

## Experiment 01

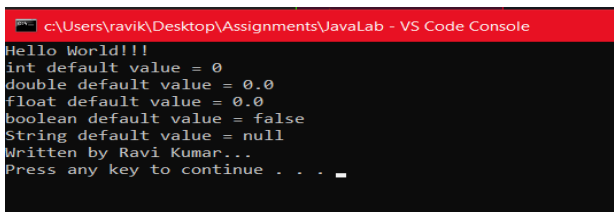
**Aim:-** WAP a java program to print "Hello World." WAP in java to print the values of various primitive data types.

### Procedure/Algorithm:-

- 1.Start Program
- 2.Define Variables for integer, double, float, Boolean, string etc.
- 3.Define function for inserting and defining the values.
- 4.Create class main
- 5.Using System.out.println print Hello World.
6. Create class instance for Test class and print values in main class.
7. End Program.

```
Program:- class Test {  
    int k;  
    double d;  
    float f;  
    boolean istrue;  
    String p;  
    public void printValue() {  
        System.out.println("int default value = "+ k);  
        System.out.println("double default value = "+ d);  
        System.out.println("float default value = "+ f);  
        System.out.println("boolean default value = "+ istrue);  
        System.out.println("String default value = "+ p);  
    }  
}  
  
public class Main {  
    public static void main(String argv[]) {  
        System.out.println("Hello World!!!");  
        Test test = new Test();  
        test.printValue();  
        System.out.println("Written by Ravi Kumar...");  
    }  
}
```

### Output:-



```
c:\Users\ravik\Desktop\Assignments\JavaLab - VS Code Console  
Hello World!!!  
int default value = 0  
double default value = 0.0  
float default value = 0.0  
boolean default value = false  
String default value = null  
Written by Ravi Kumar...  
Press any key to continue . . .
```

**Result:-** Program to write to print "Hello World." and to print the values of various primitive data types is successfully compiled and executed.

## Experiment 02

**Aim:-** WAP in java to demonstrate operator precedence.

### Procudre/Algorithm:-

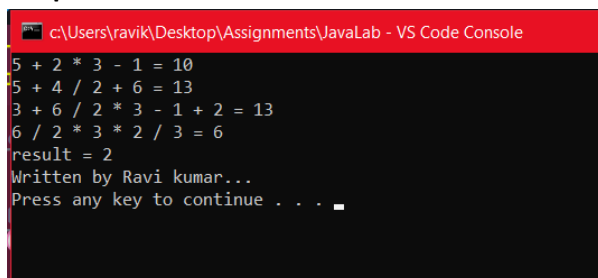
Let the input string be  $a_1a_2 \dots a_n$ . Initially, the stack contains \$.

- Repeat forever
- If only \$ is on the stack and only \$ is on the input then accept and break else
- begin
- let a be the topmost terminal symbol on the stack and let b be the current input symbols.
- If  $a < b$  or  $a = b$  then shift b onto the stack /\*Shift\*/
- else if  $a > b$  then /\*reduce\*/
- repeat pop the stack
- until the top stack terminal is related by  $<$  to the terminal most recently popped.
- else call the error-correcting routine end.

### Program:-

```
public class Main {  
    public static void main (String[] args) {  
        int result = 0;  
        result = 5 + 2 * 3 - 1;  
        System.out.println("5 + 2 * 3 - 1 = " + result);  
        result = 5 + 4 / 2 + 6;  
        System.out.println("5 + 4 / 2 + 6 = " + result);  
        result = 3 + 6 / 2 * 3 - 1 + 2;  
        System.out.println("3 + 6 / 2 * 3 - 1 + 2 = " + result);  
        result = 6 / 2 * 3 * 2 / 3;  
        System.out.println("6 / 2 * 3 * 2 / 3 = " + result);  
        int x = 2;  
        result = x++ + x++ * --x / x++ - --x + 3 >> 1 | 2;  
        System.out.println("result = " + result);  
        System.out.println("Written by Ravi kumar...");  
    }  
}
```

### Output:-



```
c:\Users\ravik\Desktop\Assignments\JavaLab - VS Code Console  
5 + 2 * 3 - 1 = 10  
5 + 4 / 2 + 6 = 13  
3 + 6 / 2 * 3 - 1 + 2 = 13  
6 / 2 * 3 * 2 / 3 = 6  
result = 2  
Written by Ravi kumar...  
Press any key to continue . . .
```

**Result:-** Program in java to demonstrate operator precedence is successfully compiled and executed.

### Experiment 03

**Aim:-** Write a program to read an integer value through Scanner class and find addition of two numbers using function.

**Algorithm:-**

Step 1: Start

Step 2: Declare variables num1, num2 and sum.

Step 3: Read values for num1 and num2.

