the root2 is greater than 0 then check the next condition that if the root1 is not equal to root then print roots are real and distinct. If the third condition is false then print roots are real and equal. If the second condition is false then print one root is real and other is not. If first condition is false then print roots are not real.

Source Code:

```
import java.util.*;
class Roots
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a value");
        double a=sc.nextInt();
        System.out.println("Enter b value");
        double b=sc.nextInt();
        System.out.println("Enter c value");
        double c=sc.nextInt();
        double root1,root2;
        root1=(-b+Math.sqrt(b*b-4*a*c))/(2*a);
        root2=(-b-Math.sqrt(b*b-4*a*c))/(2*a);
        System.out.println("Root1 is"+root1 + "Root2 is"+root2);
        if(root1>0)
        {
            if(root2>0)
            {
                if(root1!=root2)
                {
                    System.out.println("Roots are real and distinct");
                }
            }
            else
            {
                System.out.println("Roots are real and equal");
            }
        }
        else
        {
            System.out.println("One root is real and other is not");
        }
    }
}
```

}

Output:



```
C:\Windows\system32\cmd.exe
Roots are not real
C:\madhu1>javac Roots.java
C:\madhu1>java Roots
Enter a value
4
Enter b value
8
Enter c value
2
Root1 is-0.2928932188134524Root2 is-1.7071067811865475
Roots are not real
C:\madhu1>_
```

3. Write a java program to print largest of three numbers.

Description: Enter any three integer numbers as an input. Now we check the first number against the second and third number. If it false then we check for second number against third. If it is also false then accordingly third number will be declared the largest number of the given three numbers.

Source code:

```
import java.util.Scanner;
public class Biggest_Number
{
    public static void main(String[] args)
    {
        int x, y, z;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the first number:");
        x = s.nextInt();
        System.out.print("Enter the second number:");
        y = s.nextInt();
        System.out.print("Enter the third number:");
        z = s.nextInt();
```

```

        if(x > y && x > z)
        {
            System.out.println("Largest number is:"+x);
        }
        else if(y > z)
        {
            System.out.println("Largest number is:"+y);
        }
        else
        {
            System.out.println("Largest number is:"+z);
        }
    }
}

```

OUTPUT:

```

C:\Windows\system32\cmd.exe
C:\madhu1>javac BigNum.java
C:\madhu1>java BigNum
Enter the first number:45
Enter the second number:12
Enter the third number:22
Largest number is:45
C:\madhu1>

```

4. Write a java program to print 1 to 100 numbers using do-while loop.

Description: do-while loop is an exit control loop. In do-while loop the body gets executed once. From the next time the condition is checked, if the condition is true then the body of do-while loop gets executed. The body of the loop executes until the condition becomes false.

Source code:

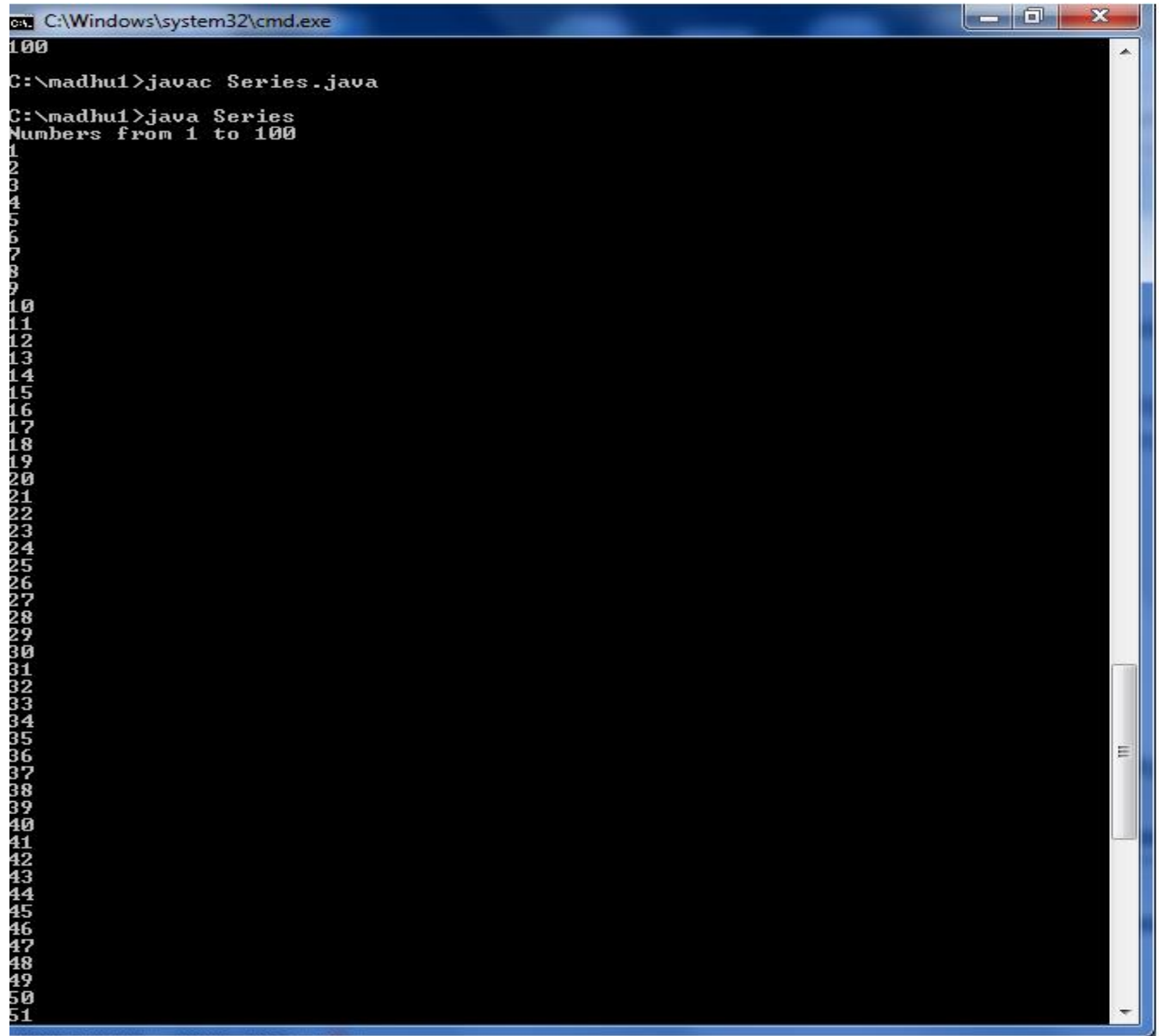
```

class Series
{

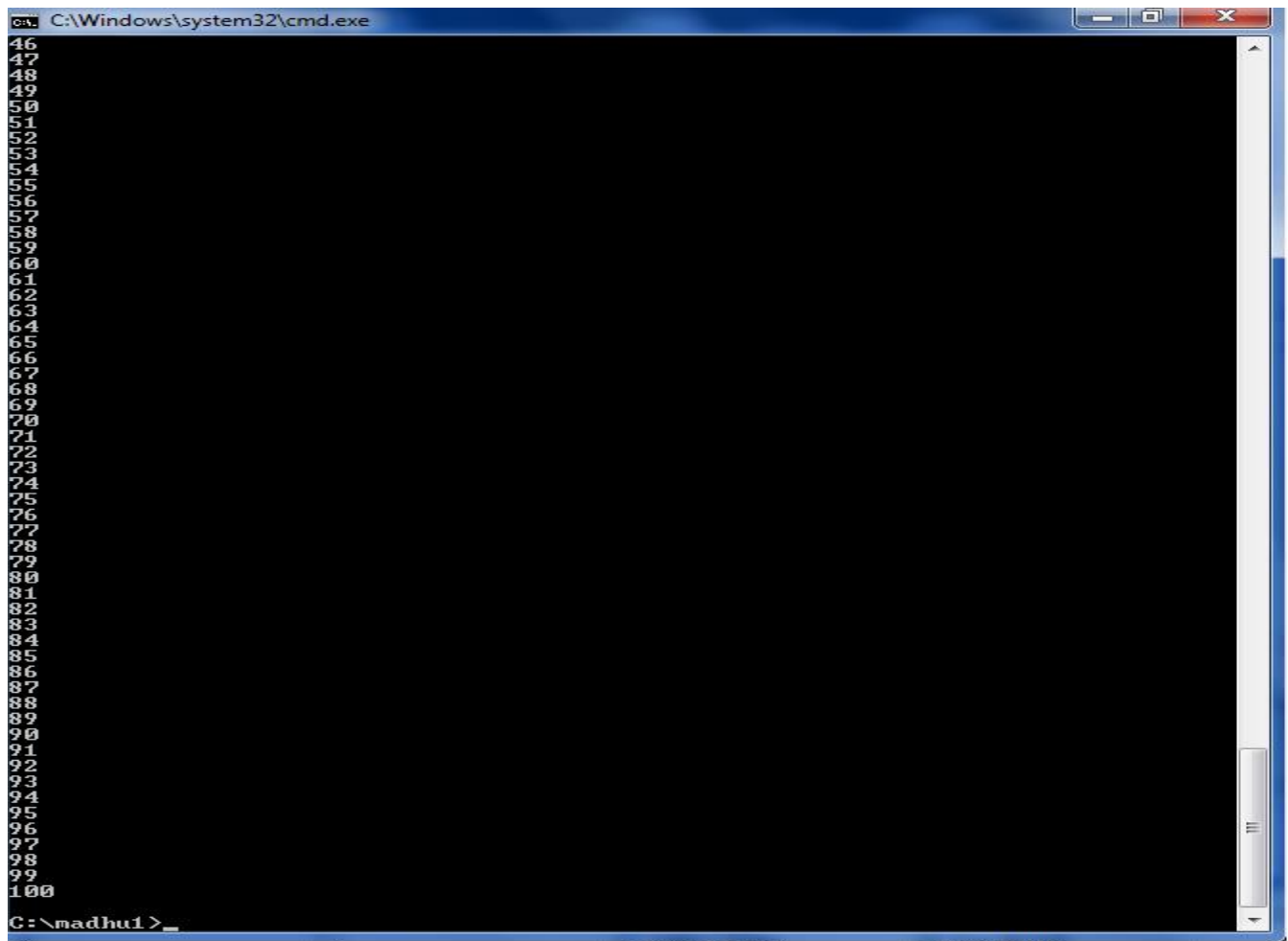
```

```
public static void main(String args[]){
System.out.println("Numbers from 1 to 100");
int i=1;
do
{
System.out.println(i);
i=i+1;
}
while(i<=100);
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
100
C:\madhu1>javac Series.java
C:\madhu1>java Series
Numbers from 1 to 100
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
```



5. Write a java program to print prime numbers up to n.

Description: We use two for loops one for counting the numbers upto n and second nested for loop for validating if the number is prime or not.

Source code:

```
import java.util.*;

class primeNos
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int i,j,n,c;
        System.out.println("Enter the number till which you want prime numbers");
        n=sc.nextInt();
        System.out.println("Prime numbers are :-");

        for(i=2;i<=n;i++)
        {
            c=0;
            for(j=1;j<=i;j++)
            {
                if(i%j==0)
                {
                    c++;
                }
            }
        }
    }
}
```

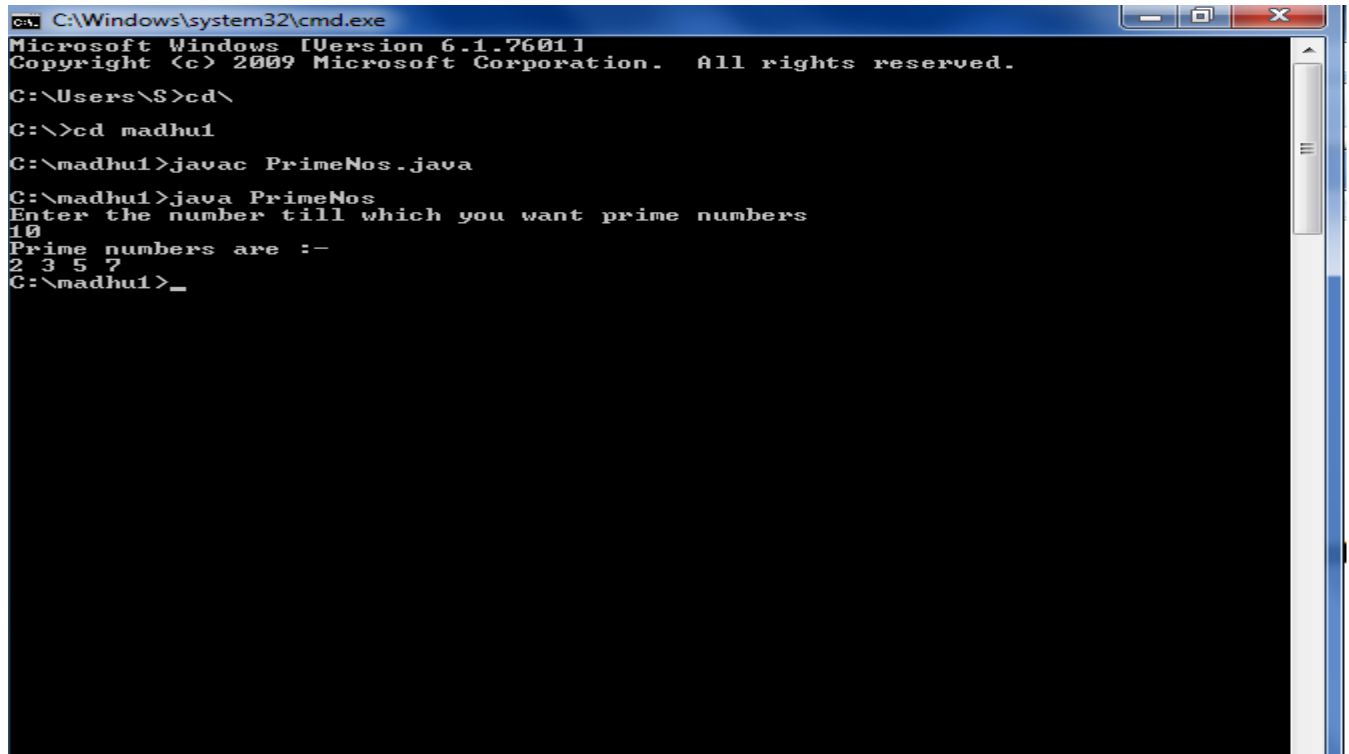
```

    }
}

if(c==2)
{
    System.out.print(i+" ");
}
}
}
}

```

Output:



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\S>cd\
C:\>cd madhu1
C:\madhu1>javac PrimeNos.java
C:\madhu1>java PrimeNos
Enter the number till which you want prime numbers
10
Prime numbers are :-
2 3 5 7
C:\madhu1>_

```

6. Write a program to print Fibonacci series.

Description: In Fibonacci series, *next number is the sum of previous two numbers* for example 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 etc. The first two numbers of Fibonacci series are 0 and 1.

There are two ways to write the Fibonacci series program in java:

- Fibonacci Series without using recursion
- Fibonacci Series using recursion

Source code:

```

class FabinocciSeries
{
    public static void main(String args[])
    {
        int a=0,b=1,sum=0,count=2,n=10;
        System.out.println("Fibinocci series");
    }
}

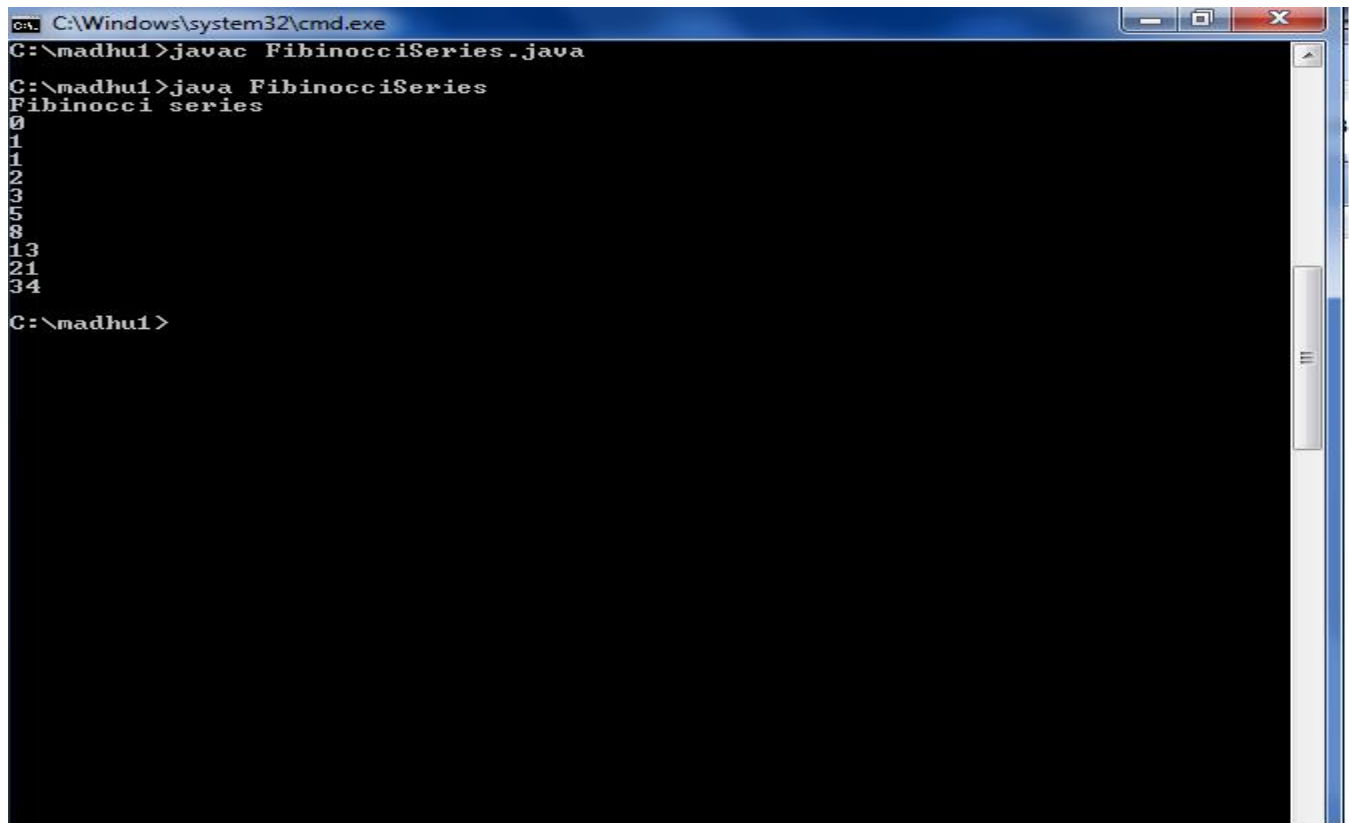
```

```

System.out.println(a);
System.out.println(b);
do
{
sum=a+b;
count=count+1;
a=b;
b=sum;
System.out.println(sum);
}while(count<n);
}
}

```

Output:



```

C:\Windows\system32\cmd.exe
C:\madhu1>javac FabinocciSeries.java
C:\madhu1>java FabinocciSeries
Fabinocci series
0
1
1
2
3
5
8
13
21
34
C:\madhu1>

```

7. Write a java program to print 4 entered numbers in ascending order.

Description: First read four numbers from the user and then add that four numbers and make use of for loop. In the for loop use the condition that I is less than the sum of numbers entered. In for loop if condition is used that the I value is matched with the entered values. If both are equal then print the number. In that way sorting is done.

Source Code:

```

import java.util.*;
class Series
{
Public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter first number");

```

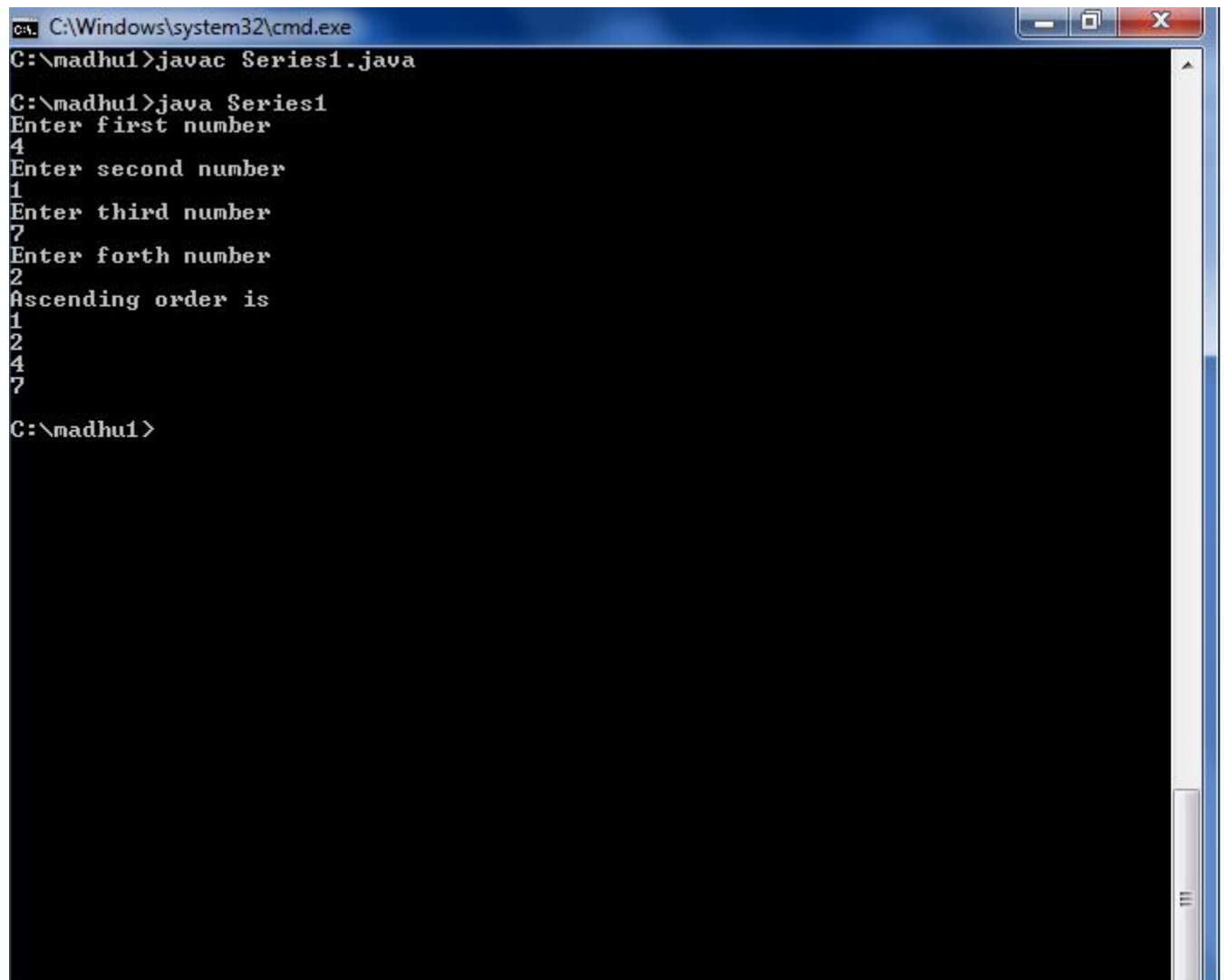


```

        int a=sc.nextInt();
        System.out.println("Enter second number");
        int b=sc.nextInt();
        System.out.println("Enter third number");
        int c=sc.nextInt();
        System.out.println("Enter forth number");
        int d=sc.nextInt();
        System.out.println("Ascending order is");
        int sum=a+b+c+d;
        for(int i=1;i<=sum;i++)
        {
            if(i==a|| i==b||i==c||i==d)
            {
                System.out.println(i);
            }
        }
    }
}
}
}

```

Output:



```

C:\Windows\system32\cmd.exe
C:\madhu1>javac Series1.java
C:\madhu1>java Series1
Enter first number
4
Enter second number
1
Enter third number
7
Enter forth number
2
Ascending order is
1
2
4
7
C:\madhu1>

```

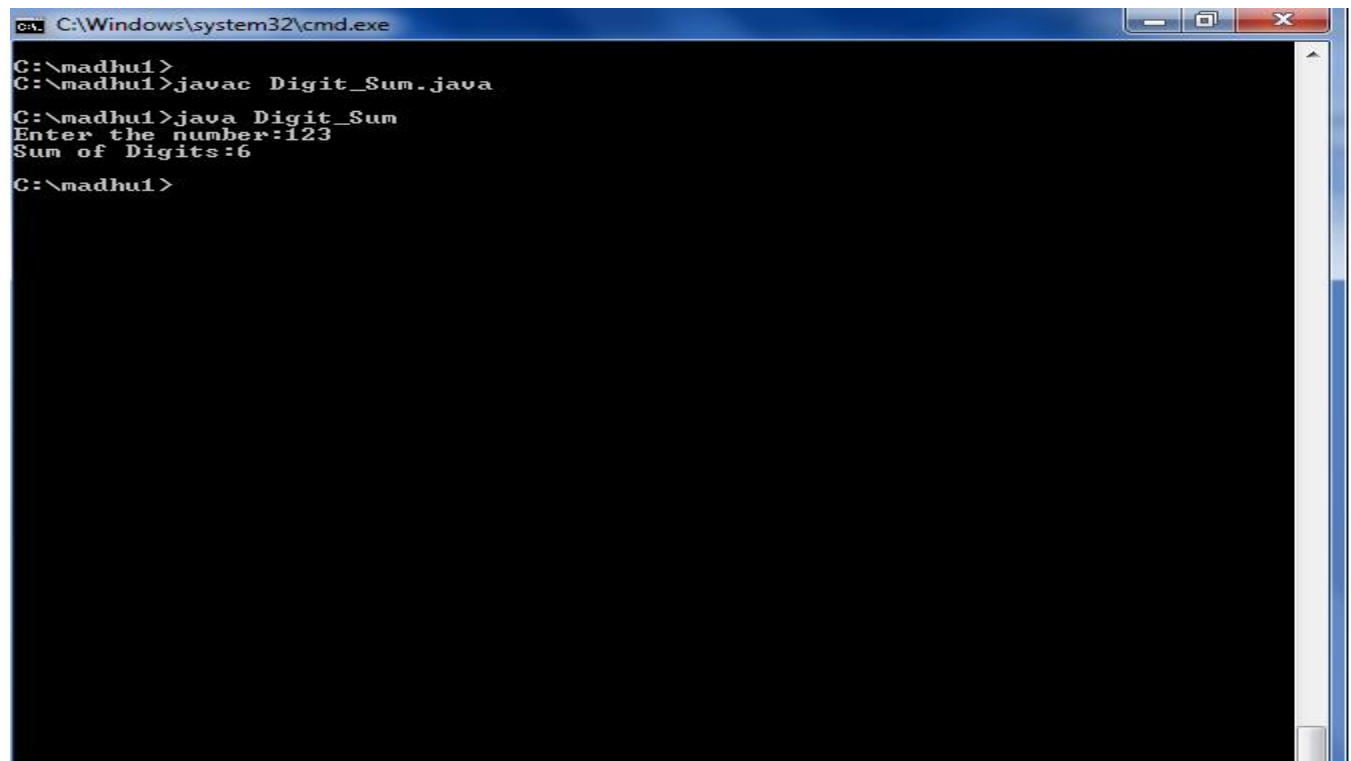
8. Write a java program for accepting a number from a keyboard and to display the sum of digits of a given number.

Description: Enter any integer number as input. After that we use modulus and division operation respectively to find the sum of digits of number as output.

Source Code:

```
import java.util.Scanner;
public class Digit_Sum
{
    public static void main(String args[])
    {
        int m, n, sum = 0;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the number:");
        m = s.nextInt();
        while(m > 0)
        {
            n = m % 10;
            sum = sum + n;
            m = m / 10;
        }
        System.out.println("Sum of Digits:"+sum);
    }
}
```

Output:



```
C:\Windows\system32\cmd.exe
C:\madhu1>
C:\madhu1>javac Digit_Sum.java
C:\madhu1>java Digit_Sum
Enter the number:123
Sum of Digits:6
C:\madhu1>
```

9. Write a java program to display “*” as a given below

(a). * (b). * * * *

 * * * * *

 * * * * * *

 * * * * * *

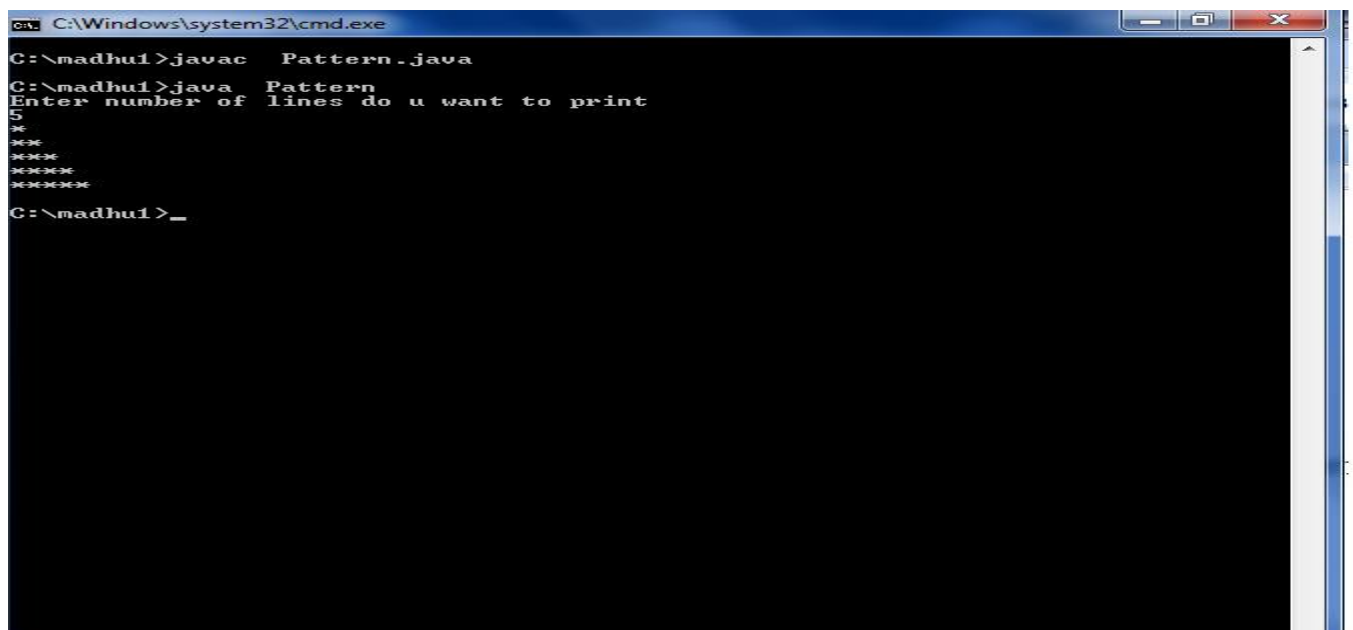
Description: First accept how many lines the pattern have to print. And then initialize I and j variables. By comparing the two variables the following patterns will be printed.

Source Code:

(a).

