**Java**

Java was developed by James Gosling and his team , from Sun Microsystems in 1995 as an object-oriented language for general-purpose business applications and for interactive, Web-based Internet applications

**Features of JAVA:**

* **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
* **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
* **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
* **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
* **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
* **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
* **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
* **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
* **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
* **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
* **Distributed** − Java is designed for the distributed environment of the internet.
* **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

**Tools Need:**

Computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).

also need the following software −

* Linux 7.1 or Windows xp/7/8 operating system
* Java JDK 8
* Microsoft Notepad or any other text editor

|  |  |
| --- | --- |
| **Comparison Between c++ and Java** | |
| **C++** | **Java** |
| C++ is platform-dependent. | Java is platform-independent. |
| C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications. |
| C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java. |
| C++ supports operator overloading. | Java doesn't support operator overloading. |
| C++ supports pointers. You can write pointer program in C++. | Java supports pointer internally. But you can't write the pointer program in java. It means java has restricted pointer support in java. |
| C++ uses compiler only. | Java uses compiler and interpreter both. |
| C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. |
| C++ supports structures and unions. | Java doesn't support structures and unions. |

How to write the simple program of java. We can write a simple hello java program easily after installing the JDK.

To create a simple java program, you need to create a class that contains main method **in any text editor (notepad,notepad++,wordpad etc)**

**Creating hello java example**

class Simple{

    public static void main(String args[ ]){

     System.out.println("Hello Java");

    }

}

**Program Execution:**

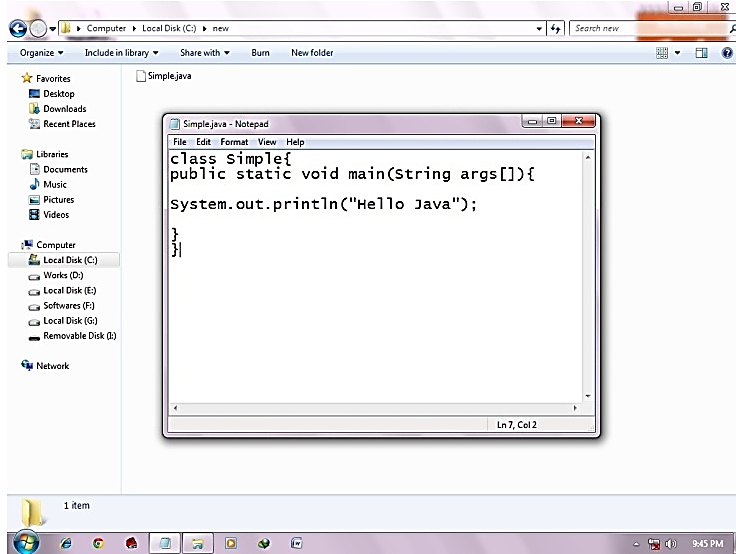
Save the above Program as **Simple.java,** To compile and run this program, you need to open command prompt by **start menu -> All Programs -> Accessories -> command prompt**.

|  |  |
| --- | --- |
| **To compile:** | javac Simple.java |
| **To execute:** | java Simple |

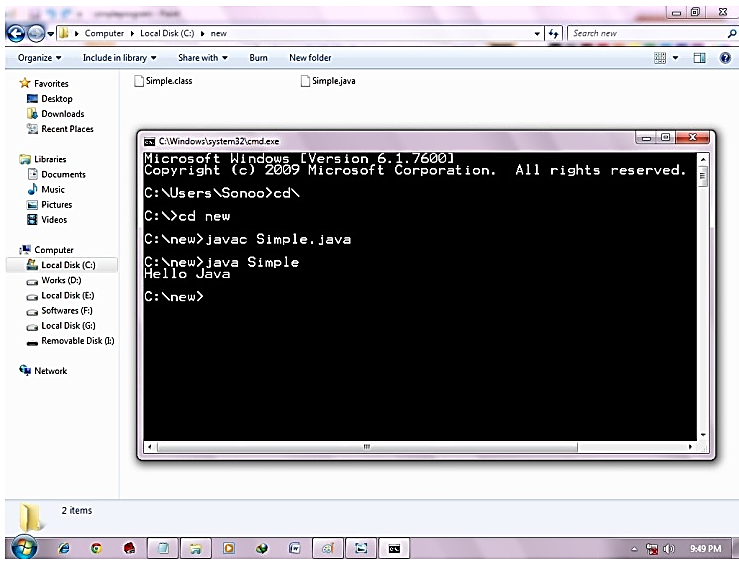
**Parameters used in first java program**