Step 4: Add num1 and num2 and assign the result to sum.

sum=num1+num2

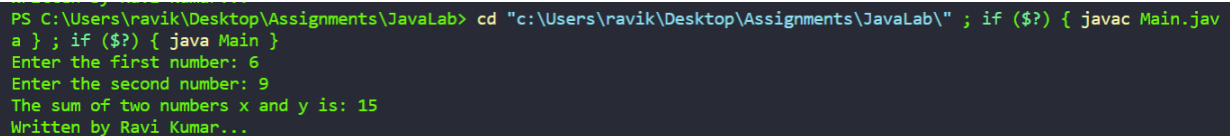
Step 5: Display sum

Step 6: Stop

**Program:-**

```
import java.util.Scanner;
public class Main {
    public static void main(String args[])
    {
        int x, y, sum;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the first number: ");
        x = sc.nextInt();
        System.out.print("Enter the second number: ");
        y = sc.nextInt();
        sum = sum(x, y);
        System.out.println("The sum of two numbers x and y is: " + sum);
        System.out.println("Written by Ravi Kumar...");
    }
    //method that calculates the sum
    public static int sum(int a, int b)
    {
        int sum = a + b;
        return sum;
    }
}
```

**Output:-**



```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java
a } ; if ($?) { java Main }
Enter the first number: 6
Enter the second number: 9
The sum of two numbers x and y is: 15
Written by Ravi Kumar...
```

**Result:-** Program to read an integer value through Scanner class and find addition of two numbers using function is successfully compiled and executed.

## Experiment 04

**Aim:-** Write a program that uses length property for displaying any number of command line arguments

**Algorithm:-** Write your java program and save it with .java extension in your local system

1. Open your cmd(command prompt) and navigate to the directory where you saved your program
2. Compile your program using command 'javac filename.java'. This will convert the .java file to .class file
3. Write the run command 'java filename' and continue the command with all the inputs you want to give to your code
4. After writing all your inputs, run your command

**Program:-**

```
public class Main {  
    public static void main(String args[]) {  
        for(int i = 0; i<args.length; i++) {  
            System.out.println("args[" + i + "]: " + args[i]);  
            System.out.println("Written by Ravi Kumar...");  
        }  
    }  
}
```

**Output:-**

```
C:\Users\ravik\Desktop\Assignments\JavaLab>java Main ravi  
args[0]: ravi  
Written by Ravi Kumar...
```

**Result:-** Program that uses length property for displaying any number of command line arguments is compiled and executed successfully.

## Experiment 05

**Aim:-** Write a program to create constructor of a class and initialize values in it and later print them.

**Algorithm:-** 1: Start the program.

2: Declare the class as Example with a and b variables.

3: Declare the 'Constructor declaration' in class

4: Define 'Constructor definition' outside Class with a and b initialization

5: Write function for display values a and b

6: Main function declaration and definition

7: Create object for Example class in main Function

8: Call display function using Example class object.

9: Check whether the k value is 1 or 0.

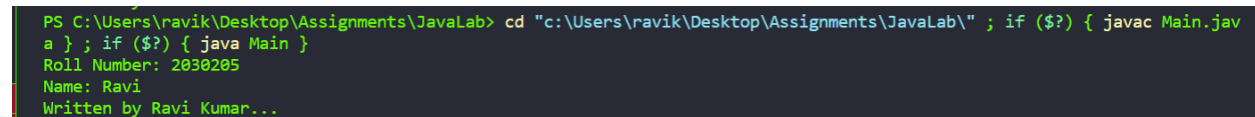
10: Stop the program.

**Program:-**

```
class Student {
    private int rno;
    private String name;
    public Student(int r, String n) {
        rno = r;
        name = n;
    }
    public void display() {
        System.out.println("Roll Number: " + rno);
        System.out.println("Name: " + name);
    }
}

public class Main {
    public static void main(String[] args) {
        Student s = new Student(2030206, "Ravi");
        s.display();
        System.out.println("Written by Ravi Kumar...");
    }
}
```

**Output:-**



```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Roll Number: 2030206
Name: Ravi
Written by Ravi Kumar...
```

**Result:-** Program to create constructor of a class and initialize values in it and later print them is compiled and executed successfully.

## Experiment 06

**Aim:-** WAP in java to sort n numbers using bubble sort.

**Algorithm:-**

```
begin BubbleSort(list)
  for all elements of list
    if list[i] > list[i+1]
      swap(list[i], list[i+1])
    end if
  end for
  return list
end BubbleSort
```

```
Program:- public class Main {
  static void bubbleSort(int[] arr) {
    int n = arr.length;
    int temp = 0;
    for(int i=0; i < n; i++){
      for(int j=1; j < (n-i); j++){
        if(arr[j-1] > arr[j]){
          //swap elements
          temp = arr[j-1];
          arr[j-1] = arr[j];
          arr[j] = temp;
        }
      }
    }
  }

  public static void main(String[] args) {
    int arr[] ={3,60,35,2,45,320,5};
    System.out.println("Array Before Bubble Sort");
    for(int i=0; i < arr.length; i++){
      System.out.print(arr[i] + " ");
    }
    System.out.println();

    bubbleSort(arr);//sorting array elements using bubble sort
    System.out.println("Array After Bubble Sort");
    for(int i=0; i < arr.length; i++){
      System.out.println(arr[i] + " ");
    }
    System.out.println("Written by Ravi Kumar...");
  }
}
```