```
import java.util.*;
class Pattern
{
public static void main(String args[])
{
int n,i,j;
Scanner sc=new Scanner(System.in);
System.out.println("Enter number of lines do u want to print");
n=sc.nextInt();
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(i>=j)
{
System.out.print("*");
}
}
System.out.println();
}
}
}
```

Output:

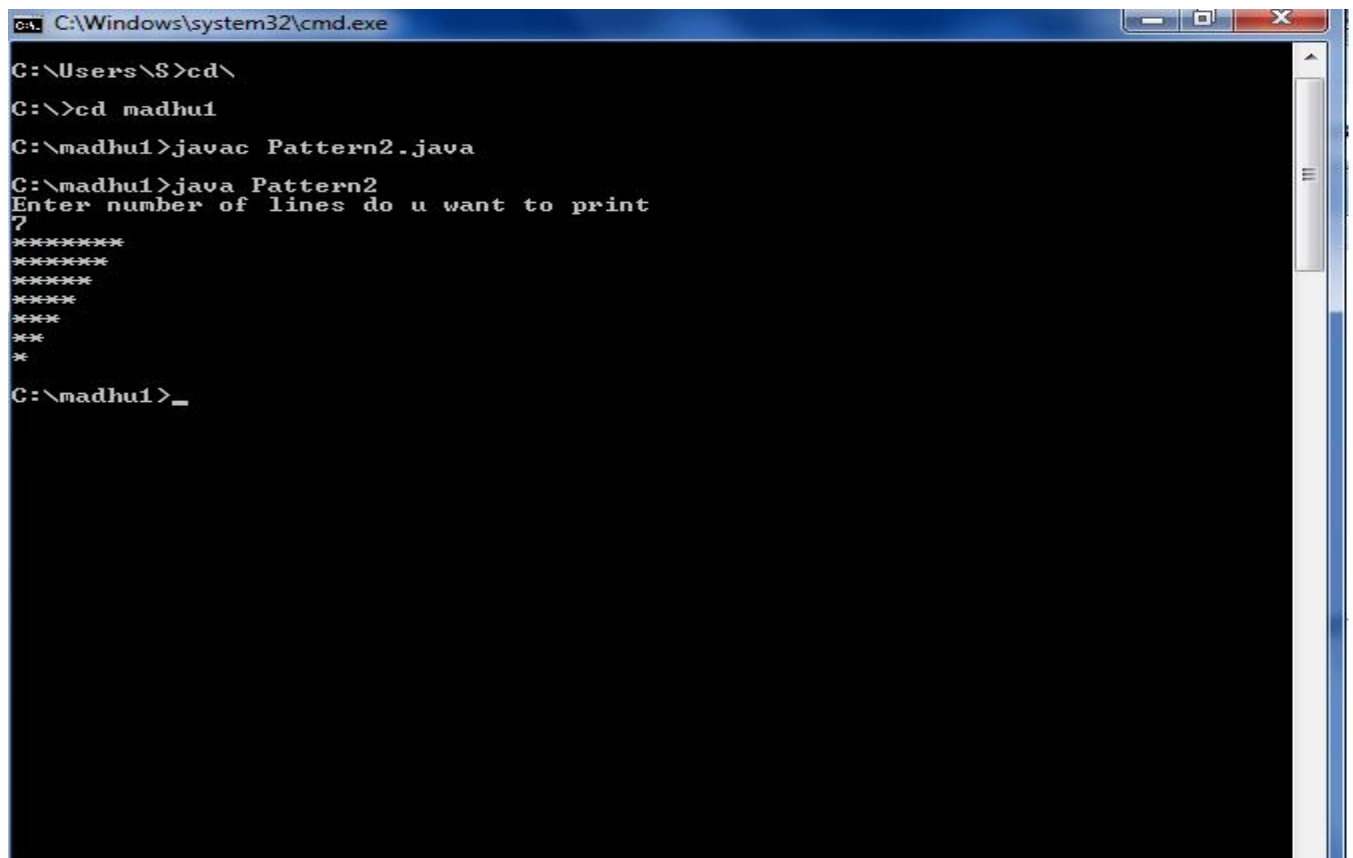


```
C:\Windows\system32\cmd.exe
C:\madhu1>javac  Pattern.java
C:\madhu1>java  Pattern
Enter number of lines do u want to print
5
*
***
***
*****
C:\madhu1>_
```

(b).

```
import java.util.*;
class Pattern2
{
public static void main(String args[])
{
int n,i,j;
Scanner sc=new Scanner(System.in);
System.out.println("Enter number of lines do u want to print");
n=sc.nextInt();
for(i=1;i<=n;i++)
{
for(j=i;j<=n;j++)
{
System.out.print("*");
}
System.out.println();
}
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
C:\Users\S>cd\
C:\>cd madhu1
C:\madhu1>javac Pattern2.java
C:\madhu1>java Pattern2
Enter number of lines do u want to print
7
*****
*****
*****
*****
****
***
**
*
C:\madhu1>_
```

10. Write a java program to print the factorial of given number using while loop.

Description: Factorial Program in Java: Factorial of n is the *product of all positive descending integers*. Factorial of n is denoted by $n!$. For example:

1. $4! = 4 \times 3 \times 2 \times 1 = 24$

2. $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

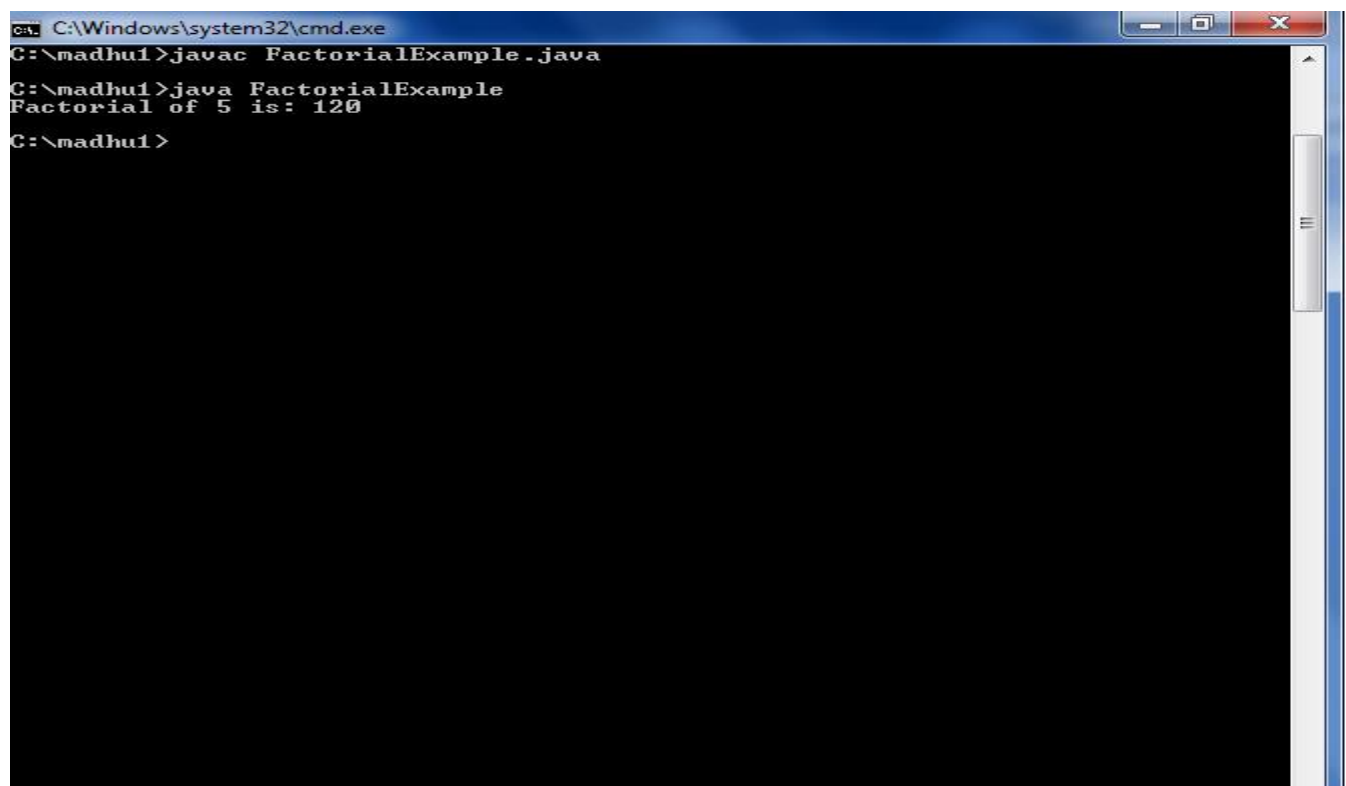
There are many ways to write the factorial program in java language. Let's see the 2 ways to write the factorial program in java.

- Factorial Program using loop
- Factorial Program using recursion

Source Code:

```
class FactorialExample{
    public static void main(String args[]){
        int i,fact=1;
        int number=5;//It is the number to calculate factorial
        for(i=1;i<=number;i++){
            fact=fact*i;
        }
        System.out.println("Factorial of "+number+" is: "+fact);
    }
}
```

Output:



```
C:\Windows\system32\cmd.exe
G:\madhu1>javac FactorialExample.java
G:\madhu1>java FactorialExample
Factorial of 5 is: 120
G:\madhu1>
```

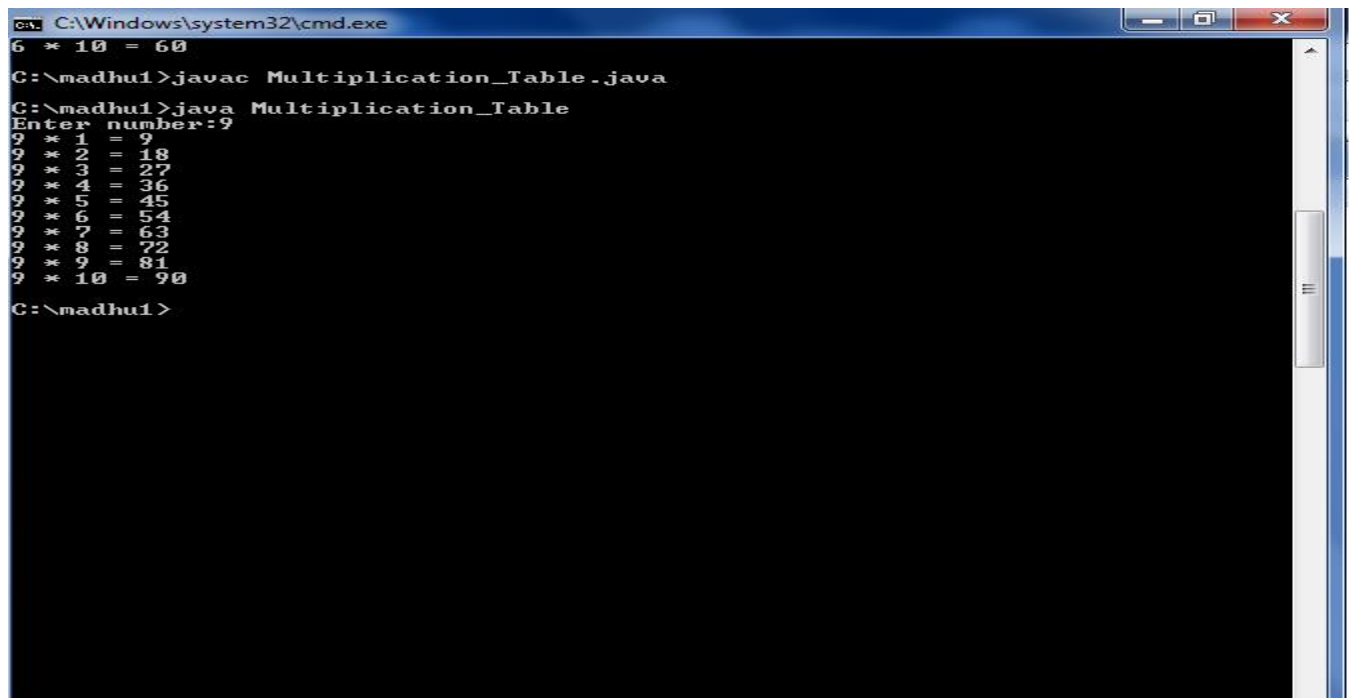
11. write a java program to print the given number to print the multiplication table.

Description: Enter any integer number as input of which you want multiplication table. After that we use for loop from one to ten to generate multiplication of that number.

Source Code:

```
import java.util.Scanner;
public class Multiplication_Table
{
    public static void main(String[] args)
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter number:");
        int n=s.nextInt();
        for(int i=1; i <= 10; i++)
        {
            System.out.println(n+" * "+i+" = "+n*i);
        }
    }
}
```

Output:



The screenshot shows a Windows command prompt window with the title bar "C:\Windows\system32\cmd.exe". The prompt is at "C:\madhu1>". The user has entered "javac Multiplication_Table.java" and "java Multiplication_Table". The program prompts "Enter number:9". The output displays the multiplication table for 9, from 9 * 1 to 9 * 10. The prompt "C:\madhu1>" is visible at the bottom.

```
C:\Windows\system32\cmd.exe
6 * 10 = 60
C:\madhu1>javac Multiplication_Table.java
C:\madhu1>java Multiplication_Table
Enter number:9
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90
C:\madhu1>
```

12. Write a java program to display the output of a following series

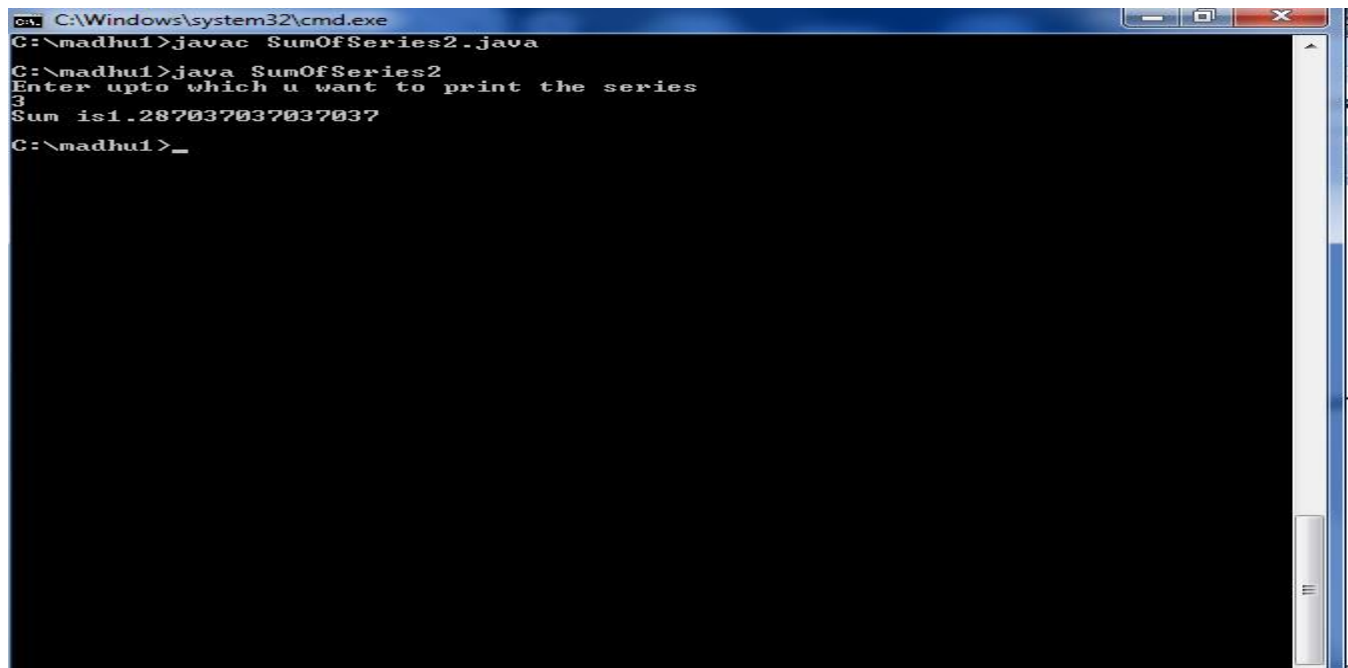
1+1/4+1/9+1/16+.....1/n^2.

Description: Prompt the user to enter up to which he wants to print the series. Initialize the variable sum with 0. Then in for loop square the i value and then add it to the sum variable. Then out of loop print the sum value.

Source Code:

```
import java.util.Scanner;
class SumOfSeries2
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter upto which u want to print the series");
int n=sc.nextInt();
double sum=0,res;
for(int i=1;i<=n;i++)
{
res=(1/(Math.pow(i,i)));
sum=res+sum;
}
System.out.println("Sum is"+sum);
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
C:\madhu1>javac SumOfSeries2.java
C:\madhu1>java SumOfSeries2
Enter upto which u want to print the series
3
Sum is1.287037037037037
C:\madhu1>_
```

EXERCISE-2

(OPERATION, EXPRESSIONS, CONTROL FLOW, STRINGS)

13. write a java program to search for an element in a given list of elements using binary search mechanism.

Description: Note that input numbers must be in *ascending* order. If they are not, you must sort them first.

Source code:

```
import java.util.*;
class Bsearch
{
void sort(int arr[],int arrsize)
{
int i,j,temp;
for(i=0;i<arrsize;i++)
{
for(j=i+1;j<arrsize;j++)
{
if(arr[i]>arr[j])
{
temp=arr[i];
arr[i]=arr[j];
arr[j]=temp;
}
}
}
}
void search(int arr[],int arrsize,int ele)
{
int i=0,flag=0;
while(i<arrsize)
{
if(ele==arr[i])
{
System.out.println("Element found in" +(i+1) +""+"position");
flag=1;
break;
}
else
{
i=i+1;
}
}
if(flag==0)
{
System.out.println("Element not found");
}
}
public static void main(String args[])
{
Bsearch b=new Bsearch();
Scanner sc=new Scanner(System.in);
int arr[]=new int[20];
int temp=0,i;
System.out.println("Enter size of array");
```



```

int arrsize=sc.nextInt();
System.out.println("Enter array elements");
for(i=0;i<arrsize;i++)
{
arr[i]=sc.nextInt();
}
System.out.println("Entered array elements are");
for(i=0;i<arrsize;i++)
{
System.out.println(""+arr[i]);
}
b.sort(arr,arrsize);
System.out.println("After sorting");
for(i=0;i<arrsize;i++)
{
System.out.println(""+arr[i]);
}
System.out.println("Enter number which u want to search");
int ele=sc.nextInt();
b.search(arr,arrsize,ele);
}
}

```

Output:

```

C:\Windows\system32\cmd.exe
?
C:\madhu1>javac Bsearch.java
C:\madhu1>java Bsearch
Enter size of array
5
Enter array elements
1
7
2
5
3
Entered array elements are
1
7
2
5
3
After sorting
1
2
3
5
7
Enter number which u want to search
7
Element found in 5 position
C:\madhu1>_

```

14.write a java program to sort for an element in a given list of elements using bubblesort.

Description:

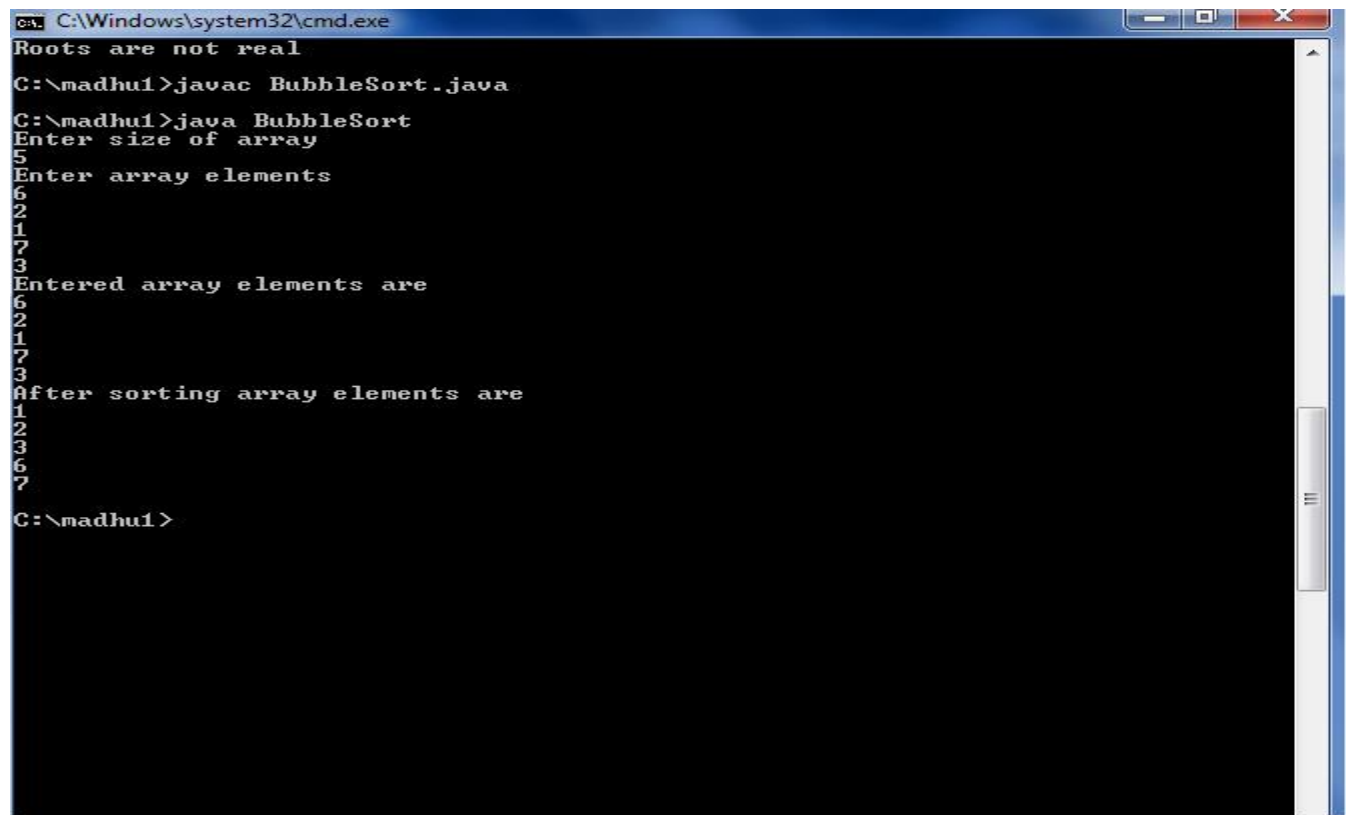
Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

Source Code:

```
import java.util.*;
class BubbleSort
{
void bubblesort(int arr[],int arrsize)
{
int temp,i,j;
for(i=0;i<arrsize -1;i++)
{
for(j=0;j<arrsize-i-1;j++)
{
if(arr[j]>arr[j+1])
{
temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
}
}
}
System.out.println("After sorting array elements are");
for(i=0;i<arrsize;i++)
{
System.out.println(""+arr[i]);
}
}
public static void main(String args[])
{
BubbleSort b=new BubbleSort();
Scanner sc=new Scanner(System.in);
int arr[]=new int[20];
int i;
System.out.println("Enter size of array");
int arrsize=sc.nextInt();
System.out.println("Enter array elements");
for(i=0;i<arrsize;i++)
{
arr[i]=sc.nextInt();
}
System.out.println("Entered array elements are");
for(i=0;i<arrsize;i++)
{
System.out.println(""+arr[i]);
}
b.bubblesort(arr,arrsize);
System.out.println("After sorting array elements are");
for(i=0;i<arrsize;i++)
{
System.out.println(arr[i]);
}
```

```
}  
}  
}
```

Output:



```
C:\Windows\system32\cmd.exe  
Roots are not real  
C:\madhu1>javac BubbleSort.java  
C:\madhu1>java BubbleSort  
Enter size of array  
5  
Enter array elements  
6  
2  
1  
7  
3  
Entered array elements are  
6  
2  
1  
7  
3  
After sorting array elements are  
1  
2  
3  
6  
7  
C:\madhu1>
```

15.write a java program to sort for an element in a given list of elements using mergesort.

Description:

Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. The merge() function is used for merging two halves. The merge(arr, l, m, r) is key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one.

Source Code:

```
import java.util.*;  
public class MSort  
{  
    public static void merge(int a[],int l,int m,int h)  
    {  
        int i , j, c=l;  
        int b[]=new int[h+1];  
        for(i = l, j = m+1; i<=m && j<=h; c++)  
        {  
            if(a[i] <= a[j])  
            {
```