* **class** keyword is used to declare a class in java.
* **public** keyword is an access modifier which represents visibility, it means it is visible to all.
* **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require creating object to invoke the main method. So it saves memory.
* **void** is the return type of the method, it means it doesn't return any value.
* **main** represents the starting point of the program.
* **String[ ] args** is used for command line argument.
* **System.out.println()** is used print statement.

To write the simple program, open notepad by **start menu -> All Programs -> Accessories -> notepad** and write simple program as displayed below:



As displayed in the above diagram, write the simple program of java in notepad and saved it as Simple.java. To compile and run this program, you need to open command prompt by **start menu -> All Programs -> Accessories -> command prompt**.



**Basic Structure of JAVA Program:**

A Java program involves the following sections:

|  |
| --- |
| * Documentation Section * Package Statement * Import Statements * Interface Statement * Class Definition * Main Method Class   + Main Method Definition |

# How to set path in Java:

To set the temporary path of JDK, you need to follow following steps:

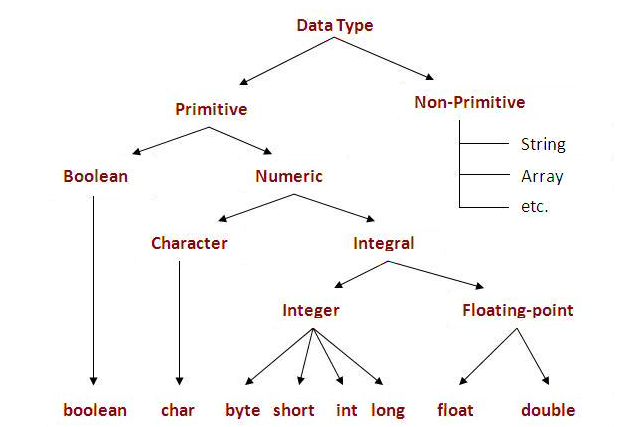
* Open command prompt
* Copy the path of jdk/bin directory
* Write in command prompt: set path=copied\_path

**For Example:**

**set path=C:\Program Files\Java\jdk1.6.0\_23\bin**

# Variables and Data Types in Java

A variable is a container which holds the value while the java program is executed. A variable is assigned with a datatype.

There are two types of data types in java: **primitive and non-primitive.**

# Operators in java:

**Operator** in java is a symbol that is used to perform operations.

There are many types of operators in java which are given below:

* Unary Operator [ ++, -- ]
* Arithmetic Operator, [ +, - , \* , /, %]
* Relational Operator [ >, >=, <, <=, = =,! =]
* Logical Operator [&& , || , !]
* Ternary Operator [ ? :]
* Assignment Operator [ =]

**Java If-else Statement**

The Java *if statement* is used to test the condition. It checks Boolean condition: *true* or *false*. There are various types of if statement in java.

* if statement
* if-else statement
* if-else-if ladder
* nested if statement

## Java if Statement: The Java if statement tests the condition. It executes the if block if condition is true.

**Syntax: if(condition) {   //code to be executed }**

**Example**: public class IfExample

{ public static void main(String[] args)

{ int age=20;

if(age>18){ System.out.print("Eligible to VOTE"); }

} }

## Java if-else Statement:The Java if-else statement also tests the condition. It executes the *if block* if condition is true otherwise *else block* is executed.