**Output:-**

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Array Before Bubble Sort
3 60 35 2 45 320 5
Array After Bubble Sort
2
3
5
35
45
60
320
Written by Ravi Kumar...
PS C:\Users\ravik\Desktop\Assignments\JavaLab>
```

Result:- Program in java to sort n numbers using bubble sort is compiled and executed successfully.

## Experiment 07

**Aim:-** WAP in java to find addition of two Matrices.

**Algorithm:-** Input matrix 1 and matrix 2.

If the number of rows and number of columns of matrix 1 and matrix 2 is equal,

for i=1 to rows[matrix 1]

for j=1 to columns [matrix 1]

Input matrix 1 [i,j]

Input matrix 2 [i,j]

matrix 3 [i,j]= matrix 1 [i,j]+ matrix 2 [i,j];

Display matrix 3 [i,j];

Program:- public class Main{

public static void main(String args[]){

//creating two matrices

int a[][]={{1,3,4},{2,4,3},{3,4,5}};

int b[][]={{1,3,4},{2,4,3},{1,2,4}};

//creating another matrix to store the sum of two matrices

int c[][]=new int[3][3]; //3 rows and 3 columns

//adding and printing addition of 2 matrices

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

c[i][j]=a[i][j]+b[i][j]; //use - for subtraction

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

}

System.out.println();

}

System.out.println("Written by Ravi Kumar...");}}

**Output:-**

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
2 6 8
4 8 6
4 6 9
Written by Ravi Kumar...
```

**Result:-** Program in java to find addition of two Matrices is compiled and executed successfully.



## Experiment 08

**Aim:-** Write a java code to implement the concept of method overloading.

**Algorithm:-** Start the program.

Create the class display with variables and functions.

Pass arguments in the first class display with variables

Pass arguments in the second class display with another variables

In the main(), using temporary variable assign it with the constructor class

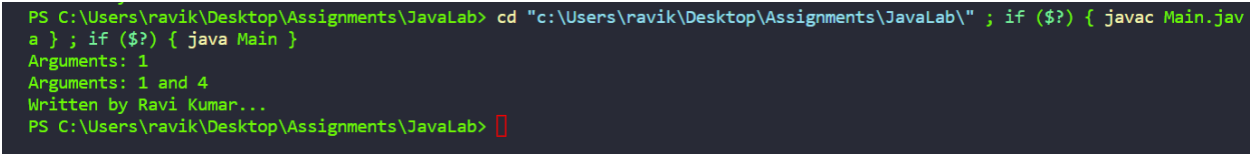
Display the options in the main().

Stop the program.

**Program:-**

```
class Main {  
    private static void display(int a){  
        System.out.println("Arguments: " + a);  
    }  
    private static void display(int a, int b){  
        System.out.println("Arguments: " + a + " and " + b);  
    }  
    public static void main(String[] args) {  
        display(1);  
        display(1, 4);  
        System.out.println("Written by Ravi Kumar...");  
    }  
}
```

**Output:-**



```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
Arguments: 1  
Arguments: 1 and 4  
Written by Ravi Kumar...  
PS C:\Users\ravik\Desktop\Assignments\JavaLab>
```

**Result:-** Program to implement the concept of method overloading is compiled and executed successfully.

## Experiment 09

**Aim:-** Create a class Shape and override area () method to calculate area of rectangle, square and circle.

**Algorithm:-** Start

Input length and breadth

area = length \* breadth

print area

Take radius as input from the user using std input.

Calculate the area of circle using,

area = (3.14)\*r\*r

Print the area to the screen using the std output.

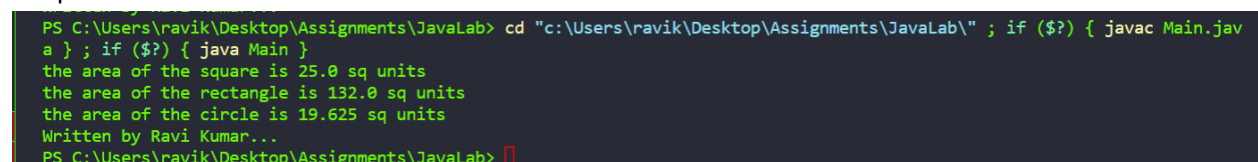
Stop

**Program:-** class OverloadDemo

```
{
    void area(float x)
    {
        System.out.println("the area of the square is "+Math.pow(x, 2)+" sq units");
    }
    void area(float x, float y)
    {
        System.out.println("the area of the rectangle is "+x*y+" sq units");
    }
    void area(double x)
    {
        double z = 3.14 * x * x;
        System.out.println("the area of the circle is "+z+" sq units");
    }
}

class Main
{
    public static void main(String args[])
    {
        OverloadDemo ob = new OverloadDemo();
        ob.area(5);
        ob.area(11,12);
        ob.area(2.5);
        System.out.println("Written by Ravi Kumar...");
    }
}
```