```

        b[c] = a[i++];
    }
    else
    {
        b[c] = a[j++];
    }
}
while(i <= m )
{
    b[c++] = a[i++];
}
while(j<=h)
{
    b[c++] = a[j++];
}
for(i = l ; i <= h; i++)
{
    a[i] = b[i];
}
}

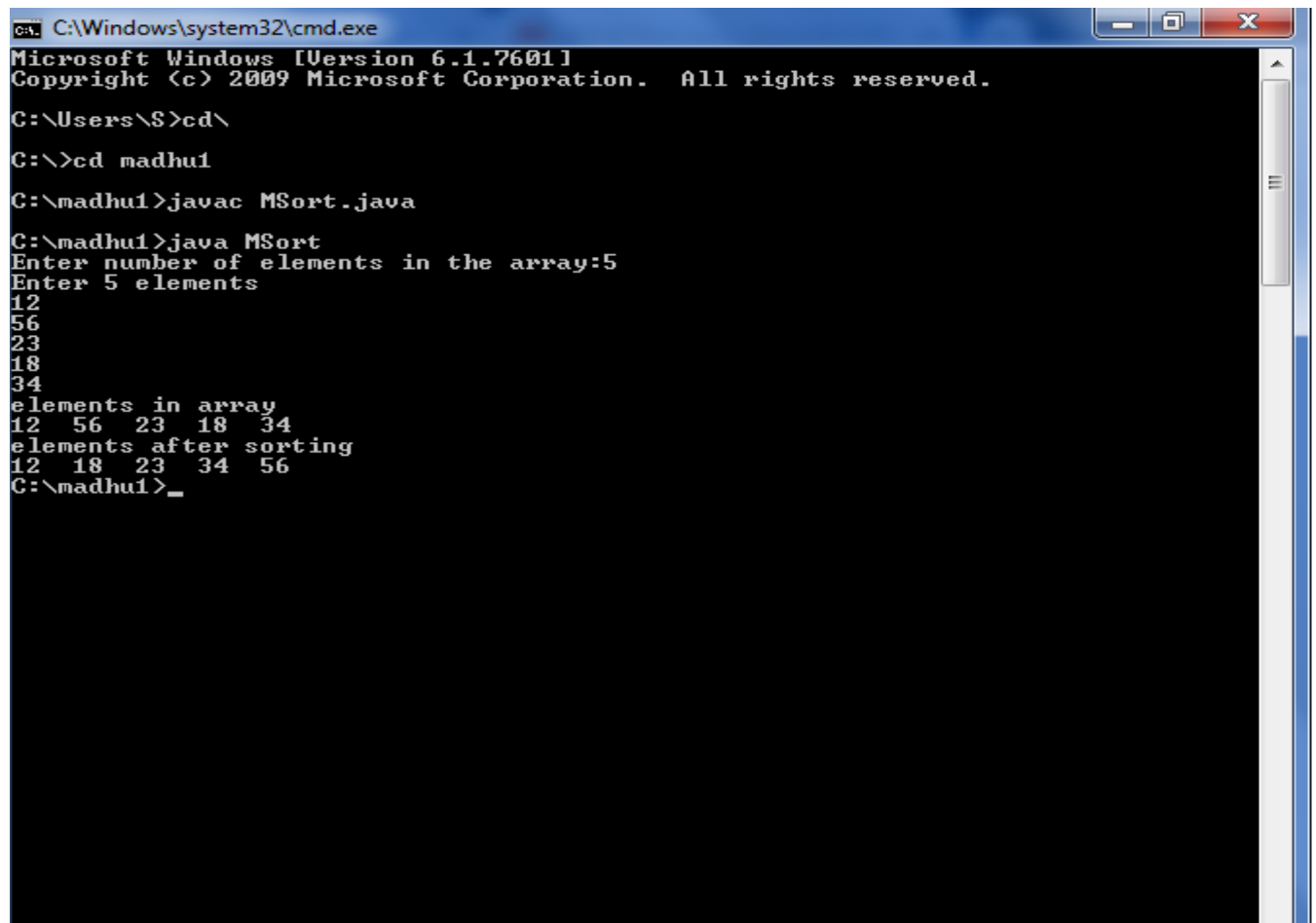
public static void Sort(int a[],int l,int h)
{
    if(l<h)
    {
        int m=(l+h)/2;
        Sort(a,l,m);
        Sort(a,m+1,h);
        merge(a,l,m,h);
    }
}

public static void main(String[] args)
{
    int n,i, res;
    Scanner s = new Scanner(System.in);
    System.out.print("Enter number of elements in the array:");
    n = s.nextInt();
    int a[] = new int[n];
    System.out.println("Enter "+n+" elements ");
    for( i=0; i < n; i++)
    {
        a[i] = s.nextInt();
    }
    System.out.println( "elements in array ");
    for( i=0; i < a.length; i++)
    {
        System.out.print(a[i]+" ");
    }
    Sort(a,0,n-1);
    System.out.println( "\nelements after sorting");
    for( i=0; i < a.length; i++)
    {
        System.out.print(a[i]+" ");
    }
}

```

```
}
```

Output:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\S>cd\
C:\>cd madhu1
C:\madhu1>javac MSort.java
C:\madhu1>java MSort
Enter number of elements in the array:5
Enter 5 elements
12
56
23
18
34
elements in array
12 56 23 18 34
elements after sorting
12 18 23 34 56
C:\madhu1>_
```

16.write a java program using String Buffer to perform append, insert, delete and reverse operations.

Description:

StringBuffer is a peer class of String that provides much of the functionality of strings. String represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences.

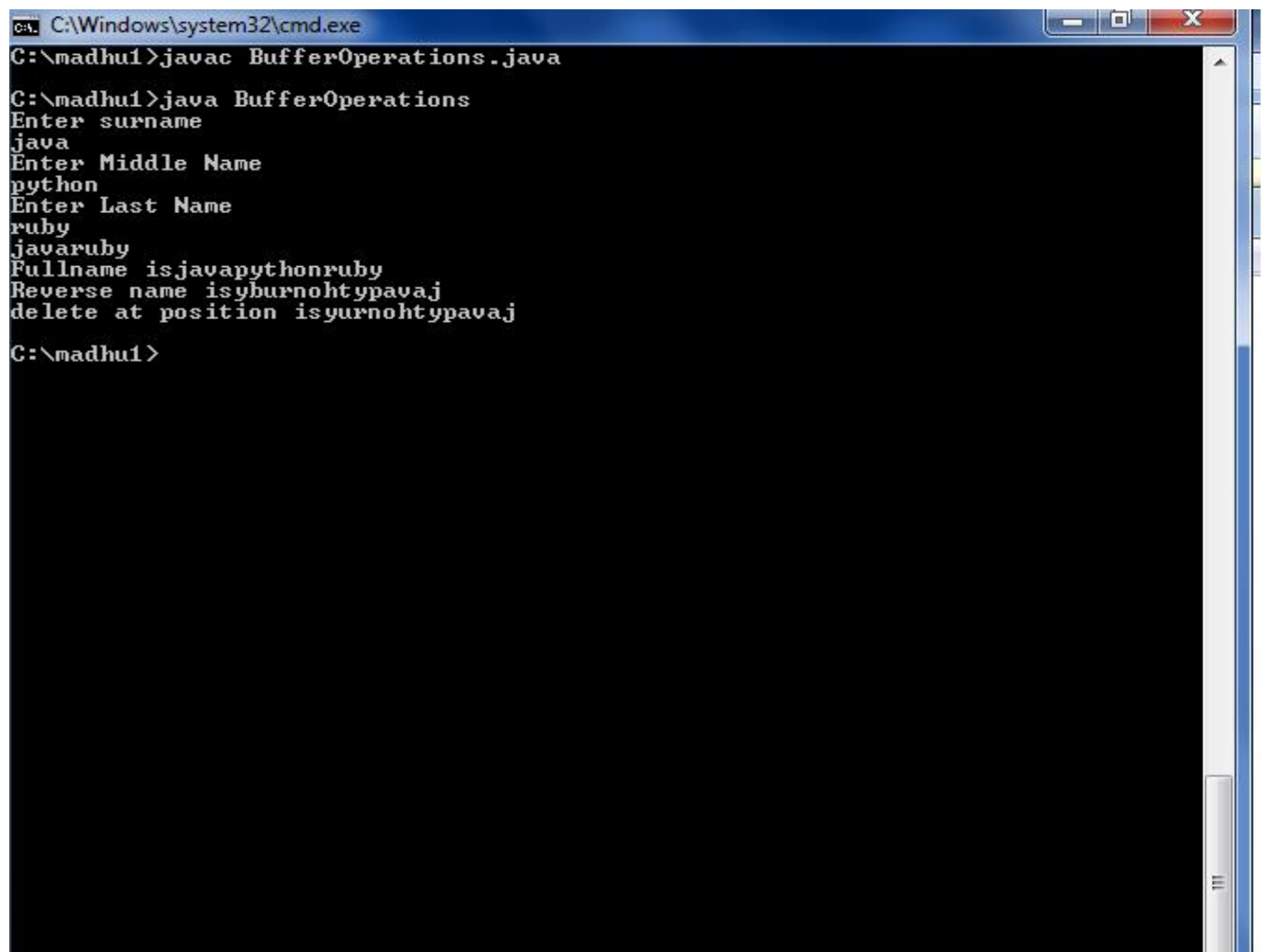
StringBuffer may have characters and substrings inserted in the middle or appended to the end. It will automatically grow to make room for such additions and often has more characters preallocated than are actually needed.

Source Code:

```
import java.util.*;
class BufferOperations
{
public static void main(String args[])
{
Scanner s=new Scanner(System.in);
System.out.println("Enter surname");
```

```
String surname=s.nextLine();
System.out.println("Enter Middle Name");
String middlename=s.nextLine();
System.out.println("Enter Last Name");
String lastname=s.nextLine();
StringBuffer sb=new StringBuffer();
sb.append(surname);
sb.append(lastname);
System.out.println(sb);
int n=surname.length();
sb.insert(n,middlename);
System.out.println("Fullname is"+sb);
System.out.println("Reverse name is"+sb.reverse());
System.out.println("delete at position is"+sb.delete(1,2));
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
C:\madhu1>javac BufferOperations.java
C:\madhu1>java BufferOperations
Enter surname
java
Enter Middle Name
python
Enter Last Name
ruby
Fullname isjavapythonruby
Reverse name isyburnohtypavaj
delete at position isyurnohtypavaj
C:\madhu1>
```

17. write a java program to sort the elements of an array in ascending order.

Description:

In this program, we need to sort the given array in ascending order such that elements will be arranged from smallest to largest. This can be achieved through two loops. The outer loop will select an element, and inner loop allows us to compare selected element with rest of the elements.

Elements will be sorted in such a way that the smallest element will appear on extreme left which in this case is 1. The largest element will appear on extreme right which in this case is 8.

Source code:

```
import java.util.*;
class Sorting
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
int arr[]=new int[20];
int temp=0,i;
System.out.println("Enter size of array");
int arrsize=sc.nextInt();
System.out.println("Enter array elements");
for(i=0;i<arrsize;i++)
{
arr[i]=sc.nextInt();
}
System.out.println("Entered array elements are");
for(i=0;i<arrsize;i++)
{
System.out.println(""+arr[i]);
}
for(i=0;i<arrsize;i++)
{
for(int j=i+1;j<arrsize;j++)
{
if(arr[i]>arr[j])
{
temp=arr[i];
arr[i]=arr[j];
arr[j]=temp;
}
}
}
System.out.println("After sorting array elements are");
for(i=0;i<arrsize;i++)
{
System.out.println(""+arr[i]);
}
System.out.println("Sorting done");
}
}
```

Output:

```
C:\Windows\system32\cmd.exe
Element found in 5 position
C:\madhu1>javac Sorting.java
C:\madhu1>java Sorting
Enter size of array
5
Enter array elements
1
6
2
4
3
Entered array elements are
1
6
2
4
3
After sorting array elements are
1
2
3
4
6
Sorting done
C:\madhu1>_
```

EXERCISE-3 (Class, Objects)

18. write a java program to implement class mechanism. Create a class, methods and invoke them inside main method.

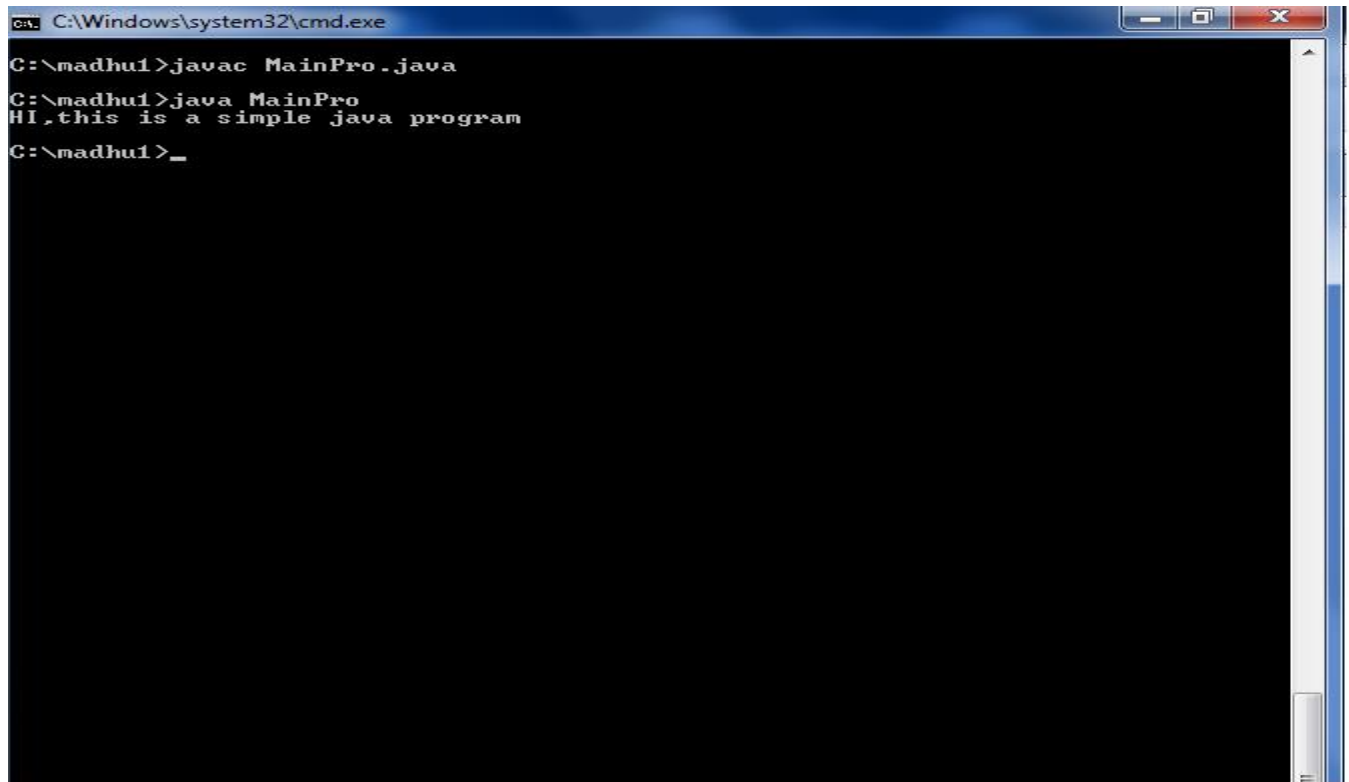
Description:

Create a class using the keyword class and write a method inside the class. Then create a main class and then create an object. With the help of the object call the method which is declared in another class.

Source Code:

```
class MainPro
{
    public static void main(String args[])
    {
        Display d=new Display();
        d.print();
    }
}
class Display
{
    void print()
    {
        System.out.println("HI,this is a simple java program");
    }
}
```


Output:

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The command prompt shows the following sequence of commands and output:

```
C:\madhu1>javac MainPro.java
C:\madhu1>java MainPro
HI,this is a simple java program
C:\madhu1>_
```

19.write a java program to implement constructor.

Description:

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the new() keyword, at least one constructor is called.

It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.

There are two types of constructors in Java: no-arg constructor, and parameterized constructor.

There are two rules defined for the constructor.

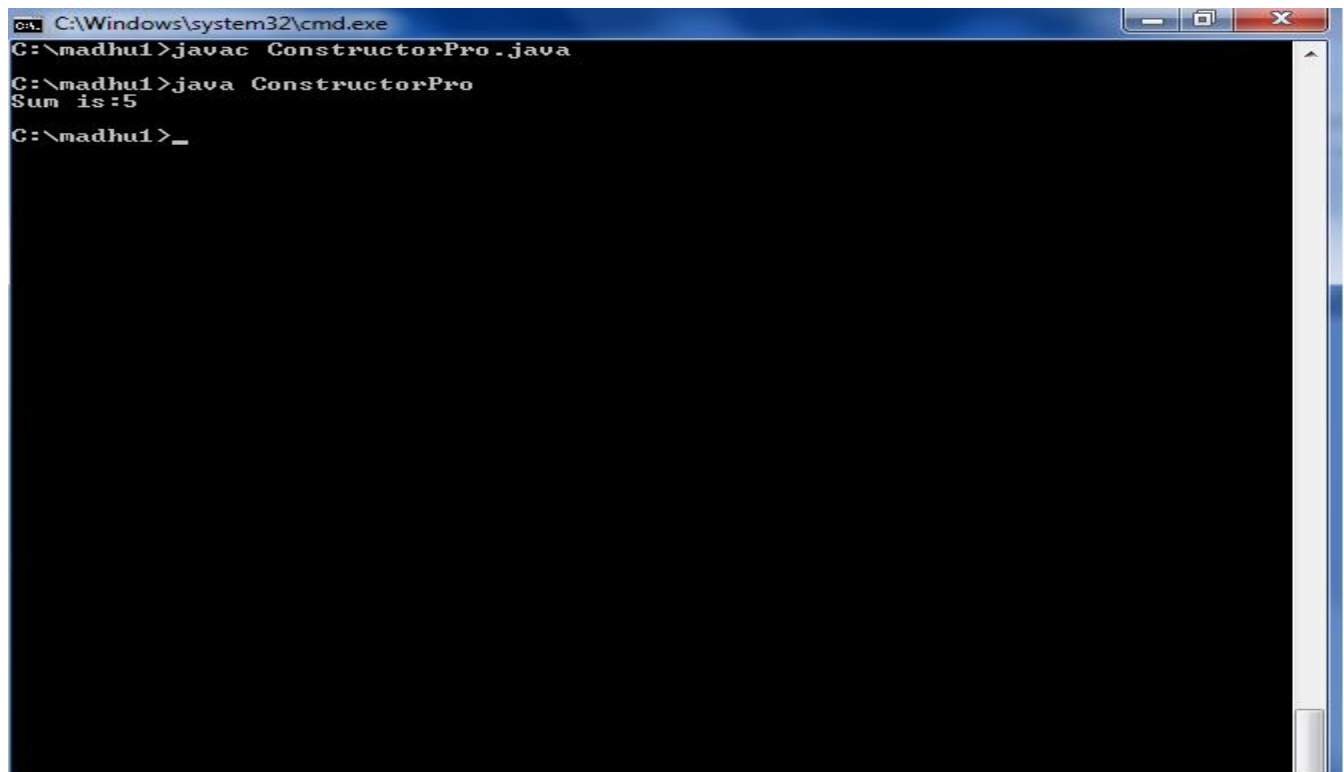
1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

Source code:

```
class ConstructorPro
```

```
{
public static void main(String args[])
{
Cons1 c1=new Cons1(2,3);
c1.add();
}
}
class Cons1
{
int a,b;
Cons1(int a,int b)
{
this.a=a;
this.b=b;
}
void add()
{
int c;
c=a+b;
System.out.println("Sum is:"+c);
}
}
```

Output:



The screenshot shows a Windows command prompt window with the title bar 'C:\Windows\system32\cmd.exe'. The command prompt is at the directory 'C:\madhu1'. The user has entered the command 'javac ConstructorPro.java' to compile the program. The next command entered is 'java ConstructorPro', which has been executed, resulting in the output 'Sum is:5'. The prompt is now waiting for the next command.

```
C:\Windows\system32\cmd.exe
C:\madhu1>javac ConstructorPro.java
C:\madhu1>java ConstructorPro
Sum is:5
C:\madhu1>_
```

20.write a java program to initialize the object through

- (a) by reference variable**
- (b) by method**
- (c) by constructor**

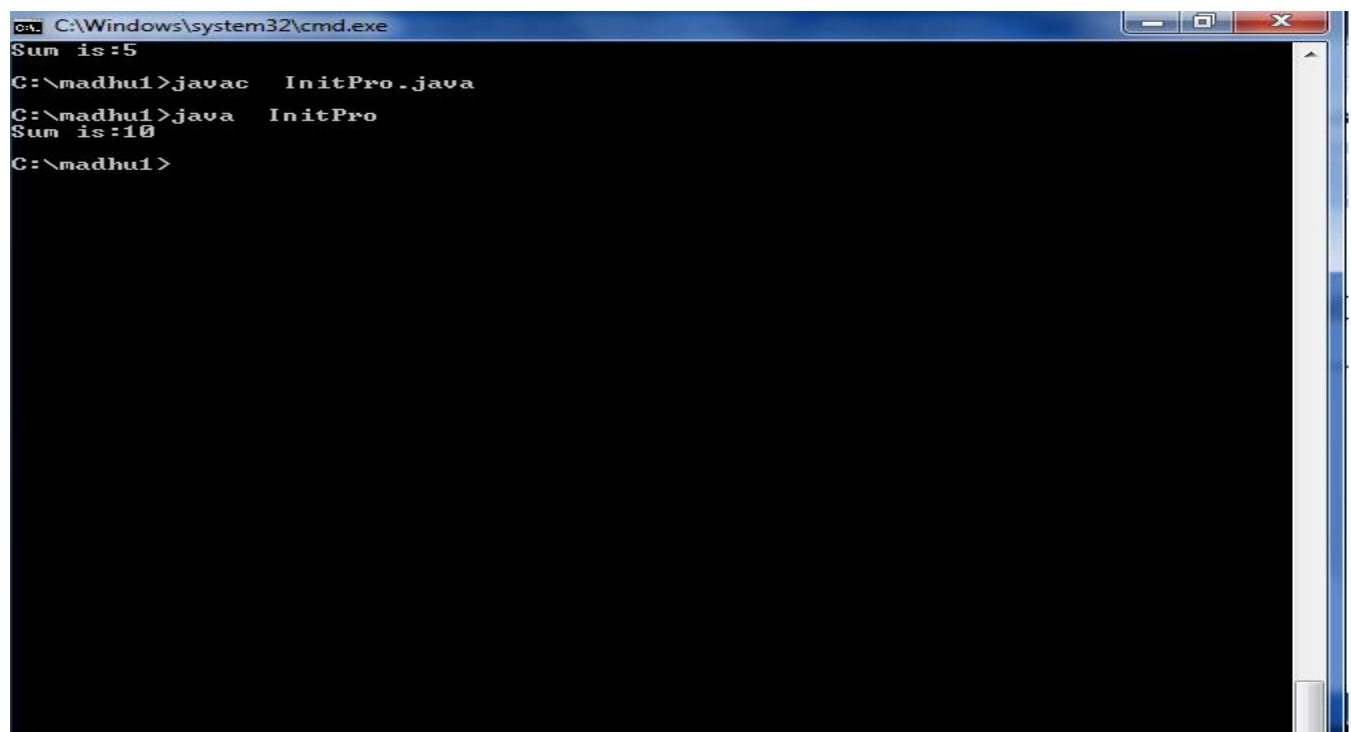
Description:

To initialize the object, we can do that through reference variables or by method or by a constructor.

Source Code:

```
class Init
{
int a,b,d;
Init(int a,int b)
{
this.a=a;
this.b=b;
}
void display(int c)
{
int e=c;
int sum=a+b+d+e;
System.out.println("Sum is:"+sum);
}
}
class InitPro
{
public static void main(String args[])
{
Init i=new Init(1,2);
i.d=3;
i.display(4);
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
Sum is:5
C:\madhu1>javac InitPro.java
C:\madhu1>java InitPro
Sum is:10
C:\madhu1>
```

EXERCISE-4

(METHODS)

21.write a java program to implement constructor overloading.

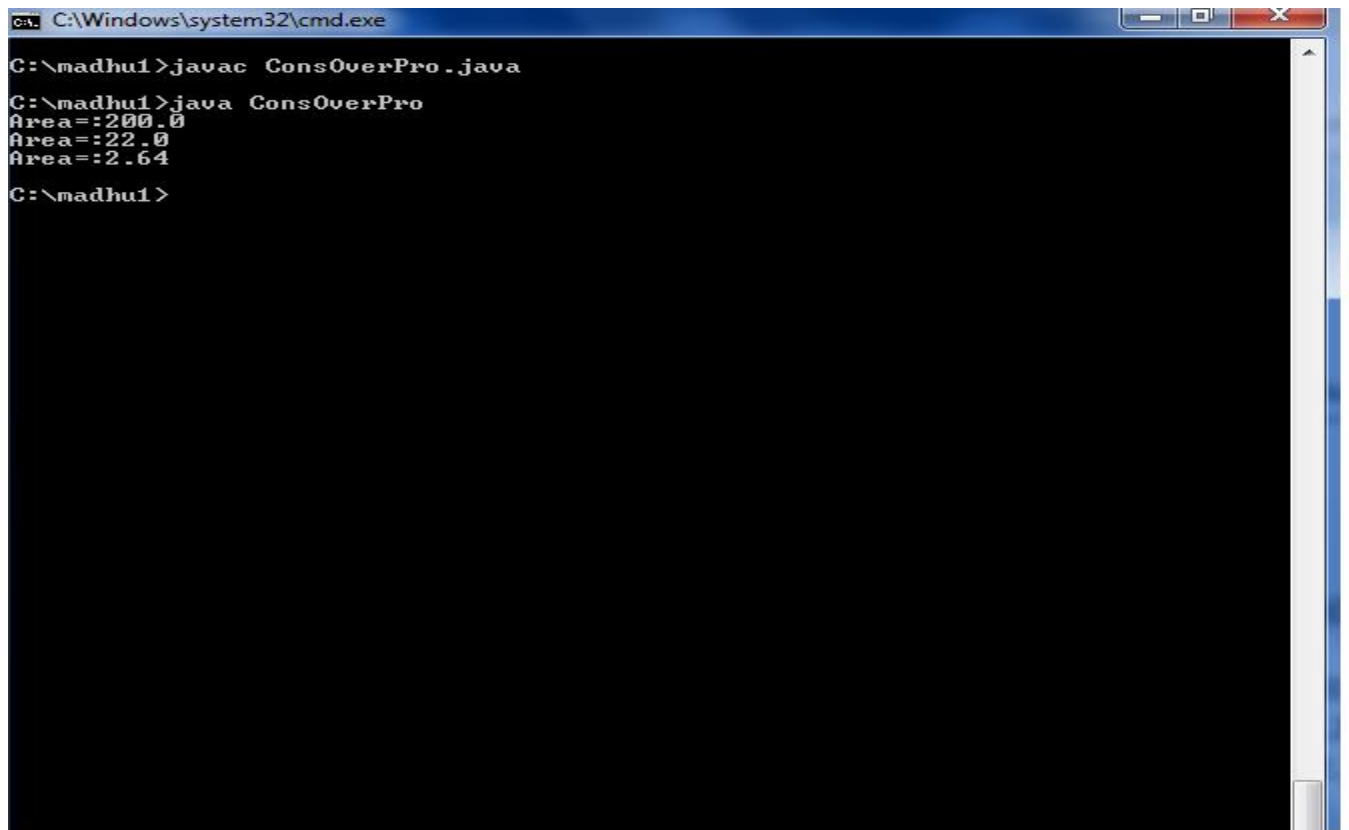
Description:

Constructor overloading is a concept of having more than one constructor with different parameters list, in such a way so that each constructor performs a different task.