**Syntax: if(condition){  //code if condition is true  }**

**else{  //code if condition is false  }**

**Example:** public class IfElseExample {

public static void main(String[] args) {

int number=13;

if(number%2==0){ System.out.println("even number"); }

else{ System.out.println("odd number"); }

} }

## Java if-else-if ladder Statement: The if-else-if ladder statement executes one condition from multiple statements.

**Syntax: if(condition1)**

**{  //code to be executed if condition1 is true**

**}else if(condition2){**

**//code to be executed if condition2 is true**

**}**

**else if(condition3){**

**//code to be executed if condition3 is true**

**}**

**...**

**else{**

**//code to be executed if all the conditions are false**

**}**

**Example:**

public class Test {

public static void main(String args[]) {

int x = 30;

if( x == 10 ) {

System.out.print("Value of X is 10");

}else if( x == 20 ) {

System.out.print("Value of X is 20");

}else if( x == 30 ) {

System.out.print("Value of X is 30");

}else {

System.out.print("This is else statement");

}

}

}

# Java Switch Statement

**Syntax: switch(expression){**

**case value1:**

**//code to be executed;**

**break;**

**case value2:**

**//code to be executed;**

**break;**

**......**

**default:**

**code to be executed if all cases are not matched;**

**}**

**Example:** public class Test {

public static void main(String args[]) {

char grade = 'C';

switch(grade) {

case 'A' :

System.out.println("Excellent!");

break;

case 'B' :

case 'C' :

System.out.println("Well done");

break;

case 'D' :

System.out.println("You passed");

case 'F' :

System.out.println("Better try again");

break;

default :

System.out.println("Invalid grade");

}

System.out.println("Your grade is " + grade);

}}

**Loops in Java**

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true. There are three types of loops in java.

* for loop
* while loop
* do-while loop

**Java For Loop**

**Syntax: for(initialization;condition;incr/decr){**

**//code to be executed**

**}**

**Example:** public class ForExample {

public static void main(String[] args) {

for(int i=1;i<=10;i++){

System.out.println(i);

}

}

}

**Java While Loop**

**Syntax: while(condition){**

**//code to be executed**

**}**

**Example:**

public class WhileExample {

public static void main(String[] args) {

    int i=1;

    while(i<=10){

        System.out.println(i);

    i++;

    }

}

}

**Java Do While Loop**

**Syntax:**

**do{**

**//code to be executed**

**}while(condition);**

**Example:**

public class DoWhileExample {

public static void main(String[] args) {

    int i=1;

    do{

        System.out.println(i);

    i++;

    }while(i<=10);

}

}

# Java Comments

The java comments are statements that are not executed by the compiler and interpreter. The comments can be used to provide information or explanation about the variable, method, class or any statement. It can also be used to hide program code for specific time.

## Types of Java Comments: There are 3 types of comments in java.

1. Single Line Comment //This is single line comment
2. Multi Line Comment

/\* This  is

multi line  comment \*/

1. Documentation Comment

/\*\*

This

is  documentation

comment

\*/

**\*Example Programs:**

1. // Fibonacci series program in java without using recursion

class FibonacciExample{

public static void main(String args[])

{ int n1=0,n2=1,n3,i,count=10;

System.out.print(n1+" "+n2);//printing 0 and 1

for(i=2;i<count;++i)//loop starts from 2 because 0 and 1 are already printed

{ n3=n1+n2;

System.out.print(" "+n3);

n1=n2;

n2=n3;

} }}

2./\* prime number program in java. In this java program, we will take a number variable and check whether the number is prime or not.\*/

public class PrimeExample{

public static void main(String args[]){

int i,m=0,flag=0;

int n=3;//it is the number to be checked

m=n/2;

if(n==0||n==1){

System.out.println(n+" is not prime number");

}else{

for(i=2;i<=m;i++){

if(n%i==0){

System.out.println(n+" is not prime number");

flag=1;

break; } }

if(flag==0) { System.out.println(n+" is prime number"); }

}//end of else

} }

3.// factorial Program using loop in java.