**Output:-**



```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
the area of the square is 25.0 sq units
the area of the rectangle is 132.0 sq units
the area of the circle is 19.625 sq units
Written by Ravi Kumar...
PS C:\Users\ravik\Desktop\Assignments\JavaLab> █
```

**Result:-** Program Create a class Shape and override area () method to calculate area of rectangle, square and circle is compiled and executed successfully.

## Experiment 10

**Aim:-** Write a program to show how method overriding is implemented in java. Create a child class of Animal named 'Bird' and override the parent class methods. Add a new method named fly ().

**Algorithm:-**

**Program:-**

```
class Animal {
    public void eat()
    {
        System.out.println("eat method");
    }
    public void sleep()
    {
        System.out.println("sleep method");
    }
}

class Bird extends Animal{
    @Override
    public void eat() {
        super.eat();
        System.out.println("override eat");
    }

    @Override
    public void sleep() {
        super.sleep();
        System.out.println("override sleep");
    }

    public void fly()
    {
        System.out.println("in fly method");
    }
}

class Main{
    public static void main(String[] args) {
        Animal a =new Animal();
        Bird b = new Bird();
        a.eat();
        a.sleep();
        b.eat();
        b.sleep();
        b.fly();
    }
}
```

```
System.out.println("Written by Ravi Kumar...");  
  
    }  
}
```

#### Output:-

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
eat method  
sleep method  
eat method  
override eat  
sleep method  
override sleep  
in fly method  
Written by Ravi Kumar...  
PS C:\Users\ravik\Desktop\Assignments\JavaLab>
```

**Result:-** Program to show how method overriding is implemented in java. Create a child class of Animal named 'Bird' and override the parent class methods. Add a new method named fly is compiled and executed successfully.

## Experiment 11

**Aim:-** Write a program to implement the concept of abstract classes.

**Algorithm:-**

1. Start

2. Create class Animal using keyword abstract

3. Create function as MakeSound() and assign some values.

4. Implement inheritance using extend keyword to inherit Animal class into other class

5. Create another function as makeSound() and assign some values.

6. Create main class

7. Using temporary variable create an object for parent class

8. Display the values.

9. Stop/End Program

**Program:-** abstract class Animal {

```
    abstract void makeSound();
```

```
    public void eat() {
```

```
        System.out.println("I can eat.");
```

```
    }
```

```
}
```

```
class Dog extends Animal {
```

```
    // provide implementation of abstract method
```

```
    public void makeSound() {
```

```
        System.out.println("Bark bark");
```

```
    }
```

```
}
```

```
class Main {
```

```
    public static void main(String[] args) {
```

```
        // create an object of Dog class
```

```
        Dog d1 = new Dog();
```

```
        d1.makeSound();
```

```
        d1.eat();
```

```
        System.out.println("Written by Ravi Kumar...");
```

```
    }
```

```
}
```

**Output:-**

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Bark bark
I can eat.
Written by Ravi Kumar...
```

**Result:-** Program to implement the concept of abstract classes is compiled and executed successfully.

## Experiment 12

**Aim:-** Write programs for Exception handling using try, catch, throw and finally.

**Algorithm:-** try

```
{
    // statement(s) that might cause exception
}
catch
{
    // statement(s) that handle an exception
    // examples, closing a connection, closing
    // file, exiting the process after writing
    // details to a log file.
}
Finally
{
    //Statement(s) that displays when no exception is founded
}
```

**Program:-** class Main {

```
    public static void main (String args[]) {
        //try block
        try
        {
            System.out.println ("::Try Block::");
            int data = 125 / 5;
            System.out.println ("Result:" + data);
        }
        //catch block
        catch (NullPointerException e) {
            System.out.println ("::Catch Block::");
            System.out.println (e);
        }
        //finally block
        finally {
            System.out.println (":: Finally Block::");
            System.out.println ("No Exception::finally block executed");
        }
        System.out.println ("rest of the code...");
        System.out.println("Written by Ravi Kumar...");
    }
}
```

### Output:-

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
::Try Block::  
Result:25  
:: Finally Block::  
No Exception::finally block executed  
rest of the code...  
Written by Ravi Kumar...  
PS C:\Users\ravik\Desktop\Assignments\JavaLab> []
```

Result:- Programs for Exception handling using try, catch, throw and finally is executed successfully.

### Experiment 13

**Aim:-** Write a java program to implement the usage of customized exceptions

**Algorithm:-** try