Source Code:

```
class ConsOver
{
double x1;
double x2;
ConsOver(int a,int b)
{
this.x1=a;
this.x2=b;
}
ConsOver(int a,double b)
{
this.x1=a;
this.x2=b;
}
ConsOver(double a,double b)
{
this.x1=a;
this.x2=b;
}
void rectangleArea()
{
System.out.println("Area="+x1*x2);
}
}
class ConsOverPro
{
public static void main(String args[])
{
ConsOver c1=new ConsOver(10,20);
ConsOver c2=new ConsOver(10,2.2);
ConsOver c3=new ConsOver(1.2,2.2);
c1.rectangleArea();
c2.rectangleArea();
c3.rectangleArea();
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
C:\madhu1>javac ConsOverPro.java
C:\madhu1>java ConsOverPro
Area=:200.0
Area=:22.0
Area=:2.64
C:\madhu1>
```

22.write a java program to implement method overloading.

Description:

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

Different ways to overload the method

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

Source Code:

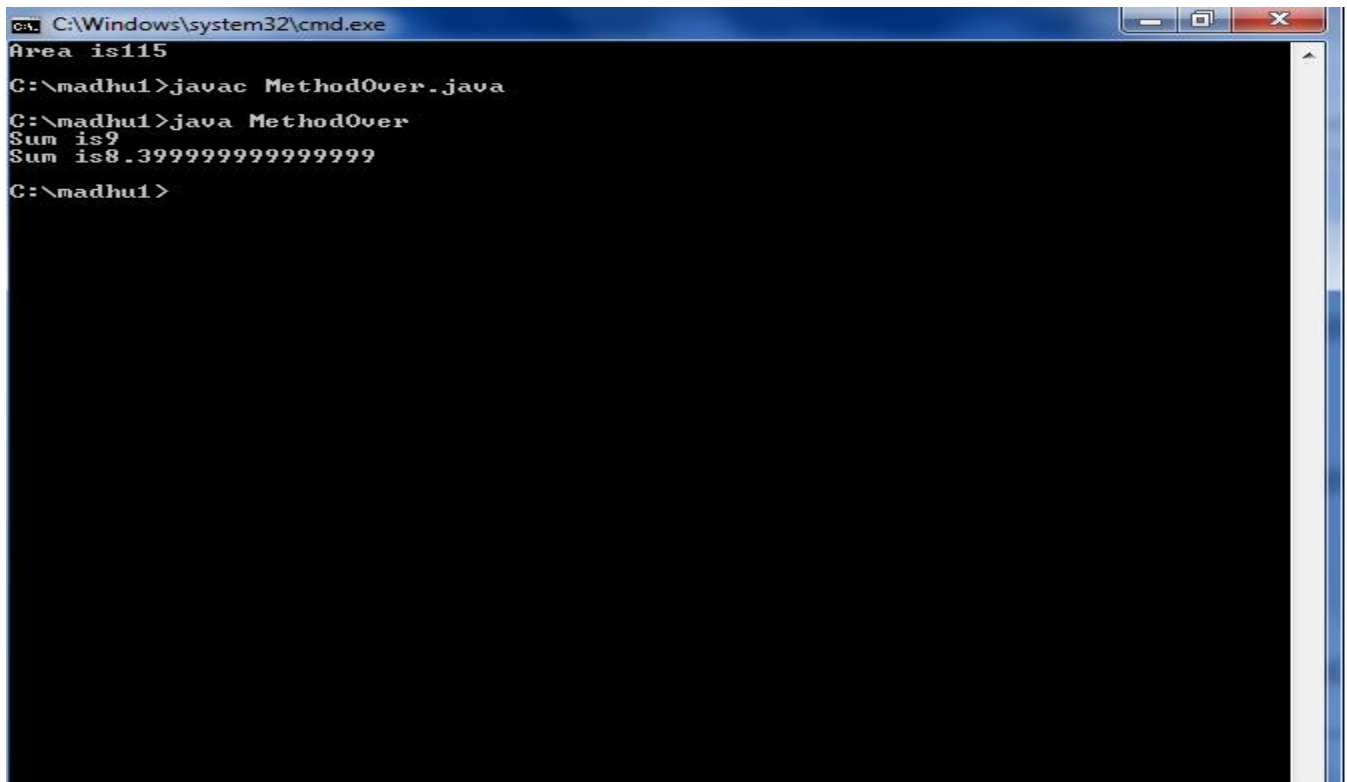
```
class Calculate
{
void sum(int a,int b)
{
System.out.println("Sum is" +(a+b));
}
void sum(double a,double b)
{
System.out.println("Sum is" +(a+b));
}
```

```

}
class MethodOver
{
public static void main(String args[])
{
Calculate a=new Calculate();
a.sum(4,5);
a.sum(4.6,3.8);
}
}

```

Output:



```

C:\Windows\system32\cmd.exe
Area is115
C:\madhu1>javac MethodOver.java
C:\madhu1>java MethodOver
Sum is9
Sum is8.3999999999999999
C:\madhu1>

```

23.write a java program to print the area of a rectangle by creating a class names “Area” taking the values of its length and breadth as parameter of its constructor and having a method names “getArea()” which returns the area of the rectangle.Length and breadth of rectangle are entered through keyboard.

Description:

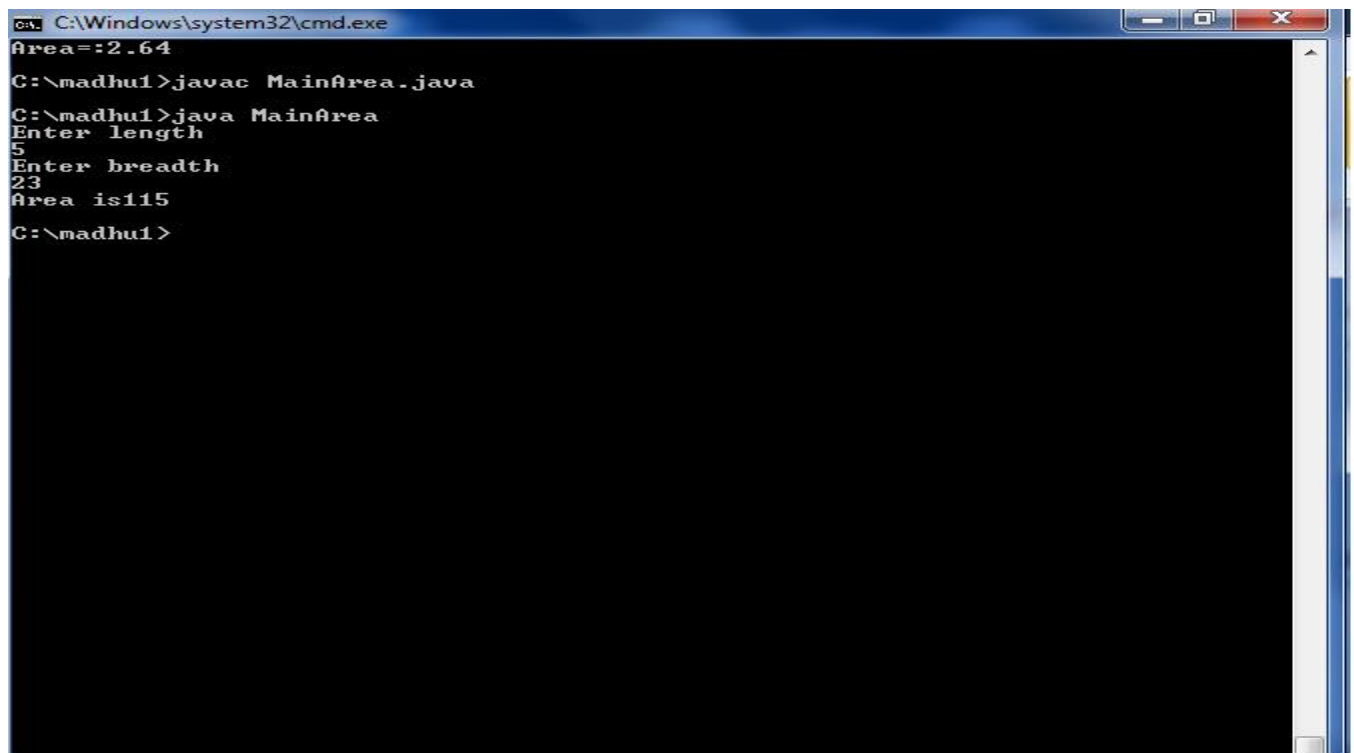
Accept the length and breadth for the user through keyboard. Create a class which has a constructor and a method to calculate the area.

Source Code:

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

```
class Area
{
int length,breadth;
Area(int a,int b)
{
this.length=a;
this.breadth=b;
}
int getArea()
{
return (length*breadth);
}
}
class MainArea
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter length");
int a=sc.nextInt();
System.out.println("Enter breadth");
int b=sc.nextInt();
Area s=new Area(a,b);
int area=s.getArea();
System.out.println("Area is"+area);
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
Area=:2.64
C:\madhu1>javac MainArea.java
C:\madhu1>java MainArea
Enter length
5
Enter breadth
23
Area is115
C:\madhu1>
```

EXERCISE-5 (INHERITANCE)

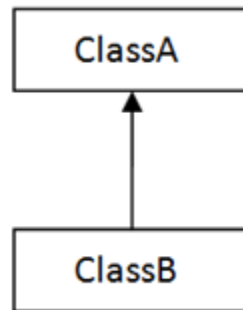
24.write a java program to implement single inheritance.

Description:

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the IS-A relationship which is also known as a *parent-child* relationship.



SINGLE INHERITANCE

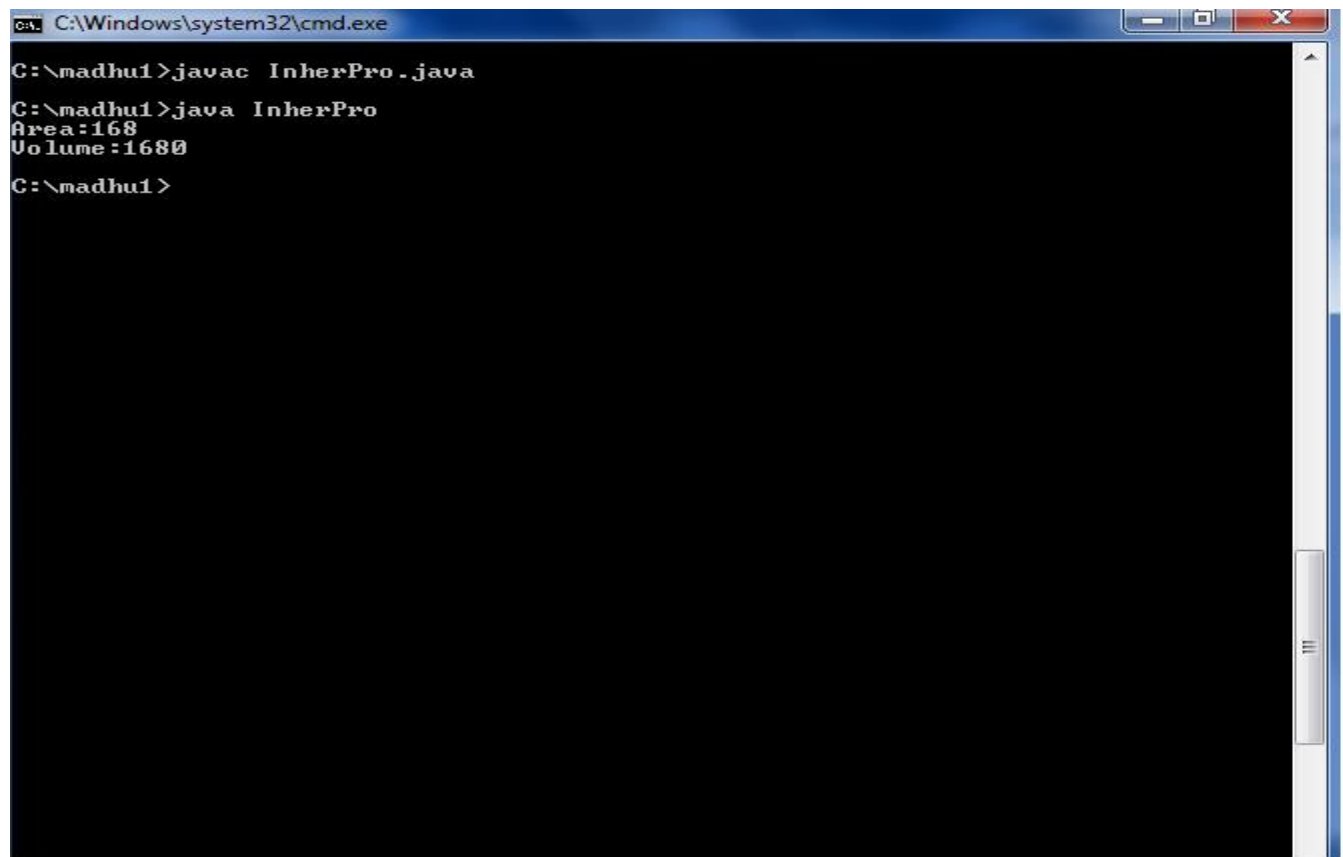
Source code:

```
class College
{
int length,breadth;
College(int x,int y)
{
this.length=x;
this.breadth=y;
}
int area()
{
return (length*breadth);
}
}
class Room extends College
{
int height;
Room(int x,int y,int z)
{
super(x,y);
height=z;
}
```



```
}  
int volume()  
{  
return (length*breadth*height);  
}  
}  
class InherPro  
{  
public static void main(String args[])  
{  
Room room=new Room(14,12,10);  
int area=room.area();  
int volume=room.volume();  
System.out.println("Area:"+area);  
System.out.println("Volume:"+volume);  
}  
}
```

Output:

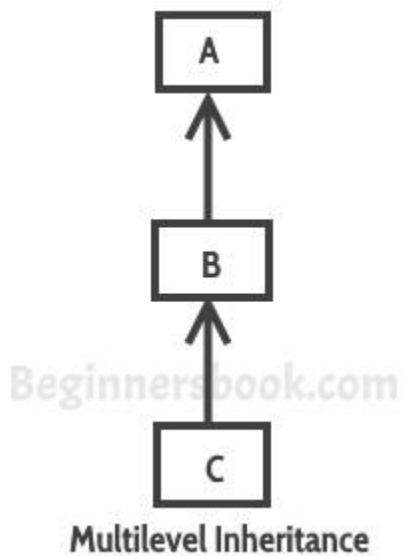


```
C:\Windows\system32\cmd.exe  
C:\madhu1>javac InherPro.java  
C:\madhu1>java InherPro  
Area:168  
Volume:1680  
C:\madhu1>
```

25. Write a java program to implement multilevel inheritance

Description:

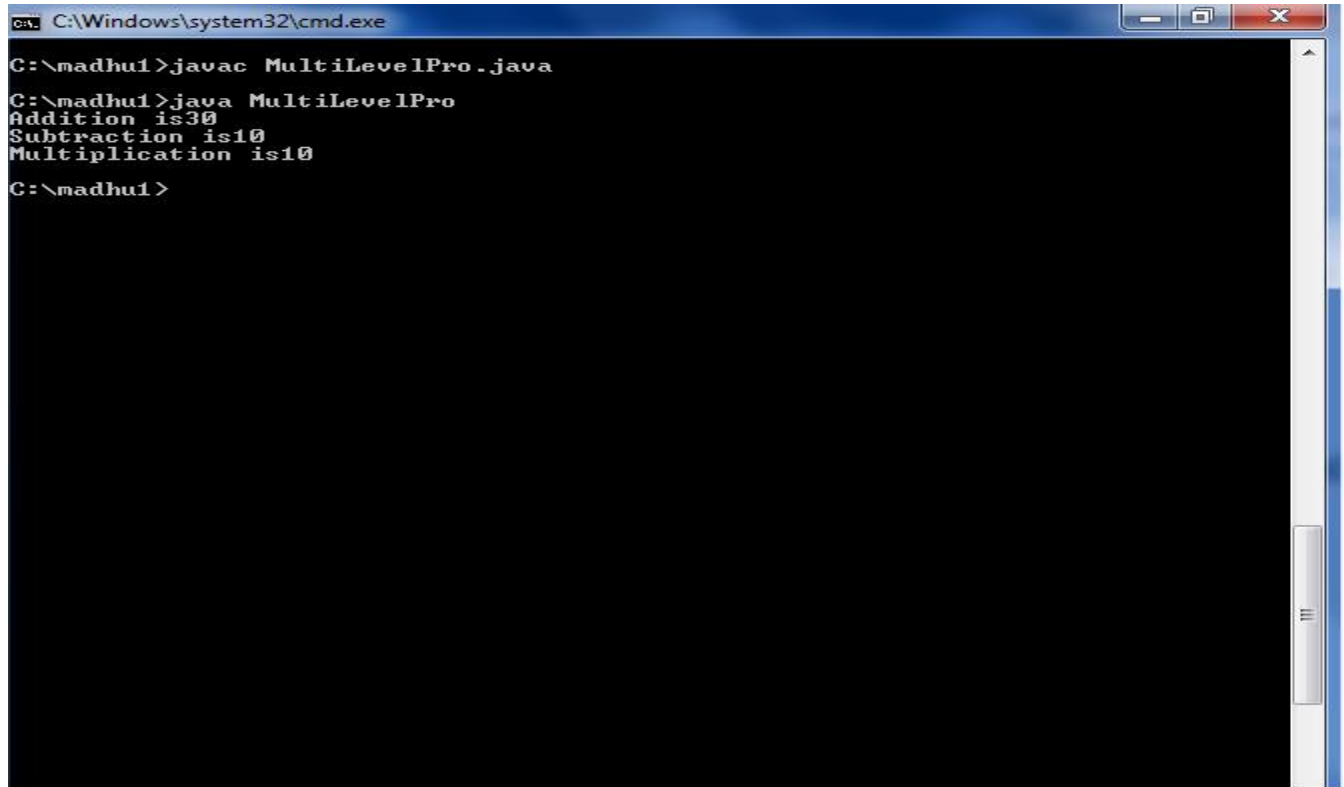
When a class extends a class, which extends another class then this is called multilevel inheritance. For example class C extends class B and class B extends class A then this type of inheritance is known as multilevel inheritance.



Source code:

```
class A
{
void add(int x,int y)
{
System.out.println("Addition is" +(x+y));
}
}
class B extends A
{
void sub(int x, int y)
{
System.out.println("Subtraction is" +(x-y));
}
}
class C extends B
{
void mul(int x,int y)
{
System.out.println("Multiplication is" +(x*y));
}
}
class MultiLevelPro
{
public static void main(String args[])
{
C obj=new C();
obj.add(10,20);
obj.sub(20,10);
obj.mul(5,2);
}
}
```

Output:



```
C:\Windows\system32\cmd.exe

C:\madhu1>javac MultiLevelPro.java
C:\madhu1>java MultiLevelPro
Addition is30
Subtraction is10
Multiplication is10
C:\madhu1>
```

26. Write a java program for abstract class to find areas of different shapes.

Description:

A class which is declared with the abstract keyword is known as an abstract class in [Java](#). It can have abstract and non-abstract methods (method with the body).

Points to Remember

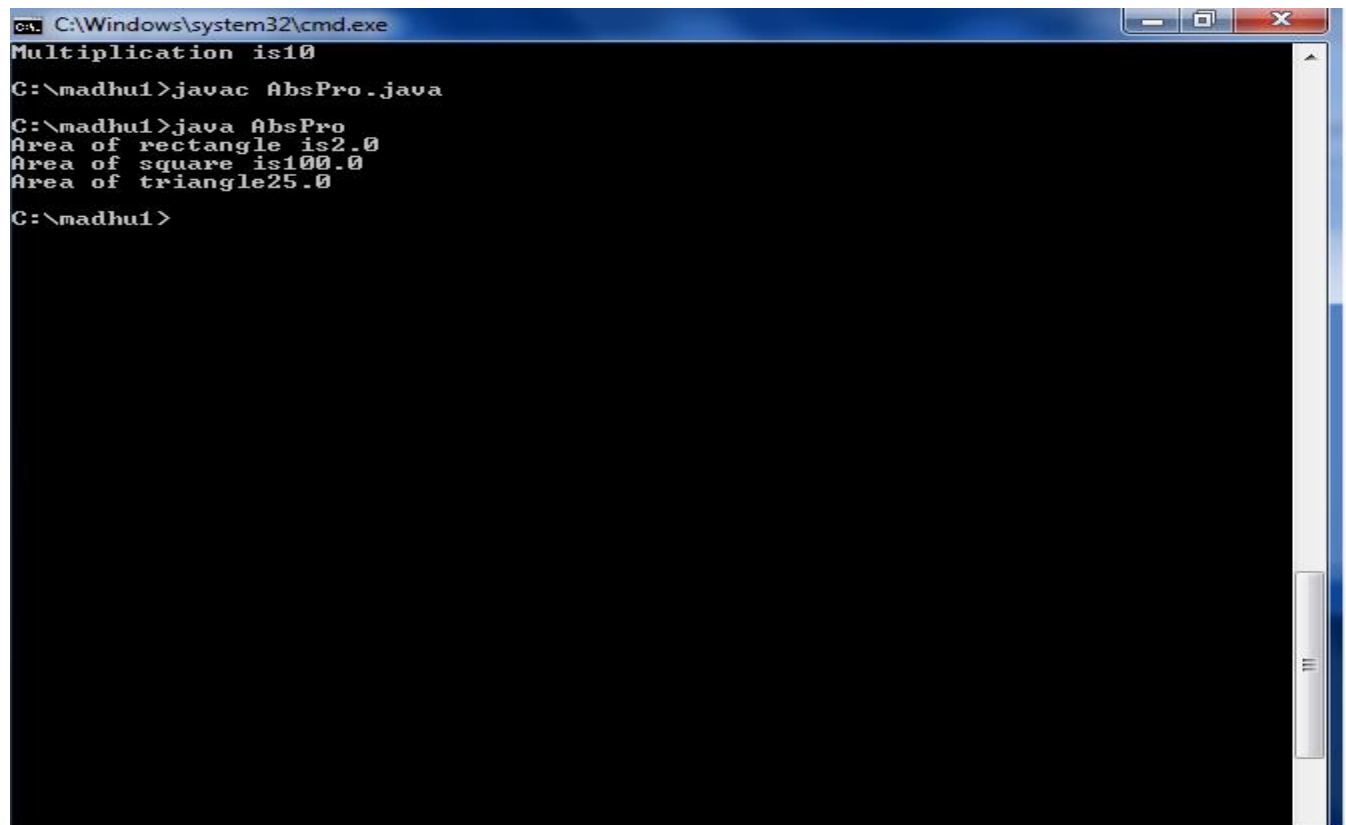
- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Source code:

```
interface Area
{
public void rectangle(float x, float y);
public void square(float x);
public void triangle(float x, float y);
}
abstract class FirstClass implements Area
{
public void rectangle(float x,float y)
{
```

```
System.out.println("Area of rectangle is" +(x*y));
}
public void square(float x)
{
System.out.println("Area of square is" +(x*x));
}
}
class SecondClass extends FirstClass
{
public void triangle(float x,float y)
{
System.out.println("Area of triangle" +(0.5*x*y));
}
}
class AbsPro
{
public static void main(String args[])
{
Area f=new SecondClass();
f.rectangle(1,2);
f.square(10);
f.triangle(10,5);
}
}
```

Output:



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The user has entered the following commands and received the corresponding output:

```
C:\Windows\system32\cmd.exe
Multiplication is10
C:\madhu1>javac AbsPro.java
C:\madhu1>java AbsPro
Area of rectangle is2.0
Area of square is100.0
Area of triangle25.0
C:\madhu1>
```

27.write a java program to demonstrate the usages of instance variables, static and local variables in the “Demo” class.

Description:

Static variable in Java is variable which belongs to the class and initialized only once at the start of the execution.

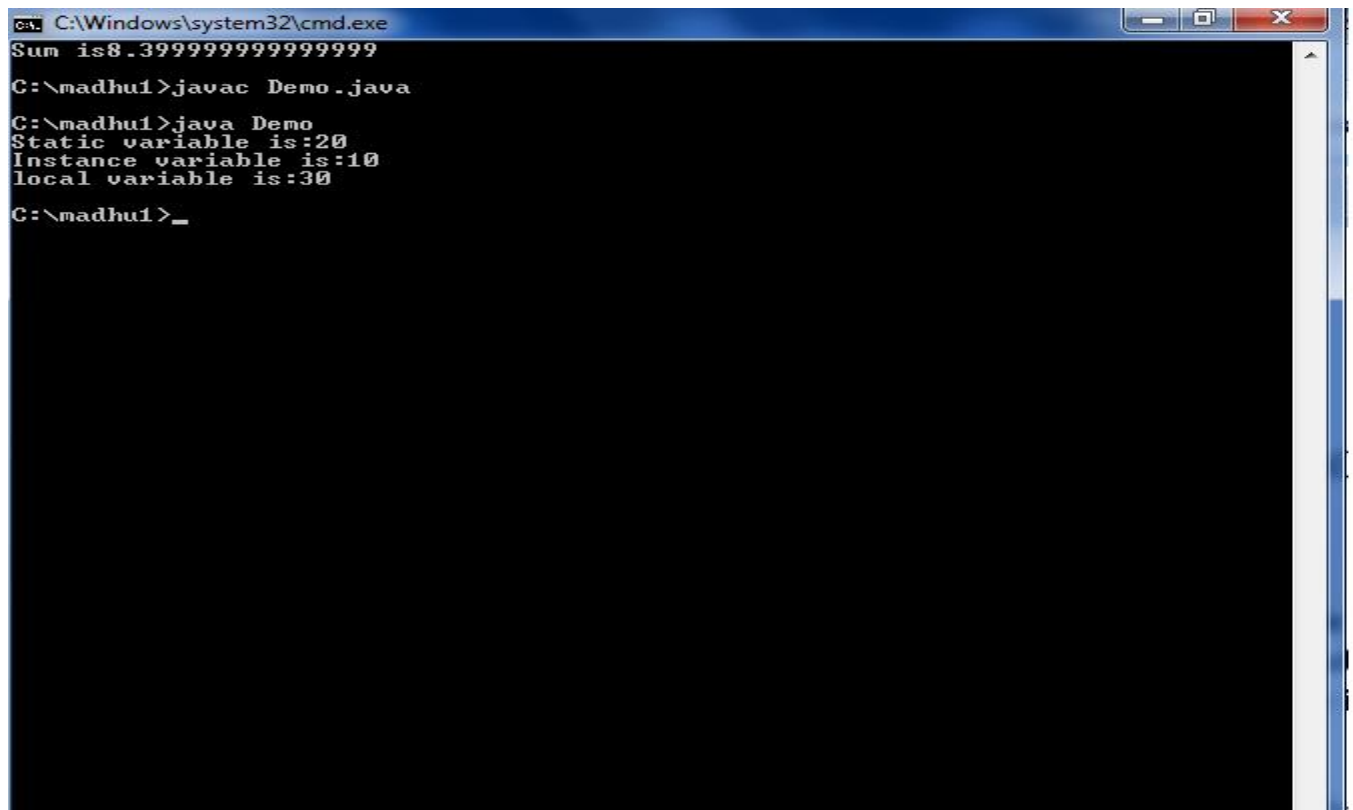
Instance variable in Java is used by Objects to store their states. Variables which are defined without the STATIC keyword and are *Outside any method declaration* are Object-specific and are known as instance variables. They are called so because their values are instance specific and are *not* shared among instances.

A *local variable* in Java is a variable that's declared within the body of a method. Then you can use the variable only within that method. Other methods in the class aren't even aware that the variable exists.