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);

 }

}

## OOPs (Object Oriented Programming System)

**Object** means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

**Object:** Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

**Class: Collection of objects** is called class. It is a logical entity.

**Inheritance: When one object acquires all the properties and behaviors of parent object** i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**Polymorphism:** When **one task is performed by different ways** i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

**Abstraction: Hiding internal details and showing functionality** is known as abstraction. For example: phone call, we don't know the internal processing. In java, we use abstract class and interface to achieve abstraction.

**Encapsulation: Binding (or wrapping) code and data together into a single unit is known as encapsulation**. For example: capsule, it is wrapped with different medicines.

### Class in Java

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

### Syntax to declare a class:

**class <class\_name>{**

**field;**

**method;**

**}**

**Example:** 1)

class Student{

int id =100;//field or data member or instance variable

String name;

public static void main(String args[]){

Student s1=new Student();//creating an object of Student

System.out.println(s1.id);//accessing member through reference variable

System.out.println(s1.name); } }

### Object and Class

### Example: Rectangle

class Rectangle{

int length;

int width;

void insert(int l, int w){

length=l;

width=w;

}

void calculateArea(){System.out.println(length\*width);}

}

class TestRectangle1{

public static void main(String args[]){

Rectangle r1=new Rectangle();

Rectangle r2=new Rectangle();

r1.insert(11,5);

r2.insert(3,15);

r1.calculateArea();

r2.calculateArea(); } }

# Inheritance in Java

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

### Terms used in Inheritence

* **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
* **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
* **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
* **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in previous class.

### Syntax of Deriving a new class:

**class Subclass-name extends Superclass-name**

**{**

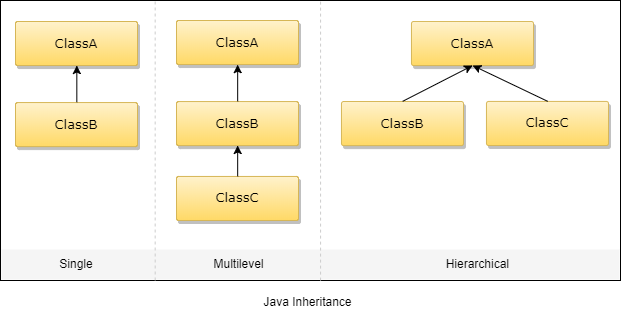
**//methods and fields**

**}**

## Types of inheritance in java

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.[In java programming, multiple and hybrid inheritance is supported through interface only.]

***Note: Multiple inheritance is not supported in java through class.***

******

1. **Single Inheritance**

When a single class gets derived from its base class, then this type of inheritance is termed as single inheritance.

**Example:**

class Teacher {

void teach() {

System.out.println("Teaching subjects");

}

}

class Students extends Teacher {

void listen() {

System.out.println("Listening to teacher");

}

}

class CheckForInheritance {

public static void main(String args[]) {

Students s1 = new Students();

s1.teach();

s1.listen();

}

}

1. **Multi-Level Inheritance:** In this type of inheritance, a derived class gets created from another derived class and can have any number of levels.

**Example** : class Shape {

   public void display() {

      System.out.println("Inside display");

   }

}

class Rectangle extends Shape {

   public void area() {

      System.out.println("Inside area");

   }

}

class Cube extends Rectangle {

   public void volume() {

      System.out.println("Inside volume");

   }

}

public class Tester {

   public static void main(String[] arguments) {

      Cube cube = new Cube();

      cube.display();

      cube.area();

      cube.volume();

   }

}

1. **Hierarchical Inheritance:** In this type of inheritance, there are more than one derived classes which get created from one single base class.

**Example:**

class A

{

public void methodA()

{

System.out.println("method of Class A");

}

}

class B extends A

{

public void methodB()

{

System.out.println("method of Class B");

}

}

class C extends A

{

public void methodC()

{

System.out.println("method of Class C");

}

}

class D extends A

{

public void methodD()

{

System.out.println("method of Class D");

}

}

class JavaExample

{

public static void main(String args[])

{

B obj1 = new B();

C obj2 = new C();

D obj3 = new D();

//All classes can access the method of class A

obj1.methodA();

obj2.methodA();

obj3.methodA();

}

}

# Java Package

A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

## Simple example of java package

The **package keyword** is used to create a package in java.