```
{
    // statement(s) that might cause exception
}
catch
{
    // statement(s) that handle an exception
    // examples, closing a connection, closing
    // file, exiting the process after writing
    // details to a log file.
}
```

**Program:-** class MyException extends Exception {

```
    public MyException(String s)
    {
        // Call constructor of parent Exception
        super(s);
    }
}
```

// A Class that uses above MyException

```
public class Main {
    // Driver Program
    public static void main(String args[])
    {
        try {
            // Throw an object of user defined exception
            throw new MyException("Ravi Kumar");
        }
        catch (MyException ex) {
            System.out.println("Caught");

            // Print the message from MyException object
            System.out.println(ex.getMessage());
        }
    }
}
```

**Output:-**

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Caught
Ravi Kumar
PS C:\Users\ravik\Desktop\Assignments\JavaLab> █
```

**Result:-** Program to implement the usage of customized exceptions compiled and executed successfully



## Experiment 14

**Aim:-** . Implement concept of multithreading in Java by

- a) Extending Thread class
- b) Implementing Runnable interface

**Algorithm:-** FIB.n < 1

if n < 1

return n

else x = FIB.n-1/

y = FIB.n - 2/

return x + y

**Program:-** class ThreadA extends Thread{

public void run( ) {

for(int i = 1; i <= 5; i++) {

System.out.println("From Thread A with i = "+ i);

}

System.out.println("Exiting from Thread A ...");

}

}

class ThreadB extends Thread {

public void run( ) {

for(int j = 1; j <= 5; j++) {

System.out.println("From Thread B with j= "+j);

}

System.out.println("Exiting from Thread B ...");

}

}

class ThreadC extends Thread{

public void run( ) {

for(int k = 1; k <= 5; k++) {

System.out.println("From Thread C with k = "+ (2\*k-1));

}

System.out.println("Exiting from Thread C ...");

}

}

public class Main {

public static void main(String args[]) {

ThreadA a = new ThreadA();

ThreadB b = new ThreadB();

ThreadC c = new ThreadC();

a.start();

b.start();

c.start();

System.out.println("... Multithreading is over ");

}

}

### Output:-

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
... Multithreading is over  
From Thread A with i = -1  
From Thread A with i = -2  
From Thread B with j= 2  
From Thread B with j= 4  
From Thread C with k = 1  
From Thread C with k = 3  
From Thread C with k = 5  
From Thread C with k = 7  
From Thread B with j= 6  
From Thread A with i = -3  
From Thread B with j= 8  
From Thread B with j= 10  
Exiting from Thread B ...  
From Thread C with k = 9  
From Thread A with i = -4  
Exiting from Thread C ...  
From Thread A with i = -5  
Exiting from Thread A ...  
PS C:\Users\ravik\Desktop\Assignments\JavaLab> █
```

**Result:-** Implement concept of multithreading in Java by

a) Extending Thread class

b) Implementing Runnable interface

is compiled and executed successfully

## Experiment 15

**Aim:-** Write a java code to implement the concept of simple inheritance, multilevel inheritance, and hierarchical inheritance

**Algorithm:-** Find the object in the knowledge base

If there is a value for the attribute report it

Otherwise look for a value of instance if none fail

Otherwise go to that node and find a value for the attribute and then report it

Otherwise search through using isa until a value is found for the attribute.

**Program:-** Simple Inheritance:- class Animal{

```
void eat(){System.out.println("eating...");}
```

```
}
```

```
class Dog extends Animal{
```

```
void bark(){System.out.println("barking...");}
```

```
}
```

```
class Main{
```

```
public static void main(String args[]){
```

```
Dog d=new Dog();
```

```
d.bark();
```

```
d.eat();
```

```
}}
```

Multilevel Inheritance:- class Animal{

```
void eat(){System.out.println("eating...");}
```

```
}
```

```
class Dog extends Animal{
```

```
void bark(){System.out.println("barking...");}
```

```
}
```

```
class BabyDog extends Dog{
```

```
void weep(){System.out.println("weeping...");}
```

```
}
```

```
class Main{
```

```
public static void main(String args[]){
```

```
BabyDog d=new BabyDog();
```

```
d.weep();
```

```
d.bark();
```

```
d.eat();
```

```
}}
```

Hierarchical Inheritance:- class Animal{

```
void eat(){System.out.println("eating...");}
```

```
}
```

```
class Dog extends Animal{
```

```
void bark(){System.out.println("barking...");}
```

```
}
```

```
class Cat extends Animal{
```

```
void meow(){System.out.println("meowing...");}
```

```
}
```

```
class Main{
```

```
public static void main(String args[]){
```

```
Cat c=new Cat();  
c.meow();  
c.eat();  
//c.bark();//C.T.Error  
}}
```

**Output:-**

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
meowing...  
eating...  
PS C:\Users\ravik\Desktop\Assignments\JavaLab>
```

**Result:-** Java code to implement the concept of simple inheritance, multilevel inheritance, and hierarchical inheritance is compiled and executed successfully.