Source Code:

```
class Demo
{
int i=10;
static int j=20;
public static void main(String args[])
{
Demo d=new Demo();
int k=30;
System.out.println("Static variable is:"+Demo.j);
System.out.println("Instance variable is:"+d.i);
System.out.println("local variable is:"+k);
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
Sum is8.399999999999999
C:\madhu1>javac Demo.java
C:\madhu1>java Demo
Static variable is:20
Instance variable is:10
local variable is:30
C:\madhu1>_
```

28.write a java program to accepts a group of numbers and returns bigger number by using method with arguments.

Description:

Accepts some numbers from user through keyboard.compare them and then print the biggest number using nested if-else.

Source Code:

```
import java.util.Scanner;
public class BiggestNumber
{
    public static void main(String[] args)
    {
        int x, y, z;
        Scanner s = new Scanner(System.in);
        SubPro s1=new SubPro();
        System.out.print("Enter the first number:");
        x = s.nextInt();
        System.out.print("Enter the second number:");
        y = s.nextInt();
        System.out.print("Enter the third number:");
        z = s.nextInt();
        s1.biggestNum(x,y,z);
    }
}
class SubPro
{
    void biggestNum(int x, int y, int z)
    {
        if(x > y && x > z)
        {
            System.out.println("Largest number is:"+x);
        }
        else if(y > z)
        {
            System.out.println("Largest number is:"+y);
        }
        else
        {
            System.out.println("Largest number is:"+z);
        }
    }
}
```

Output:

```
C:\Windows\system32\cmd.exe
local variable is:30
C:\madhu1>javac BiggestNumber.java
C:\madhu1>java BiggestNumber
Enter the first number:45
Enter the second number:2
Enter the third number:33
Largest number is:45
C:\madhu1>
```

29.write a java program to test whether a static method can directly access the static variable or non static variable .

Description:

Static method only access static variables only. It doesn't access instance variables.

Source Code:

```
class Sample2
{
static int i=10;
int j=20;
public void m2()
{
System.out.println("Instance method is called");
System.out.println("Static variable is:"+i);
System.out.println("Instance variable is:"+j);
}
public static void main(String args[])
{
Sample2.m2();
System.out.println("Static variable is:"+i);
Sample2 s=new Sample2();
System.out.println("Instance variable is:"+j);
}
}
```

Output:

```
C:\Windows\system32\cmd.exe
C:\Users\S>cd\
C:\>cd madhu1
C:\madhu1>javac SamplePro.java
SamplePro.java:13: non-static method m2() cannot be referenced from a static context
SamplePro.m2();
SamplePro.java:16: non-static variable j cannot be referenced from a static context
System.out.println("Instance variable is:"+j);
2 errors
C:\madhu1>_
```

30.write a java program to test whether a instance method can directly access either instance variable or static variable.

Description:

We can access instance variable and static variables from a method called instance method.

Source Code:

```
class Sample2
{
static int i=10;
int j=20;
public void m2()
{
System.out.println("Instance method is called");
System.out.println("Static variable is:"+i);
System.out.println("Instance variable is:"+j);
}
public static void main(String args[])
{
Sample s=new Sample();
s.m2();
}
}
```


Output:

```
C:\Windows\system32\cmd.exe
C:\madhu1>javac Sample2.java
C:\madhu1>java Sample2
Instance method is called
Static variable is:10
Instance variable is:20
C:\madhu1>
```

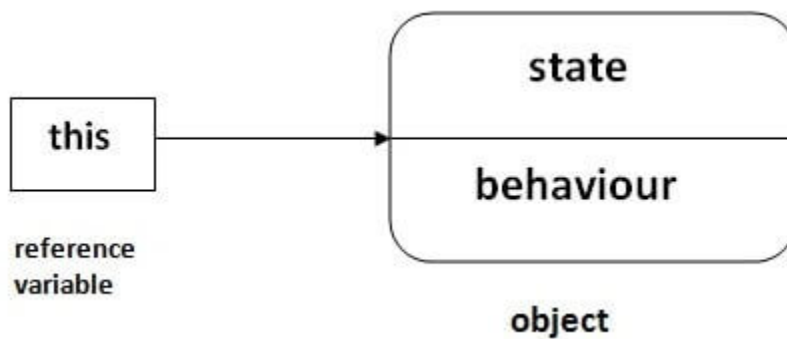
EXERCISE-6 (INHERITANCE CONTINUED)

31. Write a java program by using “this” keyword and “super” keyword.

Description:

This keyword:

There can be a lot of usage of java this keyword. In java, this is a **reference variable** that refers to the current object.



Here is given the 6 usage of java this keyword.

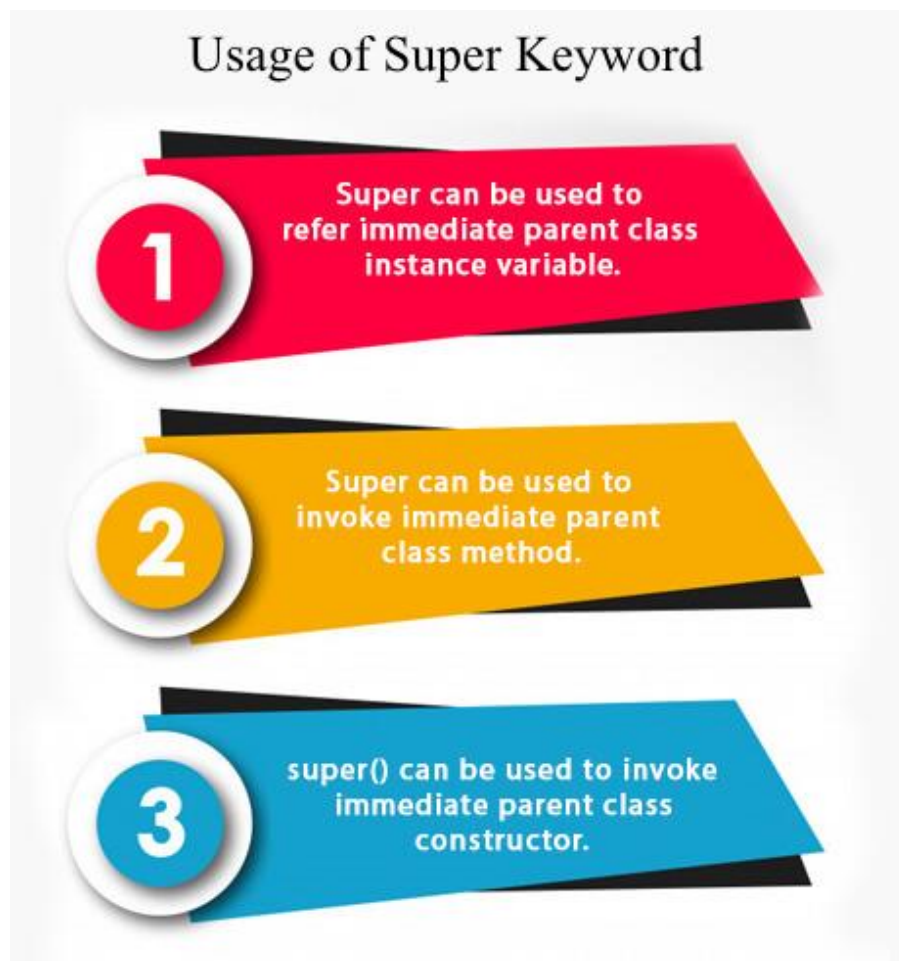
1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

Super Keyword in Java

The super keyword in Java is a reference variable which is used to refer immediate parent class object. Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of Java super Keyword

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.



Source code:

```
class Sample
{
int x;
Sample(int x)
{
this.x=x;
}
void display2()
{
System.out.println("Super class variable x value which is passed from sub class through super
keyword is"+x);
}
}
class Sub extends Sample
{
int y;
Sub(int x ,int y)
{
super(x);
this.y=y;
}
void display()
{
System.out.println("sub class variable x value is "+x);
System.out.println("sub class variable y value is "+y);
}
}
class KeyPro
{
public static void main(String args[])
{
Sub s=new Sub(100,200);
s.display();
s.display2();
}
}
```

Output:

```
C:\Windows\system32\cmd.exe
18 12 6
C:\madhu1>javac KeyPro.java
C:\madhu1>java KeyPro
sub class variable x value is 100
sub class variable y value is 200
Super class variable x value which is passed from sub class through super keyword is 100
C:\madhu1>_
```

32. Write a java program to implement interfaces. What kind of inheritance can be achieved?

Description:

The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

There are mainly three reasons to use interface. They are given below.

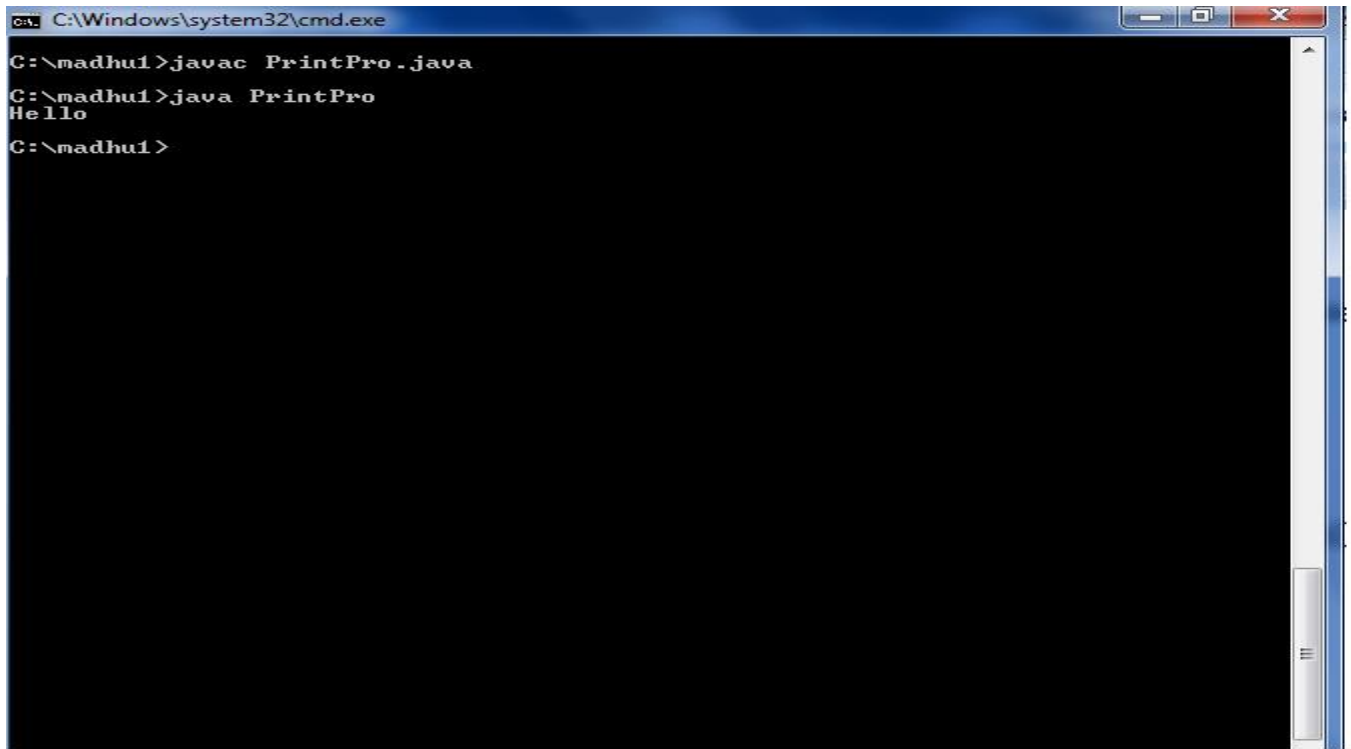
- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

Source code:

```
interface printable
{
void print();
}
class PrintPro implements printable
{
public void print()
{
System.out.println("Hello");
}
public static void main(String args[])
{
```

```
PrintPro p=new PrintPro();  
p.print();  
}  
}
```

Output:

A screenshot of a Windows command prompt window. The title bar reads 'C:\Windows\system32\cmd.exe'. The command prompt shows the following sequence of commands and output:
C:\madhu1>javac PrintPro.java
C:\madhu1>java PrintPro
Hello
C:\madhu1>
The background of the command prompt is black, and the text is white.

33. write a java program for command line arguments.

Description:

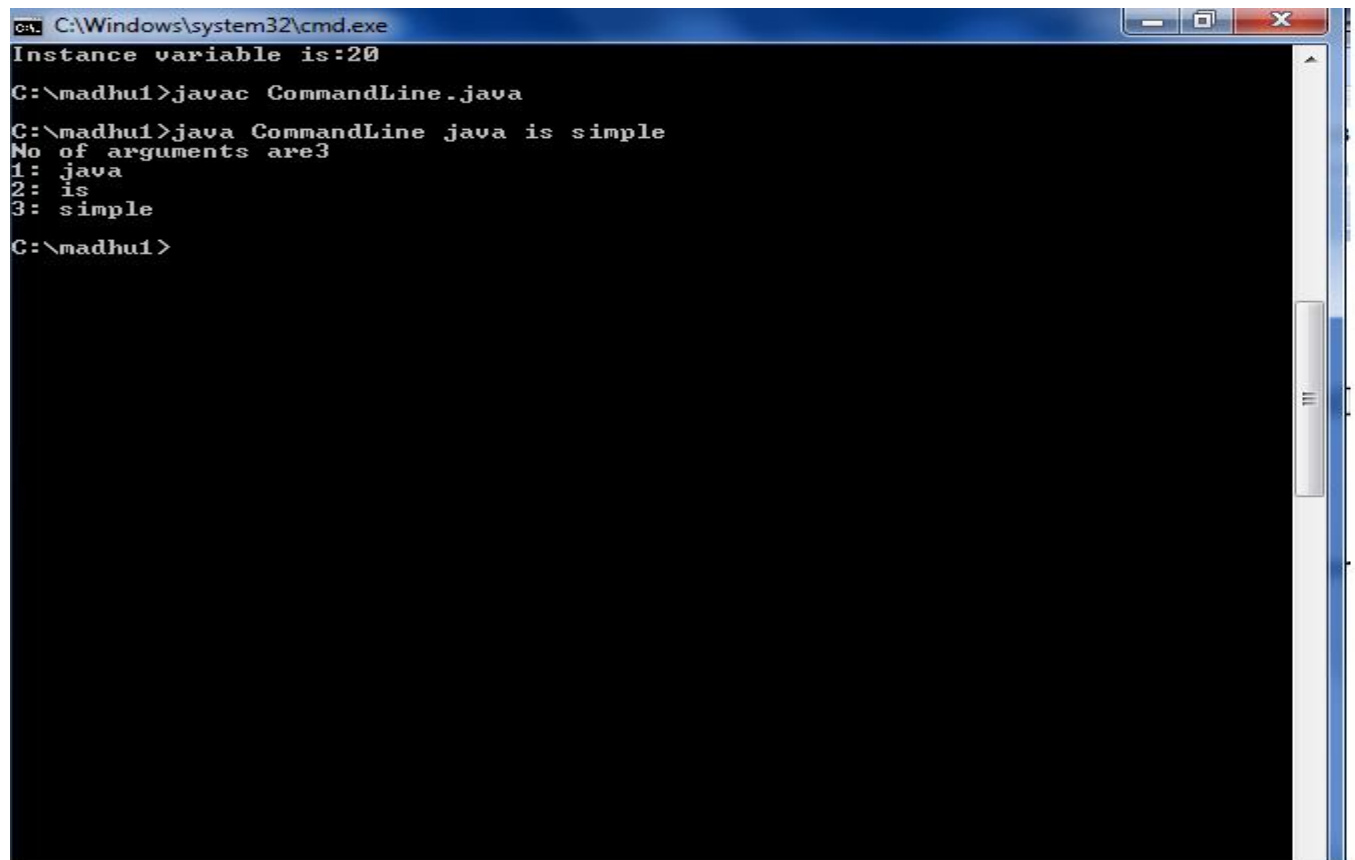
The java command-line argument is an argument i.e. passed at the time of running the java program. The arguments passed from the console can be received in the java program and it can be used as an input.

Source Code:

```
class CommandLine  
{  
public static void main(String args[])  
{  
int i=0,count;  
count=args.length;  
System.out.println("No of arguments are"+count);  
String s;  
while(i<count)  
{  
s=args[i];  
i=i+1;  
System.out.println(i+": "+" "+s);  
}  
}
```

```
}  
}
```

Output:



```
C:\Windows\system32\cmd.exe
Instance variable is:20
C:\madhu1>javac CommandLine.java
C:\madhu1>java CommandLine java is simple
No of arguments are3
1: java
2: is
3: simple
C:\madhu1>
```

34. write a java programs for (a)matrix addition (b)matrix multiplication.

Description:

We can add two matrices in java using binary + operator. A matrix is also known as array of arrays. We can add, subtract and multiply matrices.

$$\text{Matrix 1} \begin{Bmatrix} 1 & 3 & 4 \\ 2 & 4 & 3 \\ 3 & 4 & 5 \end{Bmatrix} \quad \text{Matrix 2} \begin{Bmatrix} 1 & 3 & 4 \\ 2 & 4 & 3 \\ 1 & 2 & 4 \end{Bmatrix}$$

$$\begin{matrix} \text{Matrix 1} \\ + \\ \text{Matrix 2} \end{matrix} \begin{Bmatrix} 1+1 & 3+3 & 4+4 \\ 2+2 & 4+4 & 3+3 \\ 3+1 & 4+2 & 5+4 \end{Bmatrix}$$

$$\begin{matrix} \text{Matrix 1} \\ + \\ \text{Matrix 2} \end{matrix} \begin{Bmatrix} 2 & 6 & 8 \\ 4 & 8 & 6 \\ 4 & 6 & 9 \end{Bmatrix}$$

Source code:

```
import java.util.*;
class MatAddition
{
public static void main(String args[])
{
int a[][]=new int [10][10];
int b[][]=new int [10][10];
int c[][]=new int [20][20];
int m,n,i,j;
Scanner sc=new Scanner(System.in);
System.out.println("Enter how many rows in  matrix");
m=sc.nextInt();
System.out.println("Enter how many columns in  matrix");
n=sc.nextInt();
System.out.println("Enter first matrix elements");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
a[i][j]=sc.nextInt();
}
System.out.println(" ");
}
System.out.println("Enter second matrix elements");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
b[i][j]=sc.nextInt();
}
System.out.println(" ");
}
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
c[i][j]=a[i][j]+b[i][j];
}
}

System.out.println("Addition of matrices is");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
System.out.print(" "+c[i][j]);
}
System.out.println("");
}
}
}
```

Output:

```
C:\Windows\system32\cmd.exe
C:\madhu1>javac MatAddition.java
C:\madhu1>java MatAddition
Enter how many rows in matrix
3
Enter how many columns in matrix
3
Enter first matrix elements
1 2 3
1 2 3
1 2 3
Enter second matrix elements
3 2 1
3 2 1
3 2 1
Addition of matrices is
4 4 4
4 4 4
4 4 4
C:\madhu1>_
```

(B) Description:

Java Program to multiply two matrices

We can multiply two matrices in java using binary * operator and executing another loop. A matrix is also known as array of arrays. We can add, subtract and multiply matrices. In case of matrix multiplication, one row element of first matrix is multiplied by all columns of second matrix.

$$\text{Matrix 1} \begin{Bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{Bmatrix} \quad \text{Matrix 2} \begin{Bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{Bmatrix}$$

$$\begin{matrix} \text{Matrix 1} \\ * \\ \text{Matrix 2} \end{matrix} \begin{Bmatrix} 1*1+1*2+1*3 & 1*1+1*2+1*3 & 1*1+1*2+1*3 \\ 2*1+2*2+2*3 & 2*1+2*2+2*3 & 2*1+2*2+2*3 \\ 3*1+3*2+3*3 & 3*1+3*2+3*3 & 3*1+3*2+3*3 \end{Bmatrix}$$

$$\begin{matrix} \text{Matrix 1} \\ * \\ \text{Matrix 2} \end{matrix} \begin{Bmatrix} 6 & 6 & 6 \\ 12 & 12 & 12 \\ 18 & 18 & 18 \end{Bmatrix}$$

Source code:

```
import java.util.*;
class MatMul
{
public static void main(String args[])
{
int a[][]=new int [10][10];
int b[][]=new int [10][10];
int c[][]=new int [20][20];
int m,n,i,j,k,q;
Scanner sc=new Scanner(System.in);
System.out.println("Enter how many rows in first matrix");
m=sc.nextInt();
System.out.println("Enter how many columns in first matrix");
n=sc.nextInt();
System.out.println("Enter how many rows in second matrix");
n=sc.nextInt();
System.out.println("Enter how many columns in second matrix");
q=sc.nextInt();
System.out.println("Enter first matrix elements");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
a[i][j]=sc.nextInt();
}
System.out.println(" ");
}
System.out.println("Enter second matrix elements");
for(i=0;i<n;i++)
{
for(j=0;j<q;j++)
{
b[i][j]=sc.nextInt();
}
System.out.println(" ");
}

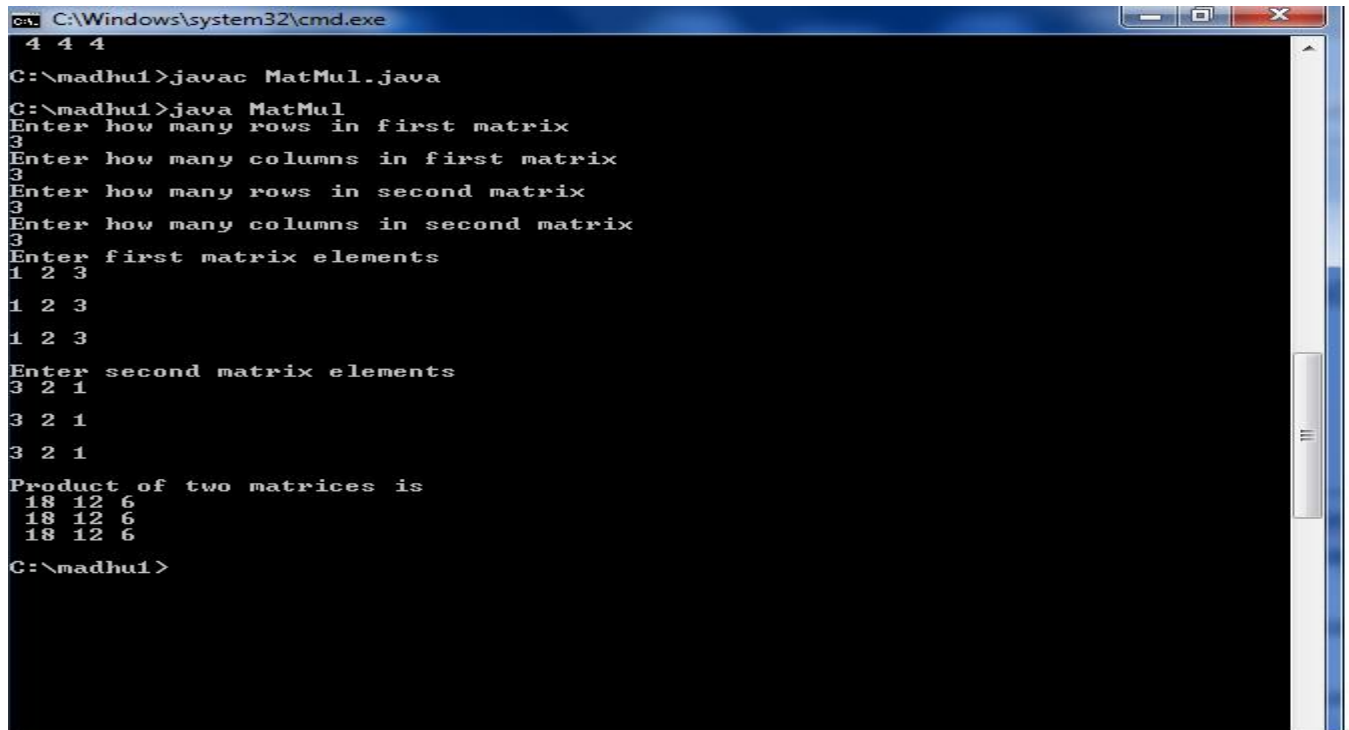
for(i=0;i<m;i++)
{
for(j=0;j<q;j++)
{
c[i][j]=0;
for(k=0;k<n;k++)
{
c[i][j]=c[i][j]+a[i][k]*b[k][j];
}
}
}
System.out.println("Product of two matrices is");
for(i=0;i<m;i++)
{
for(j=0;j<q;j++)
```

```

{
System.out.print(" "+c[i][j]);
}
System.out.println("");
}
}
}

```

Output:



```

C:\Windows\system32\cmd.exe
4 4 4
C:\madhu1>javac MatMul.java
C:\madhu1>java MatMul
Enter how many rows in first matrix
3
Enter how many columns in first matrix
3
Enter how many rows in second matrix
3
Enter how many columns in second matrix
3
Enter first matrix elements
1 2 3
1 2 3
1 2 3
Enter second matrix elements
3 2 1
3 2 1
3 2 1
Product of two matrices is
18 12 6
18 12 6
18 12 6
C:\madhu1>

```

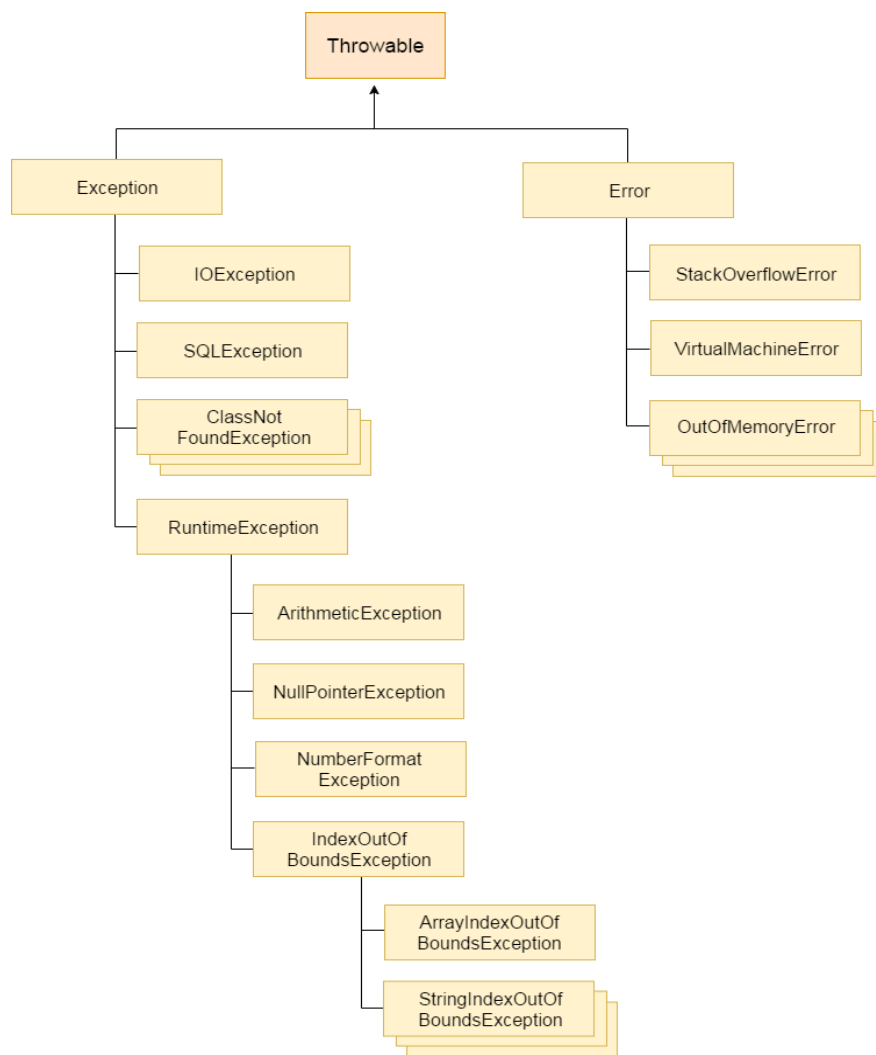
EXERCISE-7 (EXCEPTION)

35.write a java program that describes exception handling mechanism.