**//save as Simple.java**

package mypack;

public class Simple{

 public static void main(String args[]){

    System.out.println("Welcome to package");

   }

}

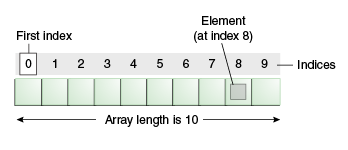
# Access Modifiers in java

**There are 4 types of java access modifiers:**

1. **private:** The private access modifier is accessible only within class.
2. **default:** If you don't use any modifier, it is treated as **default** bydefault. The default modifier is accessible only within package.
3. **protected:** The **protected access modifier** is accessible within package and outside the package but through inheritance only. The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.
4. **public:** The **public access modifier** is accessible everywhere. It has the widest scope among all other modifiers.

# Java Array

# Java array is an object the contains elements of similar data type. It is a data structure where we store similar elements. We can store only fixed set of elements in a java array.

Array in java is index based, first element of the array is stored at 0 index.

### Advantage of Java Array

* **Code Optimization:** It makes the code optimized, we can retrieve or sort the data easily.
* **Random access:** We can get any data located at any index position.

### Types of Array in java

There are two types of array.

* Single Dimensional Array
* Multidimensional Array

### Single dimensional array in java

### Syntax to Declare an Array in java

dataType[ ]  arr;  ( or)

dataType  []arr; (or)

dataType  arr[];

### Instantiation of an Array in java

arrayRefVar=new datatype[size];

**Example:**

class Testarray{

public static void main(String args[]){

int a[]=new int[5];//declaration and instantiation

a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

//printing array

for(int i=0;i<a.length;i++)//length is the property of array

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

}}

## Declaration, Instantiation and Initialization of Java Array

class Testarray1{

public static void main(String args[]){

int a[]={33,3,4,5};//declaration, instantiation and initialization

//printing array

for(int i=0;i<a.length;i++)//length is the property of array

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

}}

### Multidimensional array in java

### Syntax to Declare Multidimensional Array in java

**dataType[ ][ ]  arrayRefVar; (or)**

**dataType  [ ][ ]arrayRefVar; (or)**

**dataType  arrayRefVar[][]; (or)**

### Instantiation of an Multi Dimensional Array in java

**int[][] arr=new int[3][3];//3 row and 3 column**

**Example:**

class Testarray3{

public static void main(String args[]){

//declaring and initializing 2D array

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

//printing 2D array

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

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

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

}

System.out.println();

}

}}

# Java String

Generally, string is a sequence of characters. But in java, string is an object that represents a sequence of characters. The java.lang.String class is used to create string object.

**Example:**

1. char[] ch={'j','a','v','a','t','p','o','i','n','t'};
2. String s=new String(ch);

**Java String** class provides a lot of methods to perform operations on string such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

There are two ways to create String object:

* + 1. By string literal🡪 String  s="welcome";
  1. By new keyword🡪 String s=new String("Welcome");

**Example:**

public class StringExample{

public static void main(String args[ ]){

String s1="java";

char ch[ ]={'s','t','r','i','n','g','s'};

String s2=new String(ch);

String s3=new String("example");

System.out.println(s1);

System.out.println(s2);

System.out.println(s3);

}}

# Exception Handling in Java

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 ClassNotFound, IO etc.

## Java try block

Java try block is used to enclose the code that might throw an exception. It must be used within the method.Java try block must be followed by either catch .

## Syntax of java try-catch

try{

//code that may throw exception

}catch(Exception\_class\_Name ref){ }

**Example:**

public class Testtrycatch2{

public static void main(String args[]){

try{

int data=50/0;

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

System.out.println("rest of the code...");

}

}

**Compile by: javac Testtrycatch2.java**

**Run by: java Testtrycatch2**

java.lang.ArithmeticException: / by zero  
rest of the code...