## Experiment 16

**Aim:-** Implementation of Linear data structure using Programming language.

**Algorithm:-** Stack:- begin

if stack is full

return

endif

else

increment top

stack[top] assign value

end else

end procedure

Queue:- IF REAR = MAX - 1

Write OVERFLOW

Go to step

[END OF IF]

Step 2: IF FRONT = -1 and REAR = -1

SET FRONT = REAR = 0

ELSE

SET REAR = REAR + 1

[END OF IF]

Step 3: Set QUEUE[REAR] = NUM

Step 4: EXIT

**Program:-**

**Stack:-**

```
import java.util.Stack;
```

```
public class Main
```

```
{
```

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

```
{
```

```
//creating an instance of Stack class
```

```
Stack<Integer> stk= new Stack<>();
```

```
// checking stack is empty or not
```

```
boolean result = stk.empty();
```

```
System.out.println("Is the stack empty? " + result);
```

```
// pushing elements into stack
```

```
stk.push(78);
```

```
stk.push(113);
```

```
stk.push(90);
```

```
stk.push(120);
```

```
//prints elements of the stack
```

```
System.out.println("Elements in Stack: " + stk);
```

```
result = stk.empty();
```

```
System.out.println("Is the stack empty? " + result);
```

```
System.out.println("Written by Ravi Kumar...");
```

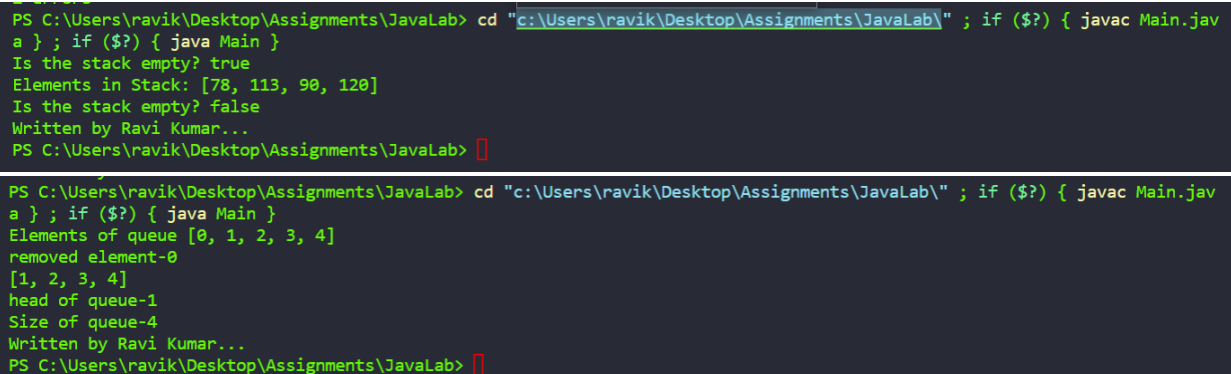
```
}
```

```
}
```

### Queue:-

```
import java.util.LinkedList;
import java.util.Queue;
public class Main {
    public static void main(String[] args)
    {
        Queue<Integer> q = new LinkedList<>();
        for (int i = 0; i < 5; i++)
            q.add(i);
        System.out.println("Elements of queue " + q);
        int removedele = q.remove();
        System.out.println("removed element-" + removedele);
        System.out.println(q);
        int head = q.peek();
        System.out.println("head of queue-" + head);
        int size = q.size();
        System.out.println("Size of queue-" + size);
        System.out.println("Written by Ravi Kumar...");
    }
}
```

### Output:-



```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Is the stack empty? true
Elements in Stack: [78, 113, 90, 120]
Is the stack empty? false
Written by Ravi Kumar...
PS C:\Users\ravik\Desktop\Assignments\JavaLab>

PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Elements of queue [0, 1, 2, 3, 4]
removed element-0
[1, 2, 3, 4]
head of queue-1
Size of queue-4
Written by Ravi Kumar...
PS C:\Users\ravik\Desktop\Assignments\JavaLab>
```

**Result:-** Implementation of Linear data structure using Programming language is compiled and executed successfully.

## Experiment 17

**Aim:-** Implementation of Non- Linear data structure using Programming language

**Algorithm:-** If root is NULL

then create root node

return

If root exists then

compare the data with node.data

while until insertion position is located

If data is greater than node.data

goto right subtree

else

goto left subtree

endwhile

insert data

end If

**Program:-** class Node {

int key;

Node left, right;

public Node(int item) {

key = item;

left = right = null;}}

class Main {

Node root;

public void traverseTree(Node node) {

if (node != null) {

traverseTree(node.left);

System.out.print(" " + node.key);

traverseTree(node.right);}

public static void main(String[] args) {

Main tree = new Main();

tree.root = new Node(1);

tree.root.left = new Node(2);

tree.root.right = new Node(3);

tree.root.left.left = new Node(4);