Description:

The Exception Handling in Java is one of the powerful *mechanism to handle the runtime errors* so that normal flow of the application can be maintained

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.



Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. Here, an error is considered as the unchecked exception. According to Oracle, there are three types of exceptions:

1. Checked Exception
2. Unchecked Exception
3. Error
4. 1) Checked Exception
5. The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.
6. 2) Unchecked Exception
7. The classes which inherit RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.
8. 3) Error
9. Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

Source code:

```

class ExceptionHandlingPro
{

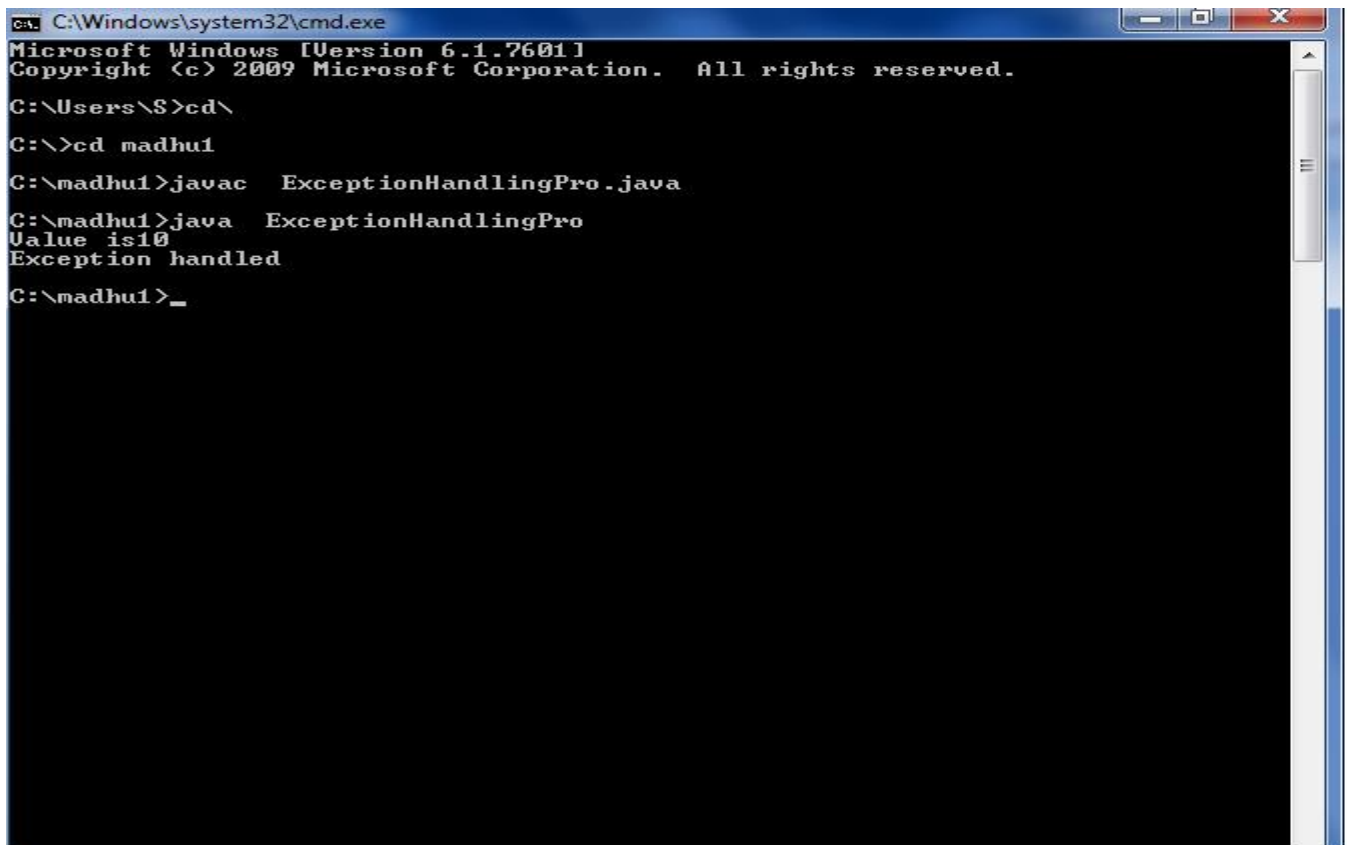
```

```

public static void main(String args[])
{
int x=20;
int y=0;
int z;
try
{
z=x/y;
System.out.println("Value is" +z);
}
catch(Exception e)
{
y=2;
z=x/y;
System.out.println("Value is" +z);
System.out.println("Exception handled");
}
}
}

```

Output:



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\S>cd\
C:\>cd madhu1
C:\madhu1>javac ExceptionHandlingPro.java
C:\madhu1>java ExceptionHandlingPro
Value is10
Exception handled
C:\madhu1>_

```

36. write a java program illustrating multi catch clauses.

Description:

A try block can be followed by one or more catch blocks. Each catch block must contain a different exception handler. So, if you have to perform different tasks at the occurrence of different exceptions, use java multi-catch block.

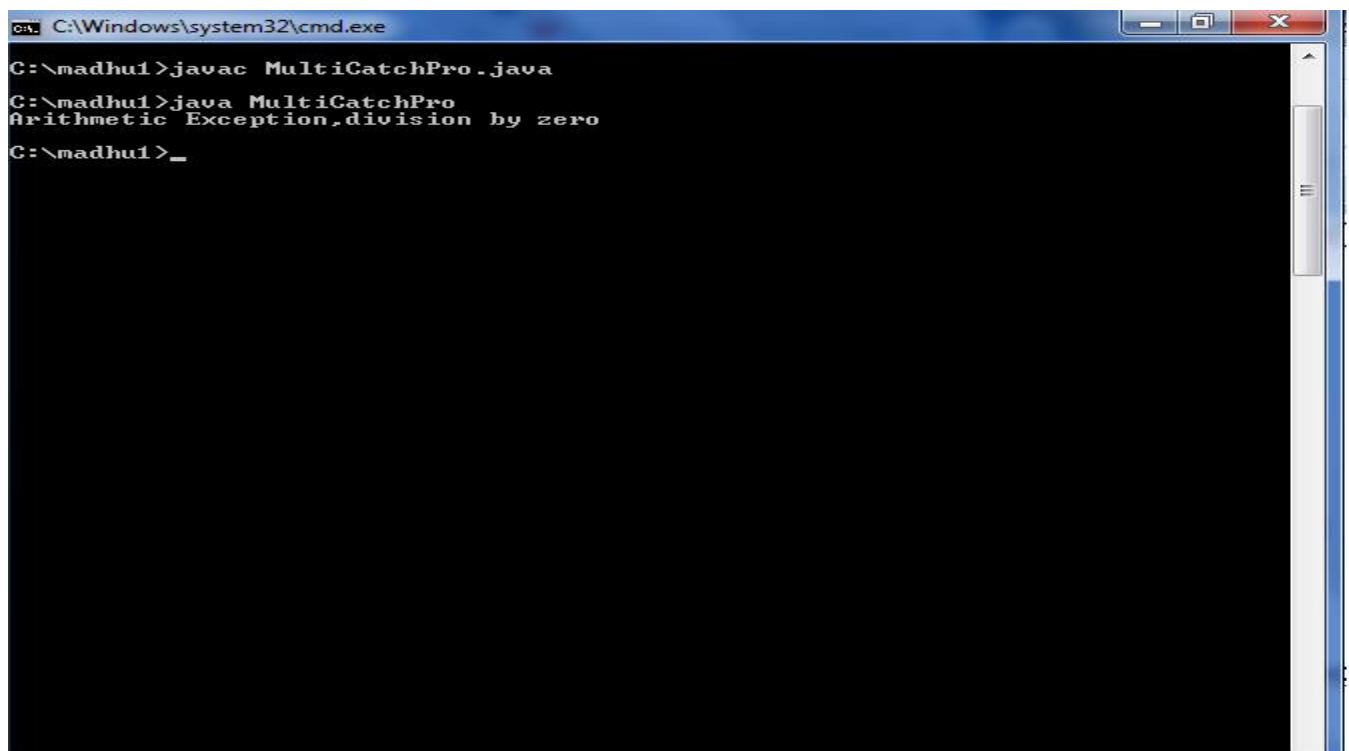
Points to remember

- At a time only one exception occurs and at a time only one catch block is executed.
- All catch blocks must be ordered from most specific to most general, i.e. catch for `ArithmeticException` must come before catch for `Exception`.

Source code:

```
class MultiCatchPro
{
public static void main(String args[])
{
int x=20;
int y=0;
int z;
try
{
z=x/y;
System.out.println("Value is" +z);
}
catch(ArithmeticException e)
{
System.out.println("Arithmetic Exception,division by zero");
}
catch(Exception e)
{
System.out.println("An exception occurs");
}
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
C:\madhu1>javac MultiCatchPro.java
C:\madhu1>java MultiCatchPro
Arithmetic Exception,division by zero
C:\madhu1>_
```

37. Write a java program to apply “final” keyword to variables, methods and classes.

Description:

The final keyword in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

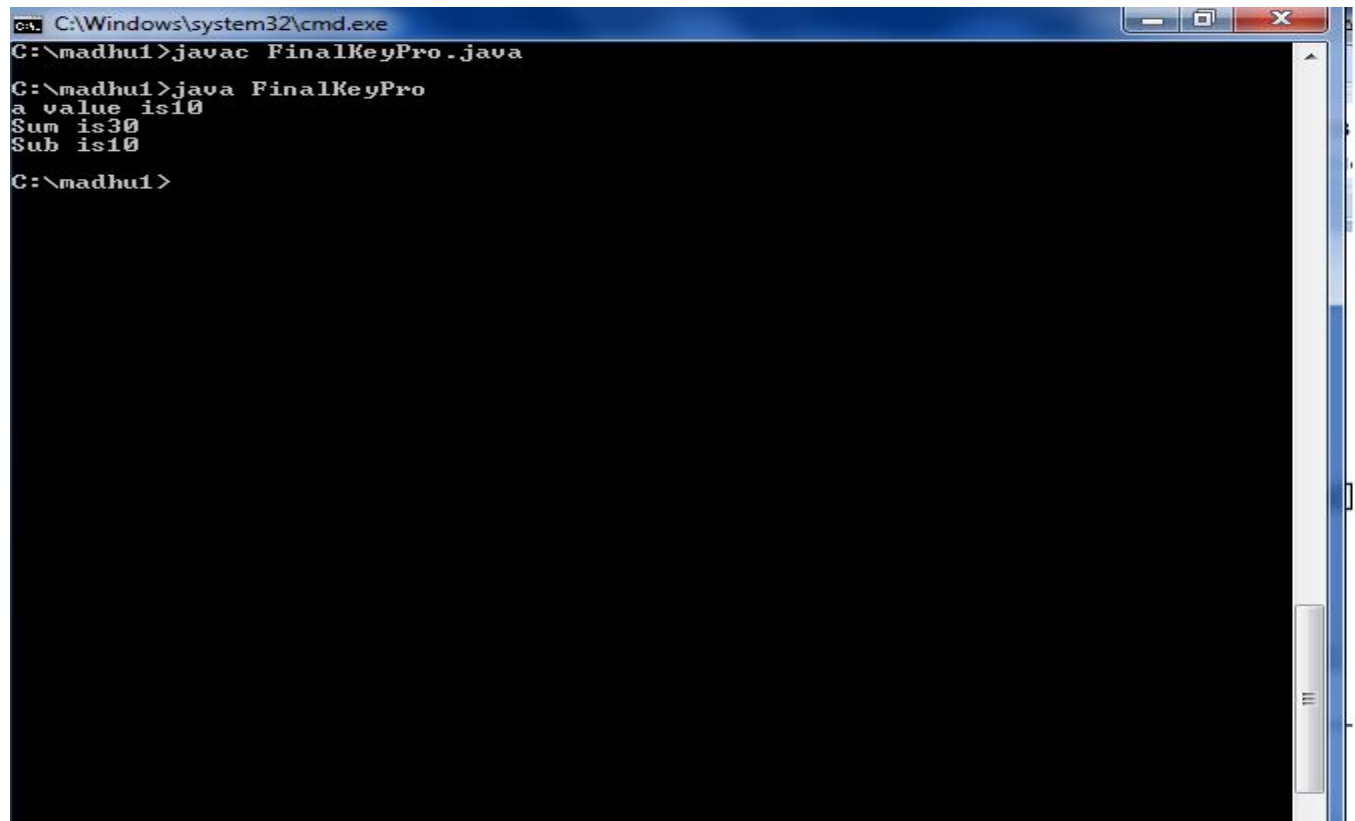
1. variable
2. method
3. class

Final variable once assigned a value can never be changed. If you make any method as final, you cannot override it. If you make any class as final, you cannot extend it.

Source code:

```
final class SuperPro
{
final void sum(int x,int y)
{
System.out.println("Sum is" +(x+y));
}
void sub(int x,int y)
{
System.out.println("Sub is" +(x-y));
}
}
class FinalKeyPro
{
public static void main(String args[])
{
final int a=10;
//a=20;
System.out.println("a value is" +a);
SuperPro s=new SuperPro();
s.sum(10,20);
s.sub(20,10);
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
C:\madhu1>javac FinalKeyPro.java
C:\madhu1>java FinalKeyPro
a value is10
Sum is30
Sub is10
C:\madhu1>
```

EXERCISE-8 **(RUNTIME POLYMORPHISM)**

38. Write a java program that implements Runtime polymorphism.

Description:

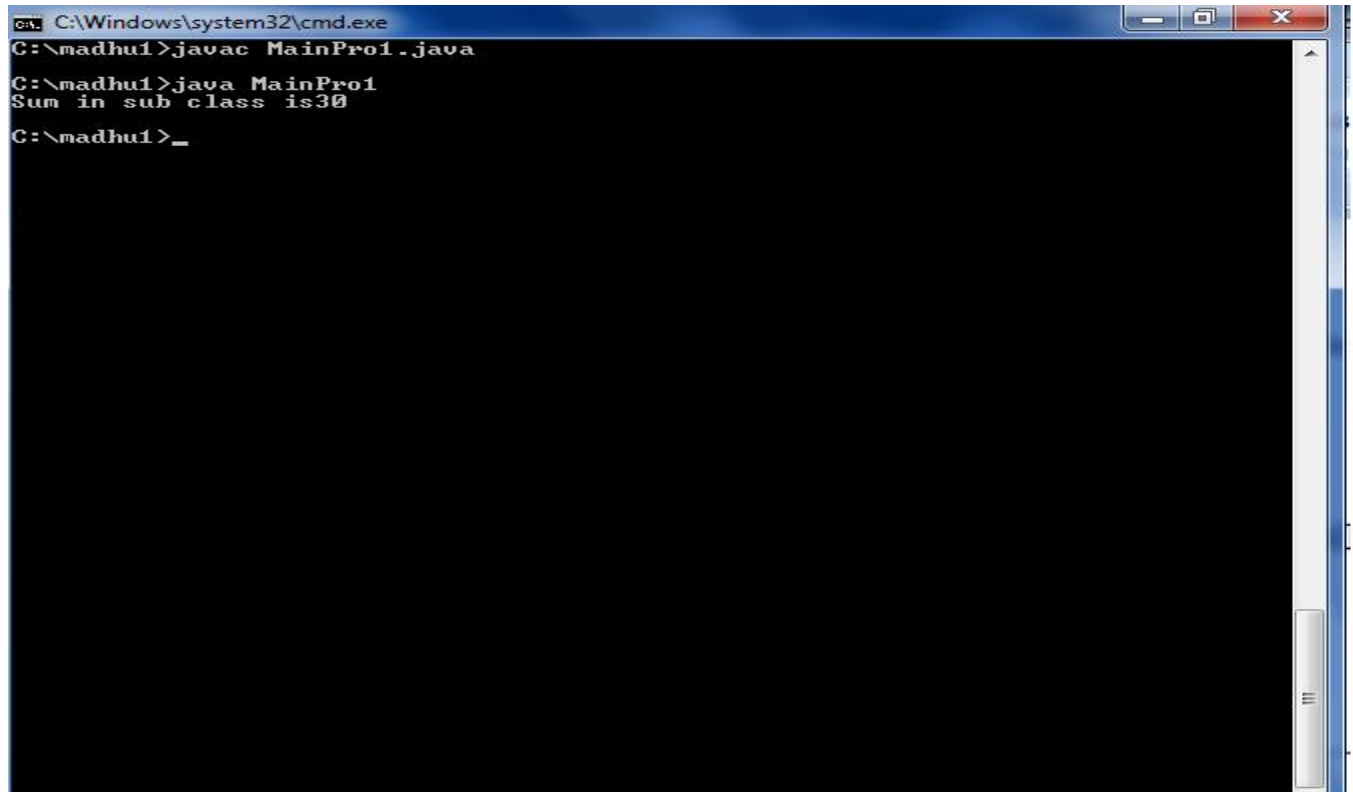
Method overriding is an example of runtime polymorphism. In method overriding, a subclass overrides a method with the same signature as that of in its superclass. During compile time, the check is made on the reference type. However, in the runtime, JVM figures out the object type and would run the method that belongs to that particular object.

Source code:

```
class A
{
public void sum(int x,int y)
{
System.out.println("Sum in super class is" +(x+y));
}
}
class B extends A
{
public void sum(int x,int y)
{
System.out.println("Sum in sub class is" +(x+y));
}
}
```

```
}  
class MainPro1  
{  
public static void main(String args[])  
{  
B obj=new B();  
obj.sum(10,20);  
}  
}
```

Output:



```
C:\Windows\system32\cmd.exe  
C:\madhu1>javac MainPro1.java  
C:\madhu1>java MainPro1  
Sum in sub class is30  
C:\madhu1>_
```

39.write a case study on runtime polymorphism, inheritance that implements in above program.

Description:

We have seen that a method defined in a super class inherited by its subclass and is used by the object created by its subclass and is used by the object created by the subclass. Method inheritance enables us to define and use methods repeatedly in subclasses without having to define the methods again in subclass.

When we want an object to respond to the same method but have different behavior when the method is called. We should override the method defined in the base class.

This is possible by defining a method in the subclass that has the same name, same arguments and same return type as a method in base class. When that method is called, the method defined in the sub class is invoked an executed instead of the one in the superclass.

EXERCISE-9

(User Defined Exceptions)

40.write a java program for creation of illustrating throw.

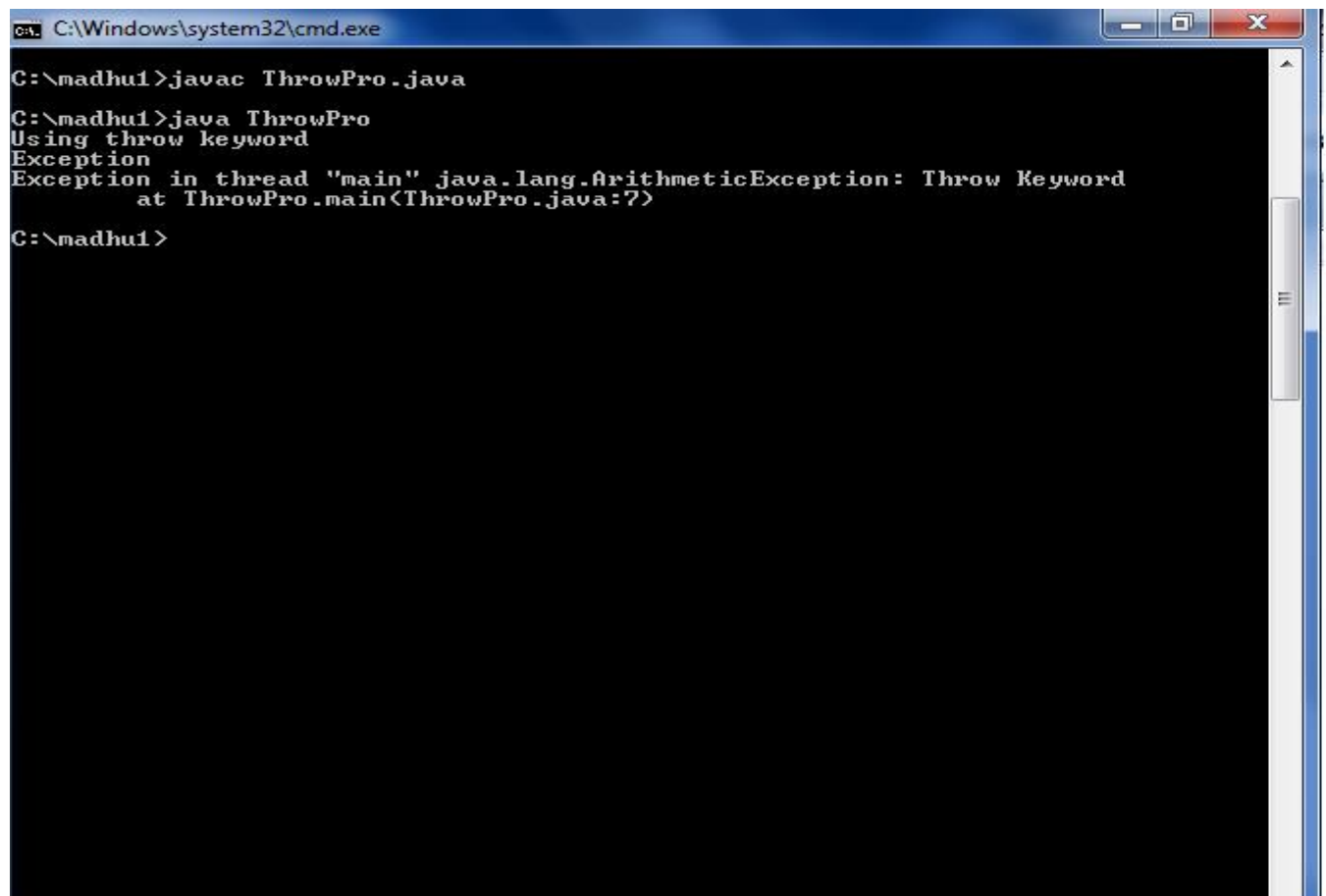
Description:

The Java throw keyword is used to explicitly throw an exception. We can throw either checked or unchecked exception in java by throw keyword. The throw keyword is mainly used to throw custom exception.

Source code:

```
class ThrowPro
{
public static void main(String args[])
{
System.out.println("Using throw keyword");
System.out.println("Exception");
throw new ArithmeticException("Throw Keyword");
}
}
```

Output:



```
C:\Windows\system32\cmd.exe

C:\madhu1>javac ThrowPro.java

C:\madhu1>java ThrowPro
Using throw keyword
Exception
Exception in thread "main" java.lang.ArithmeticException: Throw Keyword
    at ThrowPro.main<ThrowPro.java:7>

C:\madhu1>
```

41.write a java program for creation of illustrating finally.

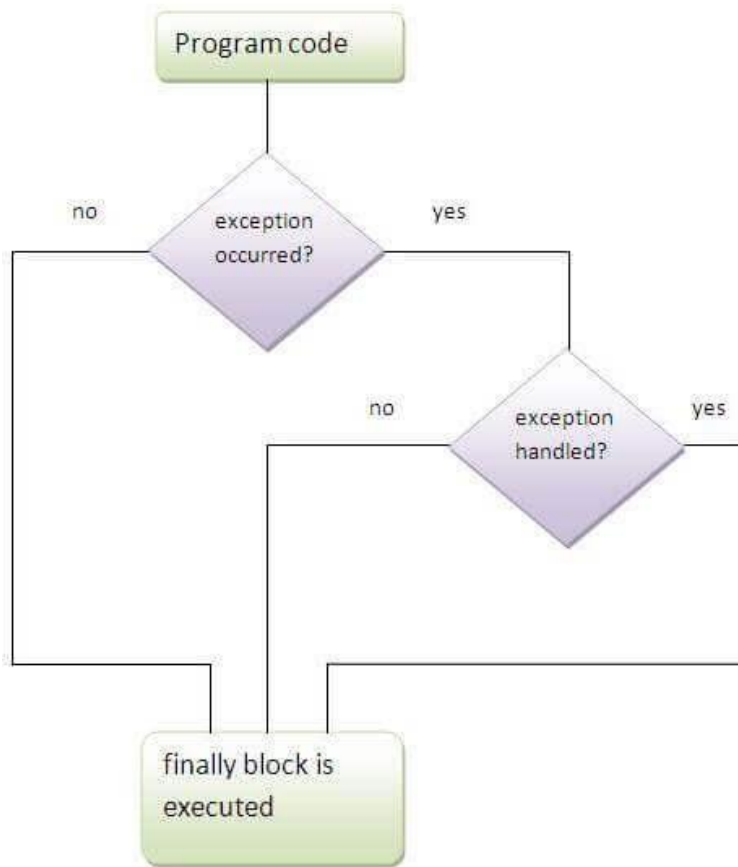
Description:

Java finally block

Java finally block is a block that is used *to execute important code* such as closing connection, stream etc.

Java finally block is always executed whether exception is handled or not.

Java finally block follows try or catch block.

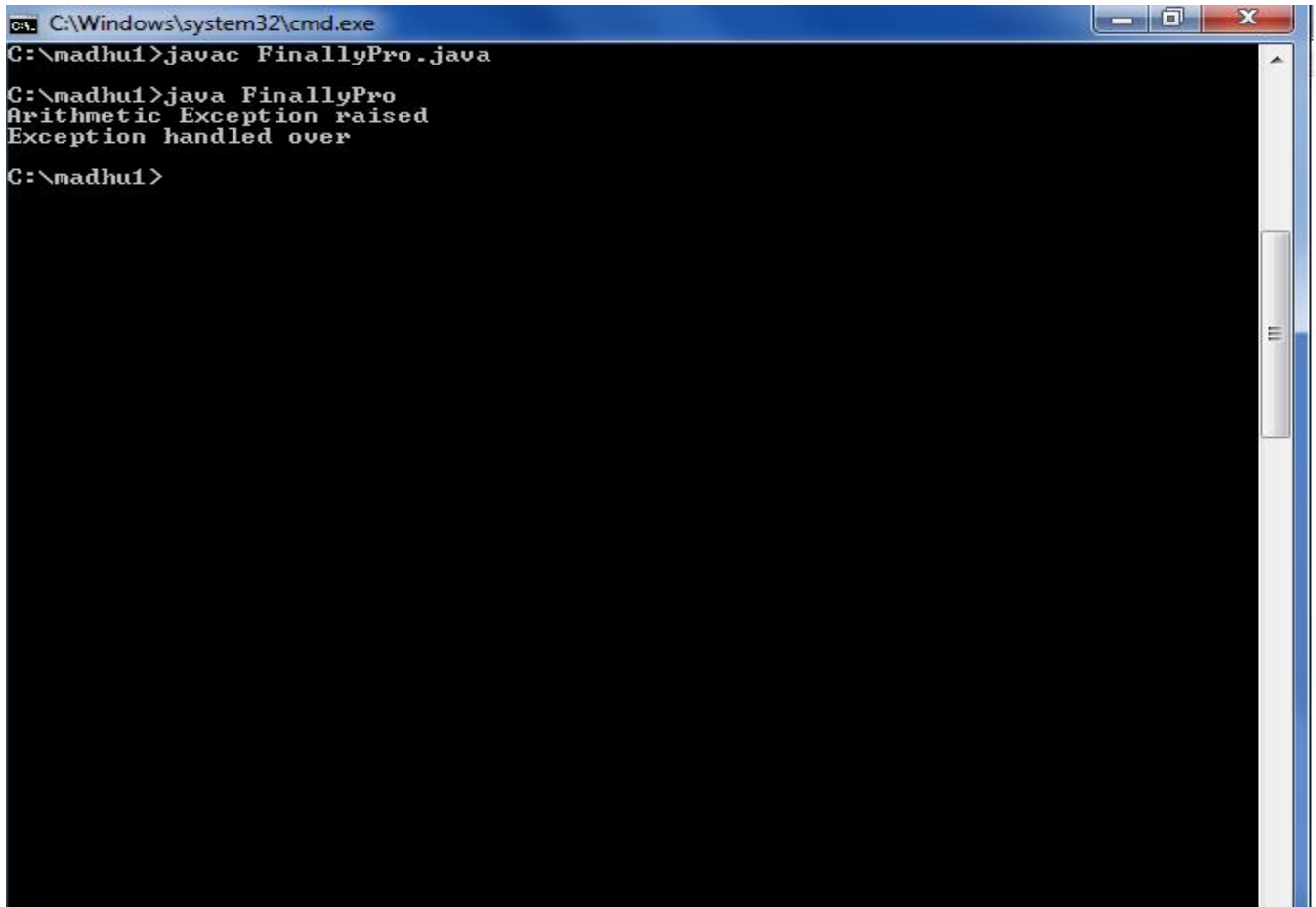


Source code:

```
class FinallyPro
{
public static void main(String args[])
{
int x=10;
int y=0;
try
{
int z=x/y;
System.out.println("Value is" +z);
}
catch(ArithmeticException e)
{
System.out.println("Arithmetic Exception raised");
}
```

```
}  
finally  
{  
System.out.println("Exception handled over");  
}  
}  
}
```

Output:



```
C:\Windows\system32\cmd.exe  
C:\madhu1>javac FinallyPro.java  
C:\madhu1>java FinallyPro  
Arithmetic Exception raised  
Exception handled over  
C:\madhu1>
```

42. write a java program for creation of java built-in exceptions.

Description:

Java defines several exception classes inside the standard package `java.lang`.

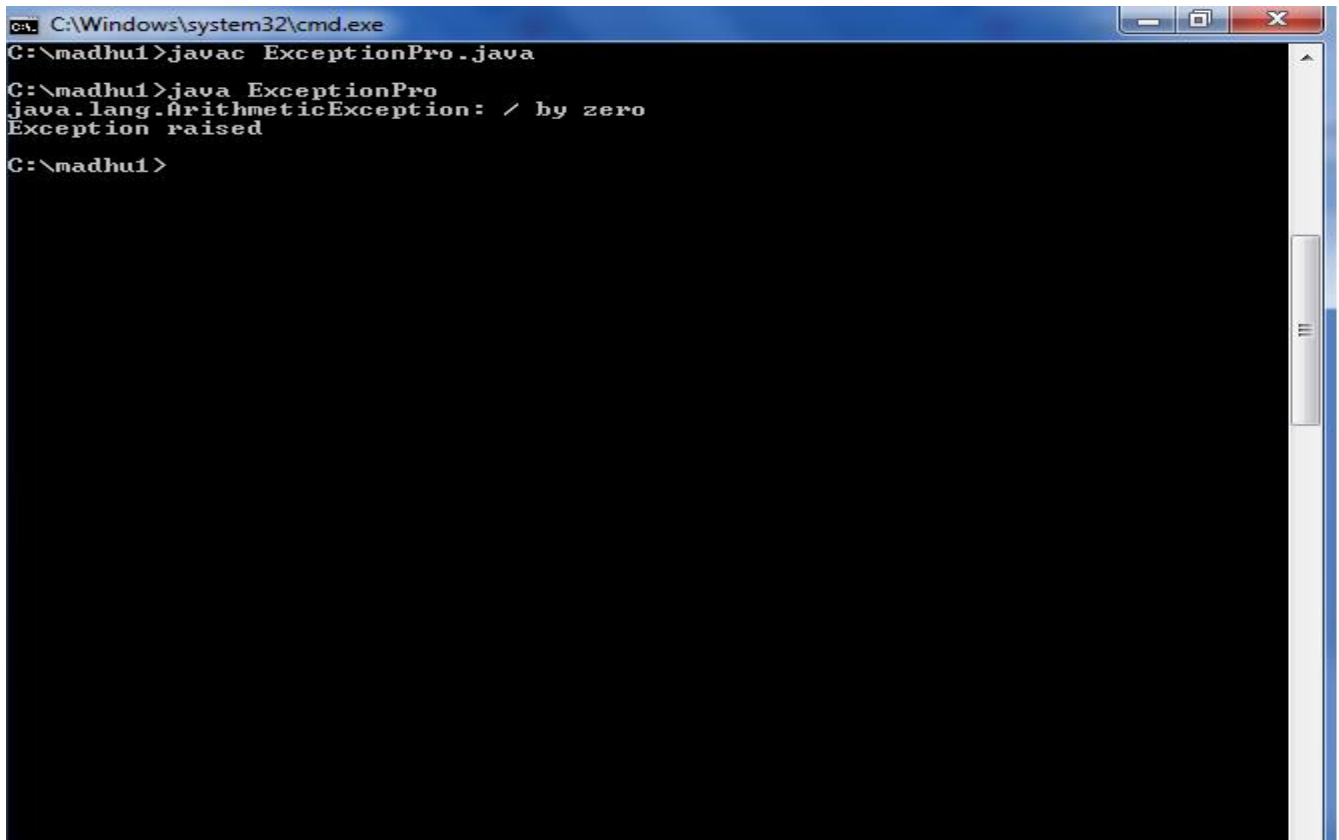
The most general of these exceptions are subclasses of the standard type `RuntimeException`. Since `java.lang` is implicitly imported into all Java programs, most exceptions derived from `RuntimeException` are automatically available.

Source code:

```
class ExceptionPro  
{  
public static void main(String args[])  
{  
int x=10;  
int y=0;
```

```
try
{
int z=x/y;
System.out.println("Value is" +z);
}
catch(Exception e)
{
System.out.println(e);
System.out.println("Exception raised");
}
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
C:\madhu1>javac ExceptionPro.java
C:\madhu1>java ExceptionPro
java.lang.ArithmeticException: / by zero
Exception raised
C:\madhu1>
```

43.write a java program for creation of user defined exceptions.

Description:

If you are creating your own Exception that is known as custom exception or user-defined exception. Java custom exceptions are used to customize the exception according to user need.

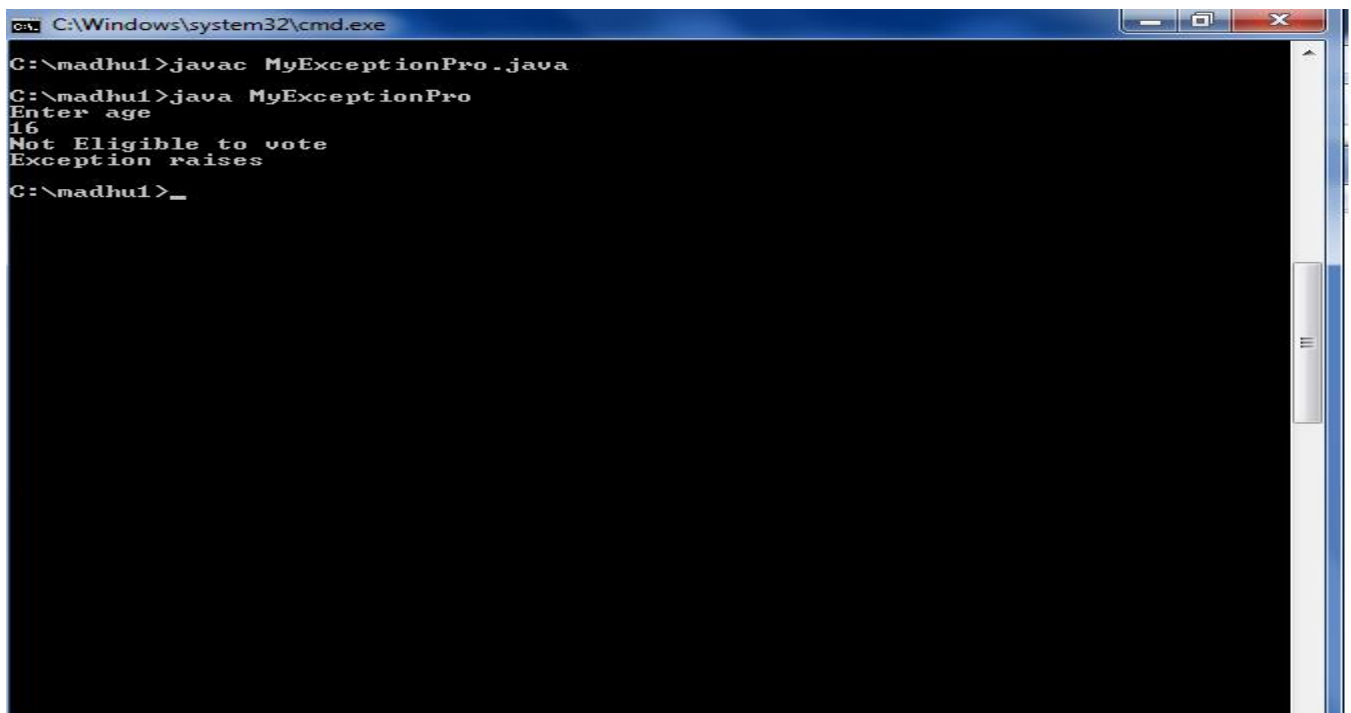
By the help of custom exception, you can have your own exception and message.

Source code:

```
import java.util.*;
import java.lang.*;
```

```
class ExceptionPro2 extends Exception
{
ExceptionPro2(String msg)
{
System.out.println(msg);
}
}
class MyExceptionPro
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter age");
int age=sc.nextInt();
try
{
if(age>18)
{
System.out.println("Eligible to vote");
}
else
{
throw new ExceptionPro2("Not Eligible to vote");
}
}
catch(ExceptionPro2 e)
{
System.out.println("Exception raises");
}
}
}
```

Output:



```
C:\Windows\system32\cmd.exe
C:\madhu1>javac MyExceptionPro.java
C:\madhu1>java MyExceptionPro
Enter age
16
Not Eligible to vote
Exception raises
C:\madhu1>_
```

EXERCISE-10

(THREADS)

44. Write a java program that creates threads by extending Thread class. First thread display “Good morning” every 1 sec, second thread Display “Hello” every 2 secs and the third thread display “Welcome” every 3 secs.

Description:

How to create thread

There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

Thread class:

Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

Commonly used Constructors of Thread class:

- Thread()
- Thread(String name)
- Thread(Runnable r)
- Thread(Runnable r,String name)

Runnable interface:

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. Runnable interface have only one method named run().

1. public void run(): is used to perform action for a thread.

Starting a thread:

start() method of Thread class is used to start a newly created thread.

It performs following tasks:

A new thread starts(with new callstack).

The thread moves from New state to the Runnable state.

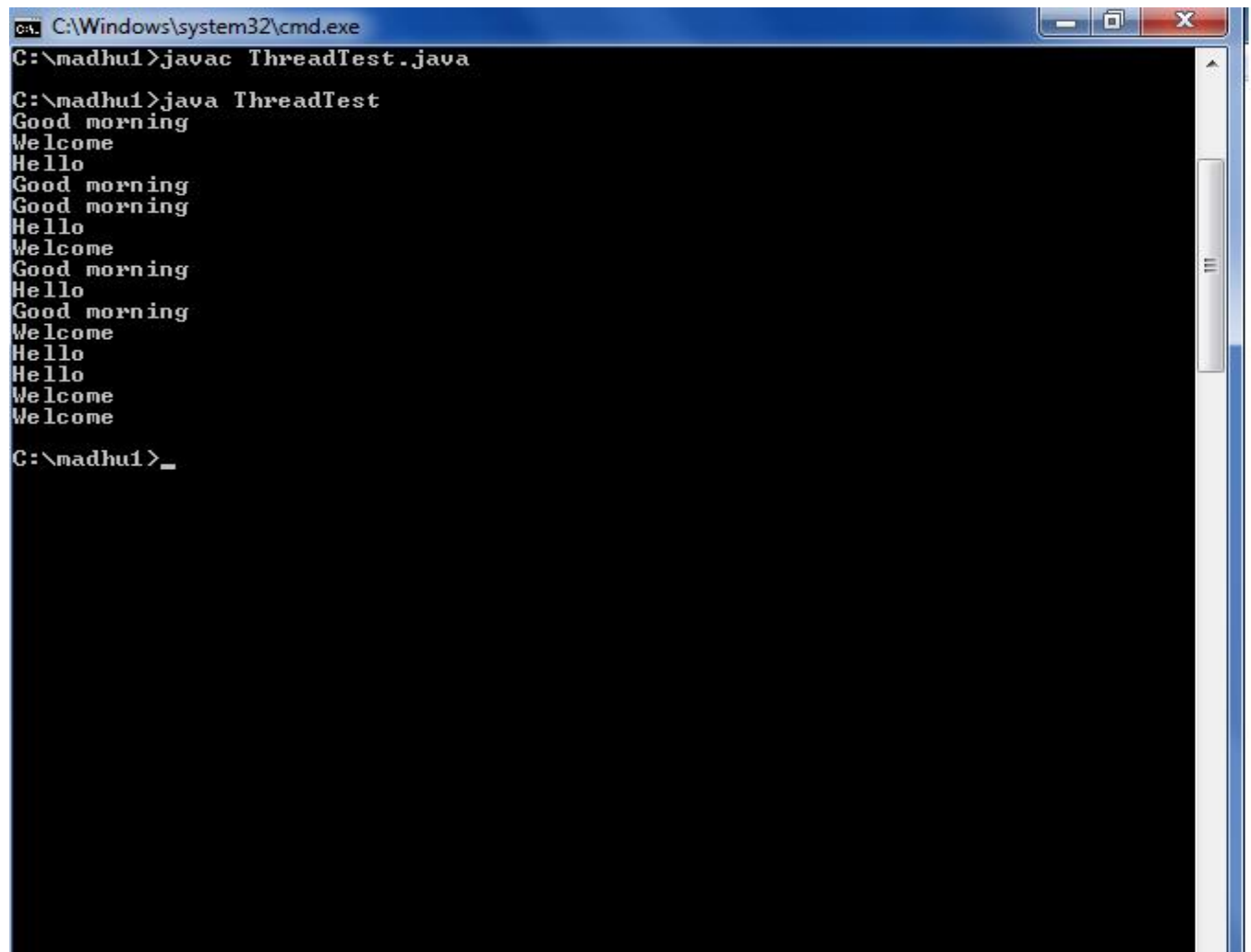
When the thread gets a chance to execute, its target run() method will run.

Source Code:

```
import java.lang.Thread;
class A extends Thread
{
    public void run()
    {
        try
        {
            for(int i=0;i<5;i++)
            {
                System.out.println("Good morning");
                Thread.sleep(1000);
            }
        }
        catch(Exception e)
        {
            System.out.println("Handled");
        }
    }
}
class B extends Thread
{
    public void run()
    {
        try
        {
            for(int i=0;i<5;i++)
            {
                System.out.println("Hello");
                Thread.sleep(2000);
            }
        }
        catch(Exception e)
        {
            System.out.println("Handled");
        }
    }
}
class C extends Thread
{
    public void run()
    {
        try
        {
            for(int i=0;i<5;i++)
            {
                System.out.println("Welcome");
                Thread.sleep(3000);
            }
        }
        catch(Exception e)
```

```
{  
System.out.println("Handled");  
}  
}  
}  
class ThreadTest  
{  
public static void main(String args[])  
{  
A a=new A();  
B b=new B();  
C c=new C();  
Thread t1=new Thread(a);  
Thread t2=new Thread(b);  
Thread t3=new Thread(c);  
t1.start();  
t2.start();  
t3.start();  
}  
}
```

Output:



```
C:\Windows\system32\cmd.exe  
C:\madhu1>javac ThreadTest.java  
C:\madhu1>java ThreadTest  
Good morning  
Welcome  
Hello  
Good morning  
Good morning  
Hello  
Welcome  
Good morning  
Hello  
Good morning  
Welcome  
Hello  
Hello  
Welcome  
Welcome  
C:\madhu1>_
```


45. Write a java program illustrating isAlive() and join().

Description:

isAlive() : It tests if this thread is alive. A thread is alive if it has been started and has not yet died. There is a transitional period from when a thread is running to when a thread is not running. After the run() method returns, there is a short period of time before the thread stops. If we want to know if the start method of the thread has been called or if thread has been terminated, we must use isAlive() method. This method is used to find out if a thread has actually been started and has yet not terminated.

join() : When the join() method is called, the current thread will simply wait until the thread it is joining with is no longer alive.

Or we can say the method that you will more commonly use to wait for a thread to finish is called join(). This method waits until the thread on which it is called terminates. Its name comes from the concept of the calling thread waiting until the specified thread joins it. Additional forms of join() allow you to specify a maximum amount of time that you want to wait for the specified thread to terminate.

Source code:

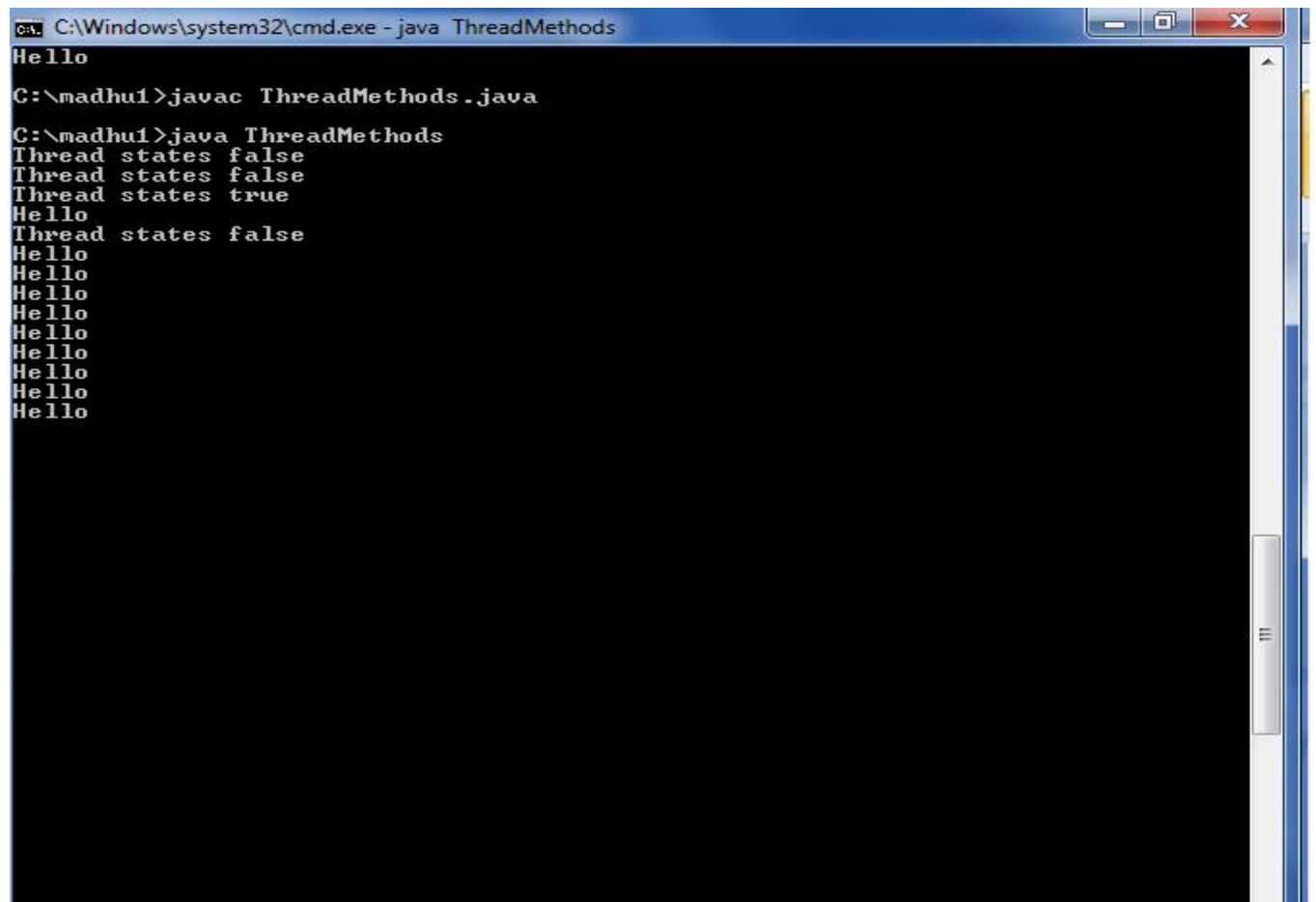
```
import java.lang.Thread;
class A extends Thread
{
    public void run()
    {
        try
        {
            for(int i=0;i<5;i++)
            {
                System.out.println("Hello");
                Thread.sleep(1000);
            }
        }
        catch(Exception e)
        {
            System.out.println("Handled");
        }
    }
}
class ThreadMethods
{
    public static void main(String args[])
    {
        A a1=new A();
        A a2=new A();
        System.out.println("Thread states" +a1.isAlive());
        System.out.println("Thread states" +a2.isAlive());
        a1.start();
        System.out.println("Thread states" +a1.isAlive());
        try
```

```

{
a2.join();
}
catch(Exception e)
{
}
System.out.println("Thread states" +a2.isAlive());
try
{
a2.start();
}
catch(Exception e)
{
}
}
}

```

Output:



```

C:\Windows\system32\cmd.exe - java ThreadMethods
Hello
C:\madhu1>javac ThreadMethods.java
C:\madhu1>java ThreadMethods
Thread states false
Thread states false
Thread states true
Hello
Thread states false
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello

```

46. Write a java program illustrating Daemon Threads.

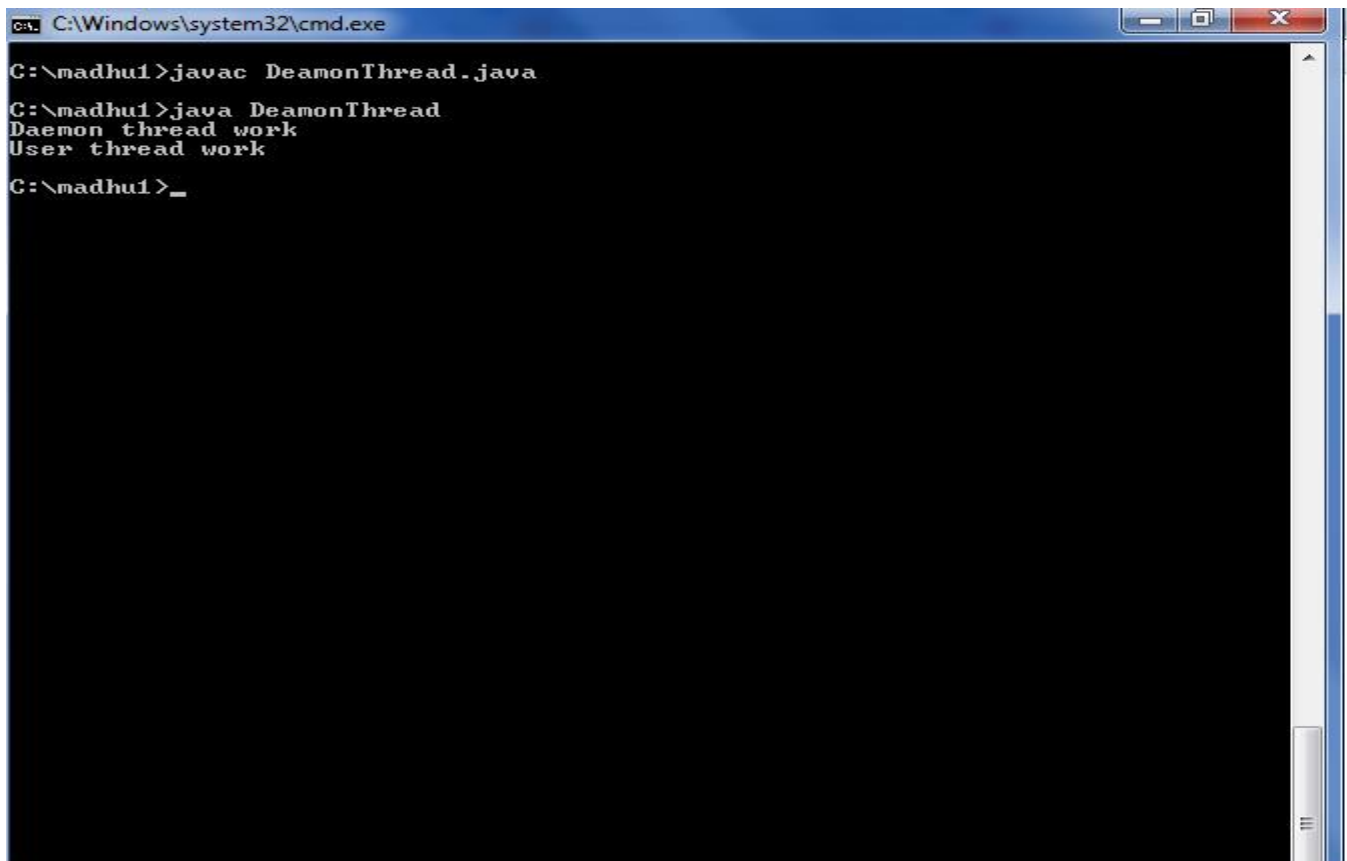
Description:

Daemon thread in java is a service provider thread that provides services to the user thread. Its life depend on the mercy of user threads i.e. when all the user threads dies, JVM terminates this thread automatically. There are many java daemon threads running automatically e.g. gc, finalizer etc.

Source code:

```
class DeamonThread extends Thread
{
public void run()
{
if(Thread.currentThread().isDaemon())
{
System.out.println("Daemon thread work");
}
else
{
System.out.println("User thread work");
}
}
public static void main(String args[])
{
DeamonThread t1=new DeamonThread();
DeamonThread t2=new DeamonThread();
t1.setDaemon(true);
t1.start();
t2.start();
}
}
```

Output:



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The user is in the directory "C:\madhu1". The commands and output are as follows:

```
C:\madhu1>javac DeamonThread.java
C:\madhu1>java DeamonThread
Daemon thread work
User thread work
C:\madhu1>_
```

EXERCISE-10

(THREADS CONTINUITY)

47. Write a java program on producer and consumer problem.

Description:

In computing, the producer-consumer problem (also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue.

- The producer's job is to generate data, put it into the buffer, and start again.
- At the same time, the consumer is consuming the data (i.e. removing it from the buffer), one piece at a time.

Source code:

```
import java.util.*;
class Producer extends Thread
{
    StringBuffer sb=new StringBuffer();
    public void run()
    {
        synchronized(sb)
        {
            for(int i=1;i<=10;i++)
            {
                sb.append(i+":");
                System.out.println("Appending");
                try
                {
                    Thread.sleep(1000);
                }
                catch(InterruptedException e)
                {
                }
            }
            sb.notify();
        }
    }
}
class Consumer extends Thread
{
    Producer prod;
    Consumer(Producer prod)
    {
        this.prod=prod;
    }
    public void run()
    {

```

```

synchronized(prod.sb)
{
try
{
prod.sb.wait();
}
catch(Exception e)
{
}
System.out.println("Data is"+prod.sb);
}
}
}

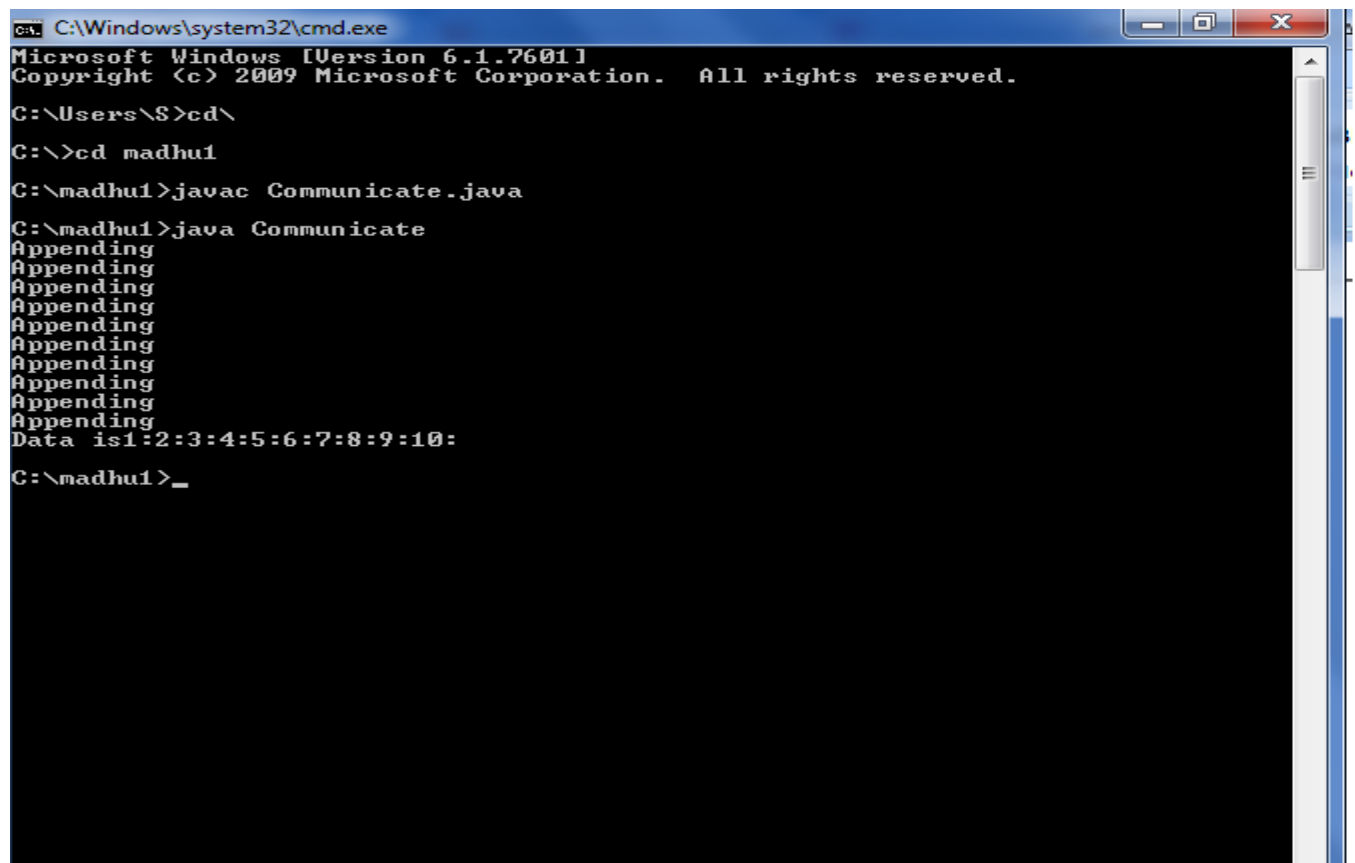
```

```

class Communicate
{
public static void main(String args[])
{
Producer p=new Producer();
Consumer c=new Consumer(p);
Thread t1=new Thread(p);
Thread t2=new Thread(c);
t2.start();
t1.start();
}
}

```

Output:



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\S>cd\
C:\>cd madhu1
C:\madhu1>javac Communicate.java
C:\madhu1>java Communicate
Appending
Appending
Appending
Appending
Appending
Appending
Appending
Appending
Appending
Appending
Data is1:2:3:4:5:6:7:8:9:10:
C:\madhu1>_

```

48. Write a case study on thread synchronization after solving the above producer consumer problem.

Description:

Synchronization:

When thread is already acting on a object, preventing any other thread from acting on the same object is called thread synchronization or thread Safe. Thread synchronization is recommended when multiple threads are used on the same object.