System.out.print("\nBinary Tree: ");

tree.traverseTree(tree.root);

System.out.println(" ");

System.out.println("Written by Ravi Kumar...");}}

**Output:-**

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
  
Binary Tree: 4 2 1 3  
Written by Ravi Kumar...  
PS C:\Users\ravik\Desktop\Assignments\JavaLab> █
```

**Result:-** Implementation of Non- Linear data structure using Programming language is compiled and executed successfully.

## Experiment 18

**Aim:-** Implementation of searching and sorting techniques using Programming language

**Algorithm:-**

**Linear Search:-**

Step 1: Set i to 1

Step 2: if i > n then go to step 7

Step 3: if A[i] = x then go to step 6

Step 4: Set i to i + 1

Step 5: Go to Step 2

Step 6: Print Element x Found at index i and go to step 8

Step 7: Print element not found

Step 8: Exit

**Bubble Sort:-** begin BubbleSort(list)

for all elements of list

if list[i] > list[i+1]

swap(list[i], list[i+1])

end if

end for

return list

**Program:-** Linear Search:-

```
public class Main{
    public static int linearSearch(int[] arr, int key){
        for(int i=0;i<arr.length;i++){
            if(arr[i] == key){
                return i;
            }
        }
        return -1;
    }
    public static void main(String a[]){
        int[] a1= {10,20,30,50,70,90};
        int key = 70;
        System.out.println(key+" is found at index: "+linearSearch(a1, key));
    }
}
```

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java
a } ; if ($?) { java Main }
70 is found at index: 4
PS C:\Users\ravik\Desktop\Assignments\JavaLab> 
```

**Bubble Sort:-**

```
public class Main {
    static void bubbleSort(int[] arr) {
        int n = arr.length;
        int temp = 0;
        for(int i=0; i < n; i++){
            for(int j=1; j < (n-i); j++){
                if(arr[j-1] > arr[j]){
                    //swap elements
                }
            }
        }
    }
}
```



```

        temp = arr[j-1];
        arr[j-1] = arr[j];
        arr[j] = temp;
    }

}

}

}

public static void main(String[] args) {
    int arr[] = {3,60,35,2,45,320,5};

    System.out.println("Array Before Bubble Sort");
    for(int i=0; i < arr.length; i++){
        System.out.print(arr[i] + " ");
    }
    System.out.println();

    bubbleSort(arr);//sorting array elements using bubble sort

    System.out.println("Array After Bubble Sort");
    for(int i=0; i < arr.length; i++){
        System.out.print(arr[i] + " ");
    }

}

}

```

**Output:-**

```

PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Array Before Bubble Sort
3 60 35 2 45 320 5
Array After Bubble Sort
2 3 5 35 45 60 320
PS C:\Users\ravik\Desktop\Assignments\JavaLab>

```

**Result:-** Implementation of searching and sorting techniques using Programming language is completed successfully

## Experiment 19

**Aim:-** Implementation of Graph traverses using Programming language.

**Program:-** DFS:-

```
import java.io.*;
import java.util.*;
class Main {
    private int V;
    private LinkedList<Integer> adj[];
    @SuppressWarnings("unchecked") Main(int v)
    {
        V = v;
        adj = new LinkedList[v];
        for (int i = 0; i < v; ++i)
            adj[i] = new LinkedList();
    }
    void addEdge(int v, int w)
    {
        adj[v].add(w); // Add w to v's list.
    }
    void DFSUtil(int v, boolean visited[])
    {
        visited[v] = true;
        System.out.print(v + " ");
        Iterator<Integer> i = adj[v].listIterator();
        while (i.hasNext()) {
            int n = i.next();
            if (!visited[n])
                DFSUtil(n, visited);
        }
    }
    // DFSUtil()
    void DFS(int v)
    {
        boolean visited[] = new boolean[V];
        DFSUtil(v, visited);
    }
    public static void main(String args[])
    {
        Main g = new Main(4);
        g.addEdge(0, 1);
        g.addEdge(0, 2);
        g.addEdge(1, 2);
        g.addEdge(2, 0);
        g.addEdge(2, 3);
        g.addEdge(3, 3);
        System.out.println("Following is Depth First Traversal " + "(starting from vertex 2)");
        g.DFS(2);
    }
}
```

```

    }
}

PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Following is Depth First Traversal (starting from vertex 2)
2 0 1 3

```

### BFS:-

```

import java.io.*;
import java.util.*;
class Main
{
    private int V; // No. of vertices
    private LinkedList<Integer> adj[];
    Main(int v)
    {
        V = v;
        adj = new LinkedList[v];
        for (int i=0; i<v; ++i)
            adj[i] = new LinkedList();
    }
    void addEdge(int v,int w)
    {
        adj[v].add(w);
    }
    void BFS(int s)
    {
        boolean visited[] = new boolean[V];
        LinkedList<Integer> queue = new LinkedList<Integer>();
        visited[s]=true;
        queue.add(s);

        while (queue.size() != 0)
        {
            s = queue.poll();
            System.out.print(s+" ");
            Iterator<Integer> i = adj[s].listIterator();
            while (i.hasNext())
            {
                int n = i.next();
                if (!visited[n])
                {
                    visited[n] = true;
                    queue.add(n);
                }
            }
        }
    }
}

```

```

public static void main(String args[])
{
    Main g = new Main(4);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    System.out.println("Following is Breadth First Traversal "+
                        "(starting from vertex 2)");

    g.BFS(2);
}
}

```

**Output:-**

```

PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java
a } ; if ($?) { java Main }
Note: Main.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
PS C:\Users\ravik\Desktop\Assignments\JavaLab> 

```

**Result:-** Implementation of Graph traverses using Programming language is completed successfully.

## Experiment 20

**Aim:-** Implementation of Non- Linear data structure using Programming language

**Algorithm:-** If root is NULL

then create root node

return

If root exists then

compare the data with node.data

while until insertion position is located

If data is greater than node.data

goto right subtree

else

goto left subtree

endwhile

insert data

end If

**Program:-**

```
class Node {
    int key;
    Node left, right;
    public Node(int item) {
        key = item;
        left = right = null; }
}

class Main {
    Node root;
    public void traverseTree(Node node) {
        if (node != null) {
            traverseTree(node.left);
            System.out.print(" " + node.key);
            traverseTree(node.right); }
    }

    public static void main(String[] args) {
        Main tree = new Main();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        System.out.print("\nBinary Tree: ");
        tree.traverseTree(tree.root);
        System.out.println(" ");
        System.out.println("Written by Ravi Kumar..."); } }
```

**Output:-**

```
PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }

Binary Tree:  4 2 1 3
Written by Ravi Kumar...
PS C:\Users\ravik\Desktop\Assignments\JavaLab> 
```

**Result:-** Implementation of Trees Concepts using Programming language is completed successful.

## Experiment 21

**Aim:-** Implementation of Shortest-path algorithms Concepts.

**Algorithm:-** function dijkstra(G, S)

```
for each vertex V in G
    distance[V] <- infinite
    previous[V] <- NULL
    If V != S, add V to Priority Queue Q
distance[S] <- 0

while Q IS NOT EMPTY
    U <- Extract MIN from Q
    for each unvisited neighbour V of U
        tempDistance <- distance[U] + edge_weight(U, V)
        if tempDistance < distance[V]
            distance[V] <- tempDistance
            previous[V] <- U
return distance[], previous[]
```

**Program:-**

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

class Main {
    static final int V = 9;
    int minDistance(int dist[], Boolean sptSet[])
    {
        // Initialize min value
        int min = Integer.MAX_VALUE, min_index = -1;

        for (int v = 0; v < V; v++)
            if (sptSet[v] == false && dist[v] <= min) {
                min = dist[v];
                min_index = v;
            }

        return min_index;
    }

    // A utility function to print the constructed distance array
    void printSolution(int dist[])
    {
        System.out.println("Vertex \t\t Distance from Source");
        for (int i = 0; i < V; i++)
            System.out.println(i + "\t\t " + dist[i]);
    }

    void dijkstra(int graph[][], int src)
    {

```

```

int dist[] = new int[V];
Boolean sptSet[] = new Boolean[V];
for (int i = 0; i < V; i++) {
    dist[i] = Integer.MAX_VALUE;
    sptSet[i] = false;
}
dist[src] = 0;
for (int count = 0; count < V - 1; count++) {
    int u = minDistance(dist, sptSet);
    sptSet[u] = true;
    for (int v = 0; v < V; v++)
        if (!sptSet[v] && graph[u][v] != 0 && dist[u] != Integer.MAX_VALUE &&
dist[u] + graph[u][v] < dist[v])
            dist[v] = dist[u] + graph[u][v];
}
printSolution(dist);
}

public static void main(String[] args){
    int graph[][] = new int[][] { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },
                                     { 4, 0, 8, 0, 0, 0, 0, 11, 0 },
                                     { 0, 8, 0, 7, 0, 4, 0, 0, 2 },
                                     { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
                                     { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
                                     { 0, 0, 4, 14, 10, 0, 2, 0, 0 },
                                     { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
                                     { 8, 11, 0, 0, 0, 0, 1, 0, 7 },
                                     { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

    Main t = new Main();
    t.dijkstra(graph, 0);
}
}

```

**Output:-**

```

PS C:\Users\ravik\Desktop\Assignments\JavaLab> cd "c:\Users\ravik\Desktop\Assignments\JavaLab\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Vertex      Distance from Source
0           0
1           4
2          12
3          19
4          21
5          11
6           9
7           8
8          14
PS C:\Users\ravik\Desktop\Assignments\JavaLab>

```

**Result:-** Implementation of Shortest-path algorithms Concepts is completed successfully.