In the above producer consumer program we are synchronization on string Buffer() object. In the producer thread sb.notify() method. This method is notification to the consumer thread that the string buffer object sb is available, and it can be used now. Mean while, what the thread is doing? It is waiting for the notification that the string buffer object sb is available. Here, there is no need of using sleep() method to go into sleep for some time wait() method stops waiting as soon as it receives the notification. So there is no time delay to receive the data from the producer.

**EXERCISE-12
(PACKAGES)**

49. write a java program illustrate class path.

Description:

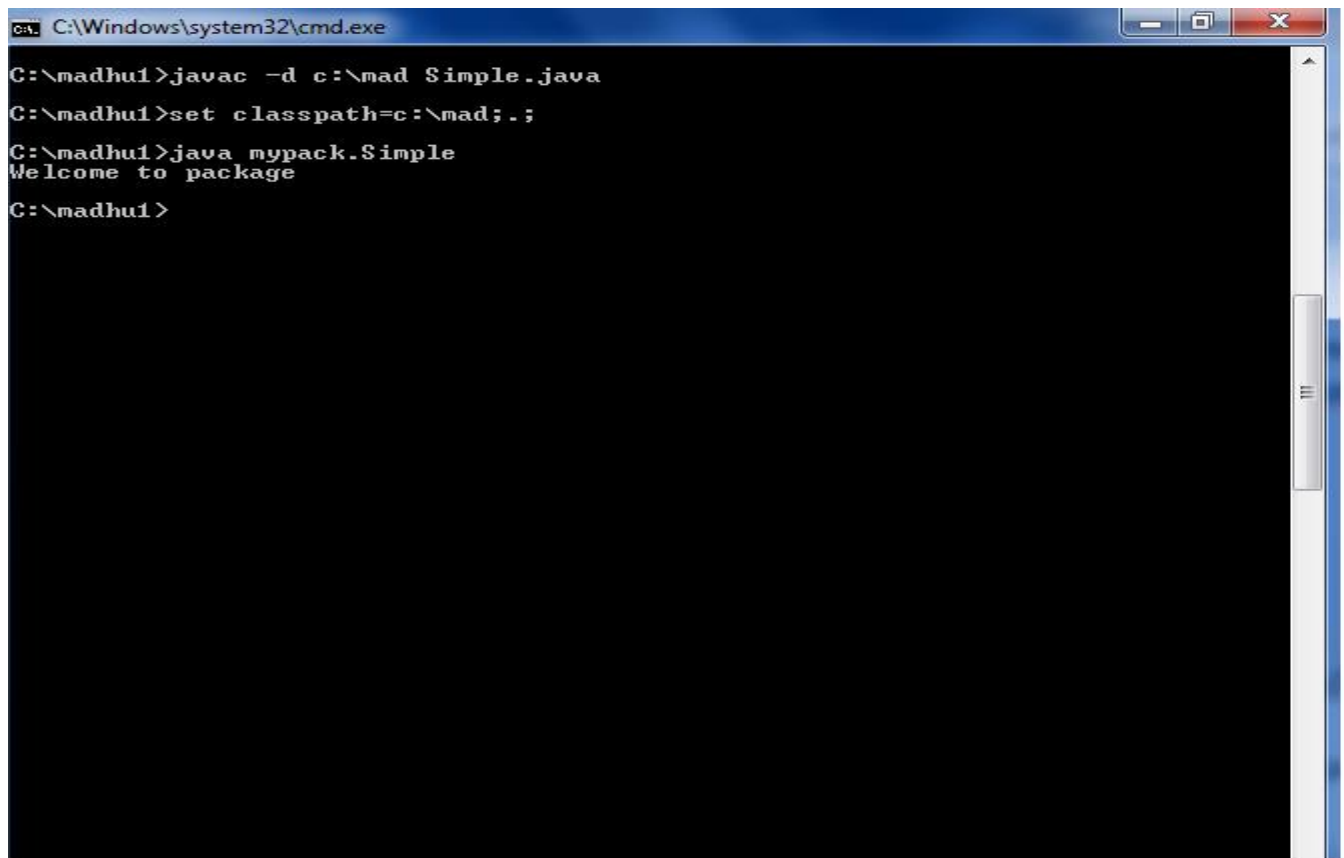
Create a package with .class file. Then run the following commands in command prompt:

```
->javac -d c:\foldername Filename.java  
->set classpath=c:\foldername;.;  
->java packagename.filename
```

Source code:

```
package mypack;  
public class Simple  
{  
public static void main(String args[])  
{  
System.out.println("Welcome to package");  
}  
}
```

Output:

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The command prompt shows the following sequence of commands and output:

```
C:\madhu1>javac -d c:\mad Simple.java
C:\madhu1>set classpath=c:\mad;.
C:\madhu1>java mypack.Simple
Welcome to package
C:\madhu1>
```

50. write a case study on including in class path in your os environment of your package.

Description:

The CLASSPATH is an environment variable that tells the java compiler where to look for class files to import. CLASSPATH is generally set to a directory or a JAR file.

To see what is there in currently in CLASSPATH variable in your system. You can type the command in windows.

echo% CLASSPATH%

Suppose, preceeding command has displayed class path as:

C:\rnr;

This means the current class path is set to rnr directory in c:\ and also the current directory represented by dot(.) our package pack does not exist in either rnr or current directory. Our package exist in D:\sub,as:

Set CLASSPATH=D:\sub;.;%CLASSPATH%.

51 .write a java program that import and use the defined package.

Description:

STEPS OF CREATING USER DEFINED PACKAGES JAVA AND USING THEM.

1. Create a package with a .class file
2. set the classpath from the directory from which you would like to access. It may be in a different drive and directory. Let us call it as a target directory.
3. Write a program and use the file from the package.

To create a package

->javac -d . classname.java

To compile a file

->javac filename.java

To run a file

->java filename

Source Code:

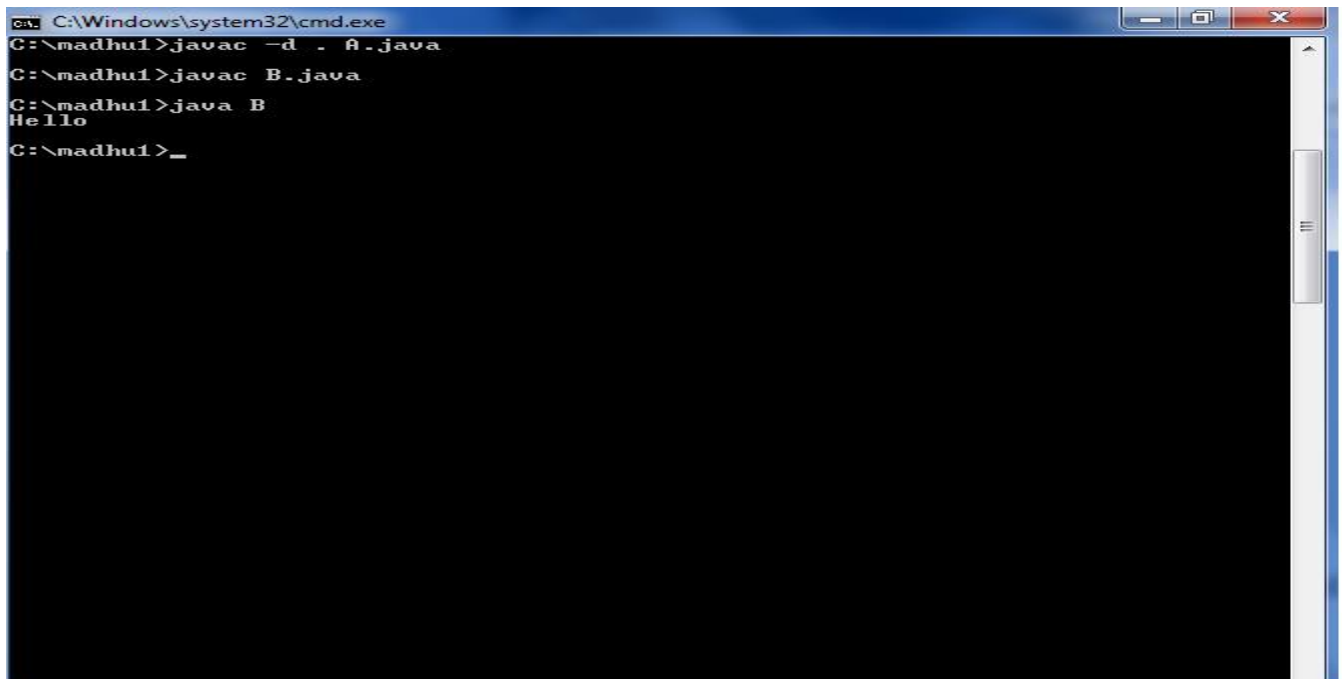
A.java

```
package pack;  
public class A  
{  
    public void msg()  
    {  
        System.out.println("Hello");  
    }  
}
```

B.java

```
import pack.A;  
class B  
{  
    public static void main(String args[])  
    {  
        A obj=new A();  
        obj.msg();  
    }  
}
```

Output:



```
C:\Windows\system32\cmd.exe  
C:\madhu1>javac -d . A.java  
C:\madhu1>javac B.java  
C:\madhu1>java B  
Hello  
C:\madhu1>_
```


EXERCISE-13

(APPLET)

52. write a java program to paint like paint brush in applet.

Description:

The method paint() is automatically called whenever there is a need to display an applet window, this need could arise in certain situations -

- When an applet window is brought up on the screen for the first time.
- When an applet window is brought up on the screen from a minimized state, this leads the redrawing of the applet window and hence the paint() method is automatically called.
- When an applet window is stretched to a new size, this leads to redrawing of the applet window to a new size and hence the paint() method is automatically called.

Source code:

Paint.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class Paint extends Applet
implements MouseMotionListener {

    int width, height;
    Image backbuffer;
    Graphics backg;

    public void init() {
        width = getSize().width;
        height = getSize().height;

        backbuffer = createImage( width, height );
        backg = backbuffer.getGraphics();
        backg.setColor( Color.black );
        backg.fillRect( 0, 0, width, height );
        backg.setColor( Color.blue );

        addMouseMotionListener( this );
    }

    public void mouseMoved( MouseEvent e ) { }
    public void mouseDragged( MouseEvent e ) {
        int x = e.getX();
        int y = e.getY();
        backg.fillOval(x-10,y-10,20,20);
        repaint();
        e.consume();
    }

    public void update( Graphics g ) {
```

```
g.drawImage( backbuffer, 0, 0, this );
}
public void paint( Graphics g ) {
update( g );
}
}
```

mypaint.html:

```
<html>
<body>
<applet code="Paint.class" width="300" height="300">
</applet>
</body>
</html>
```

->**javac filename.java**
->**appletviewer htmlfilename.html**

Output:



53. write a java program to display analog clock using applet.

Description:

Approach:

Analog clock can be created by using the Math class. Each hand of the clock will be animating with 1-second delay keeping one end at the centre. The position of the other end can be derived by the system time. The angle formed by a hand of the clock in every second will be different throughout its journey. This is why various instances make a different angle to the horizontal line.

Source code:

MyClock.java:

```
import java.applet.*;
import java.awt.*;
import java.util.*;
import java.text.*;

public class MyClock extends Applet implements Runnable {

    int width, height;
    Thread t = null;
    boolean threadSuspended;
    int hours=0, minutes=0, seconds=0;
    String timeString = "";

    public void init() {
        width = getSize().width;
        height = getSize().height;
        setBackground( Color.black );
    }

    public void start() {
        if ( t == null ) {
            t = new Thread( this );
            t.setPriority( Thread.MIN_PRIORITY );
            threadSuspended = false;
            t.start();
        }
        else {
            if ( threadSuspended ) {
                threadSuspended = false;
                synchronized( this ) {
                    notify();
                }
            }
        }
    }

    public void stop() {
        threadSuspended = true;
    }

    public void run() {
        try {
            while (true) {

                Calendar cal = Calendar.getInstance();
                hours = cal.get( Calendar.HOUR_OF_DAY );
                if ( hours > 12 ) hours -= 12;
                minutes = cal.get( Calendar.MINUTE );
                seconds = cal.get( Calendar.SECOND );
```

```

SimpleDateFormat formatter
    = new SimpleDateFormat( "hh:mm:ss", Locale.getDefault() );
Date date = cal.getTime();
timeString = formatter.format( date );

// Now the thread checks to see if it should suspend itself
if ( threadSuspended ) {
    synchronized( this ) {
        while ( threadSuspended ) {
            wait();
        }
    }
    repaint();
    t.sleep( 1000 ); // interval specified in milliseconds
}
}
catch (Exception e) { }
}

void drawHand( double angle, int radius, Graphics g ) {
    angle -= 0.5 * Math.PI;
    int x = (int)( radius*Math.cos(angle) );
    int y = (int)( radius*Math.sin(angle) );
    g.drawLine( width/2, height/2, width/2 + x, height/2 + y );
}

void drawWedge( double angle, int radius, Graphics g ) {
    angle -= 0.5 * Math.PI;
    int x = (int)( radius*Math.cos(angle) );
    int y = (int)( radius*Math.sin(angle) );
    angle += 2*Math.PI/3;
    int x2 = (int)( 5*Math.cos(angle) );
    int y2 = (int)( 5*Math.sin(angle) );
    angle += 2*Math.PI/3;
    int x3 = (int)( 5*Math.cos(angle) );
    int y3 = (int)( 5*Math.sin(angle) );
    g.drawLine( width/2+x2, height/2+y2, width/2 + x, height/2 + y );
    g.drawLine( width/2+x3, height/2+y3, width/2 + x, height/2 + y );
    g.drawLine( width/2+x2, height/2+y2, width/2 + x3, height/2 + y3 );
}

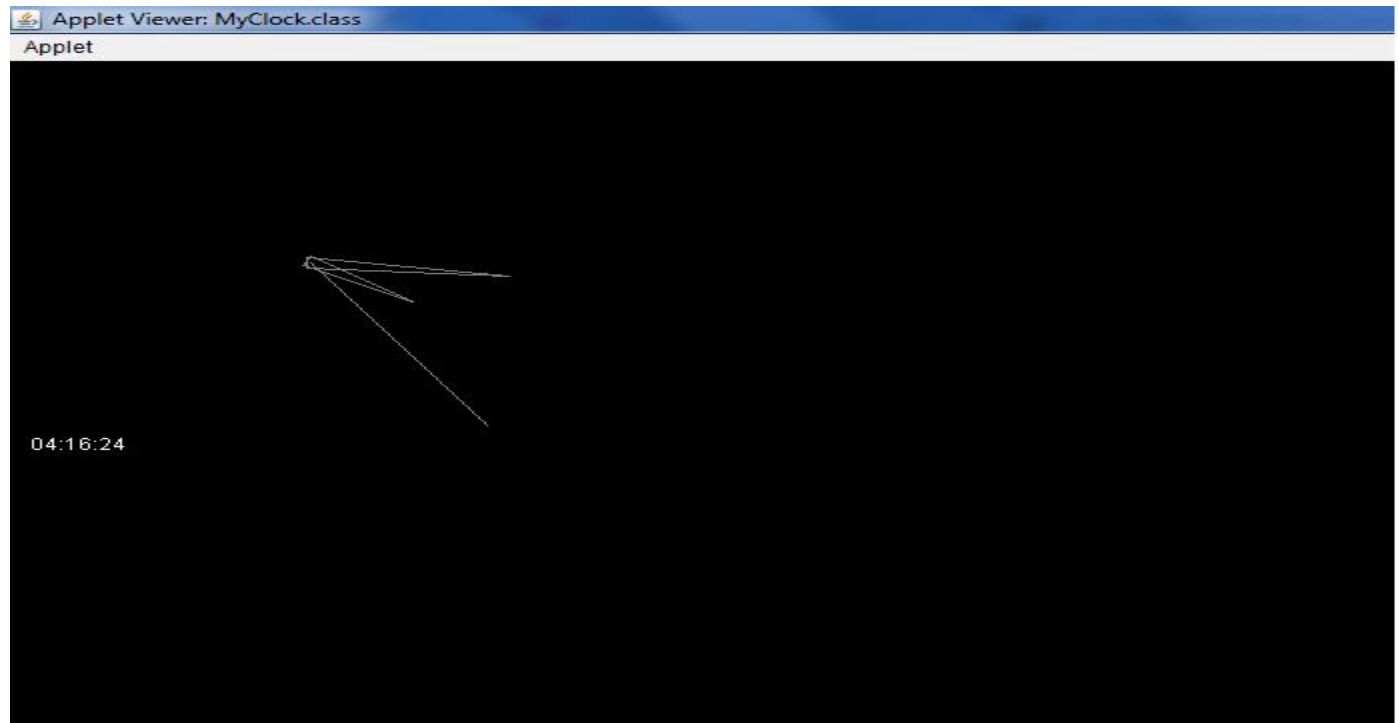
public void paint( Graphics g ) {
    g.setColor( Color.gray );
    drawWedge( 2*Math.PI * hours / 12, width/5, g );
    drawWedge( 2*Math.PI * minutes / 60, width/3, g );
    drawHand( 2*Math.PI * seconds / 60, width/2, g );
    g.setColor( Color.white );
    g.drawString( timeString, 10, height-10 );
}
}

```

myclock.html:

```
<html>
<body>
<applet code="MyClock.class" width="300" height="300">
</applet>
</body>
</html>
```

Output:



54.write a java program to create different shapes and fill colors using applet.

Description:

The method `drawOval(int x,int y,int width,int height)` is used to draw an oval starting from the co-ordinate (x,y) of given width and height.

The method `fillOval(int x,int y,int width,int height)` is used to fill an oval starting from the co-ordinate (x,y) of given width and height.

Source code:

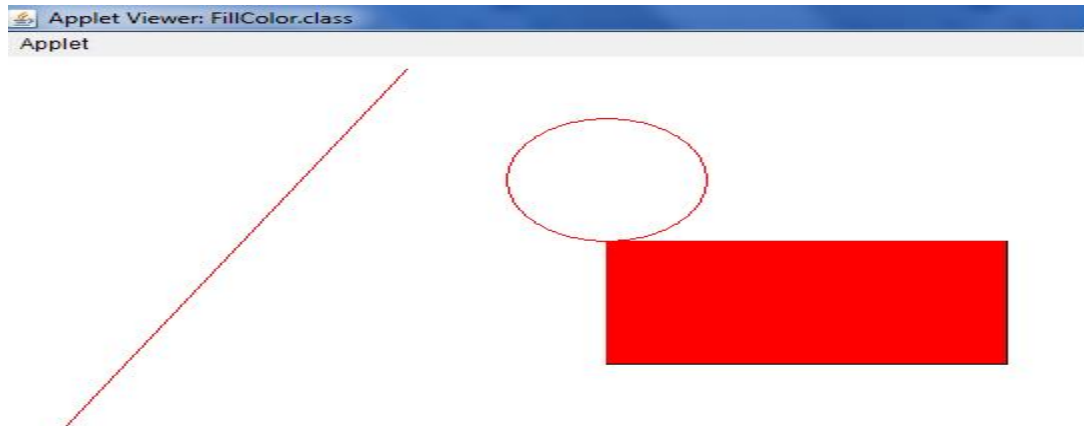
```
import java.applet.*;
import java.awt.*;
public class FillColor extends Applet {
    int x = 300, y = 100, r = 50;
    public void paint(Graphics g) {
        g.drawRect(300,150,200,100);
        g.setColor(Color.red);
        g.fillRect( 300,150, 200, 100 );
        g.drawLine(30,300,200,10);
    }
}
```

```

g.setColor(Color.red);
g.fillRect( 300,150, 200, 100 );
g.drawOval(x-r,y-r,100,100);
g.setColor(Color.red);
g.fillRect( 300,150, 200, 100 );
}
}

```

Output:



EXERCISE-14 (EVENT HANDLING)

55. Write a java program to display the x and y position of the current movement using mouse.

Description:

To find, the MouseEvent class defines two methods getX() and getY() that return x and y coordinates position of mouse action.

- int getX(): Returns the horizontal x position of the event relative to the source component.
- int getY(): Returns the vertical y position of the event relative to the source component.

Source code:

```

import java.awt.*;
import java.awt.event.*;
public class MouseXY extends Frame implements MouseListener, MouseMotionListener
{
    int x, y;
    String str="";
    public MouseXY()
    {
        addMouseListener(this);
        addMouseMotionListener(this);
    }
}

```

```
setSize(300, 300);
setVisible(true);
}

public void mousePressed(MouseEvent e)
{
    x = e.getX();
    y = e.getY();
    str = "Mouse Pressed";
    repaint();
}
public void mouseReleased(MouseEvent e)
{
    x = e.getX();
    y = e.getY();
    str = "Mouse Released";
    repaint();
}
public void mouseClicked(MouseEvent e)
{
    x = e.getX();
    y = e.getY();
    str = "Mouse Clicked";
    repaint();
}
public void mouseEntered(MouseEvent e)
{
    x = e.getX();
    y = e.getY();
    str = "Mouse Entered";
    repaint();
}
public void mouseExited(MouseEvent e)
{
    x = e.getX();
    y = e.getY();
    str = "Mouse Exited";
    repaint();
}
public void mouseMoved(MouseEvent e)
{
    x = e.getX();
    y = e.getY();
    str = "Mouse Moved";
    repaint();
}
public void mouseDragged(MouseEvent e)
{
    x = e.getX();
    y = e.getY();
    str = "Mouse dragged";
    repaint();
}
public void paint(Graphics g)
{

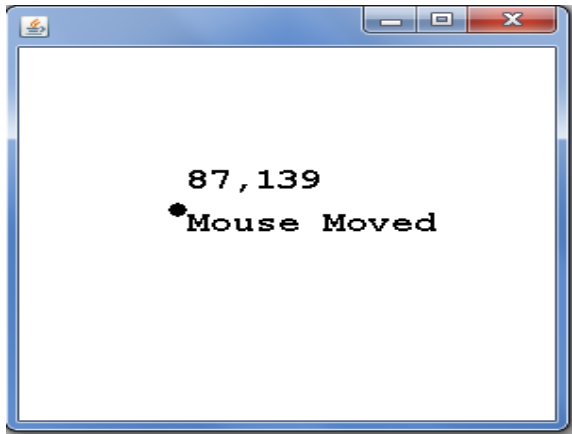
```

```

g.setFont(new Font("Monospaced", Font.BOLD, 20));
g.fillOval(x, y, 10, 10);
g.drawString(x + "," + y, x+10, y -10);
g.drawString(str, x+10, y+20);
}
public static void main(String args[])
{
    new MouseXY();
}
}

```

Output:



56. write a java program that identifies key-up, key-down event user entering text in a applet.

Description:

Java KeyListener Interface

The Java KeyListener is notified whenever you change the state of key. It is notified against KeyEvent. The KeyListener interface is found in java.awt.event package. It has three methods.

Methods of KeyListener interface

The signature of 3 methods found in KeyListener interface are given below:

1. public abstract void keyPressed(KeyEvent e);
2. public abstract void keyReleased(KeyEvent e);
3. public abstract void keyTyped(KeyEvent e);

Source code:

```
import java.awt.*;
import java.awt.event.*;
public class KeyListenerExample extends Frame implements KeyListener{
    Label l;
    TextArea area;
    KeyListenerExample(){

        l=new Label();
        l.setBounds(20,50,100,20);
        area=new TextArea();
        area.setBounds(20,80,300, 300);
        area.addKeyListener(this);

        add(l);add(area);
        setSize(400,400);
        setLayout(null);
        setVisible(true);
    }
    public void keyPressed(KeyEvent e) {
        l.setText("Key Pressed");
    }
    public void keyReleased(KeyEvent e) {
        l.setText("Key Released");
    }
    public void keyTyped(KeyEvent e) {
        l.setText("Key Typed");
    }
    }

    public static void main(String[] args) {
        new KeyListenerExample();
    }
}
```

Output:

