CORE JAVA

**What is Java**

Java is a **programming language** and a **platform**.

Java is a high level, robust, secured and object-oriented programming language.

## Where it is used?

According to Sun, 3 billion devices run java. There are many devices where Java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus etc.
2. Web Applications such as irctc.co.in, javatpoint.com etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games etc.

## Types of Java Applications

There are mainly 4 types of applications that can be created using java programming:

#### 1) Standalone Application

It is also known as desktop application or window-based application. An application that we need to install on every machine such as media player, antivirus etc. AWT and Swing are used in java for creating standalone applications.

#### 2) Web Application

An application that runs on the server side and creates dynamic page, is called web application. Currently, servlet, jsp, struts, jsf etc. technologies are used for creating web applications in java.

#### 3) Enterprise Application

An application that is distributed in nature, such as banking applications etc. It has the advantage of high level security, load balancing and clustering. In java, EJB is used for creating enterprise applications.

#### 4) Mobile Application

An application that is created for mobile devices. Currently Android and Java ME are used for creating mobile applications.

## Java Platforms / Editions

There are 4 platforms or editions of Java:

#### 1) Java SE (Java Standard Edition)

It is a java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection etc.

#### 2) Java EE (Java Enterprise Edition)

It is an enterprise platform which is mainly used to develop web and enterprise applications. It is built on the top of Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA etc.

#### 3) Java ME (Java Micro Edition)

It is a micro platform which is mainly used to develop mobile applications.

#### 4) JavaFx

It is used to develop rich internet applications. It uses light-weight user interface API.

# Features of Java

There is given many features of java. They are also known as java buzzwords. The Java Features given below are simple and easy to understand.

Java Features

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Dynamic
9. Interpreted
10. High Performance
11. Multithreaded
12. Distributed

### Simple

|  |
| --- |
| According to Sun, Java language is simple because: |
| syntax is based on C++ (so easier for programmers to learn it after C++). |
| removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc. |
| No need to remove unreferenced objects because there is Automatic Garbage Collection in java. |

### Object-oriented

|  |
| --- |
| Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour. |
| Object-oriented programming(OOPs) is a methodology that simplify software development and maintenance by providing some rules. |
| Basic concepts of OOPs are: |
| 1. Object 2. Class 3. Inheritance 4. Polymorphism 5. Abstraction 6. Encapsulation  Platform Independent java is platform independent  A platform is the hardware or software environment in which a program runs.  There are two types of platforms software-based and hardware-based. Java provides software-based platform.  The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:   1. Runtime Environment 2. API(Application Programming Interface)   Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA). Secured Java is secured because:   * **No explicit pointer** * **Java Programs run inside virtual machine sandbox**   how java is secured   * **Classloader:** adds security by separating the package for the classes of the local file system from those that are imported from network sources. * **Bytecode Verifier:** checks the code fragments for illegal code that can violate access right to objects. * **Security Manager:** determines what resources a class can access such as reading and writing to the local disk.   These security are provided by java language. Some security can also be provided by application developer through SSL, JAAS, Cryptography etc. Robust Robust simply means strong. Java uses strong memory management. There are lack of pointers that avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points make java robust. Architecture-neutral There is no implementation dependent features e.g. size of primitive types is fixed.  In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures. Portable We may carry the java bytecode to any platform. High-performance  |  |  | | --- | --- | | Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++) Distributed  |  | | --- | | We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet. |  Multi-threaded A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications etc. | |

# Simple Program of Java

### Requirement for Hello Java Example

|  |
| --- |
| For executing any java program, you need to   * install the JDK if you don't have installed it, [download the JDK](http://www.oracle.com/technetwork/java/javase/downloads/index.html) and install it. * set path of the jdk/bin directory. <http://www.javatpoint.com/how-to-set-path-in-java> * create the java program * compile and run the java program |

### Creating hello java example

Let's create the hello java program:

1. **class** Simple{
2. **public** **static** **void** main(String args[]){
3. System.out.println("Hello Java");
4. }
5. }

Understanding first java program

Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

* **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 startup of the program.
* **String[] args** is used for command line argument. We will learn it later.
* **System.out.println()** is used print statement. We will learn about the internal working of System.out.println statement later.

**Variable:**

* It is a name of memory location to store values.
* We use Variable to store data
* Local Variables contain garbage values.
* Global variables contain null values/default values.

**Global Variables:**  We declare global variables outside methods including main methods. We can use global variables in any method of that particular Class. But we have to create “Object” for that particular global variable which one we are calling in methods.

**Local Variable:** We declare local variables inside methods only. We can use these variables in only that particular method in which one we created. We cannot use in other or main method also.

**Data Type:**

i) Primitive

Boolean, char, byte, short, int, long, float, double..

ii)Non-Primitive

* These stores only address of an “Object”.

Strings

Array

Classes

Interfaces

**Class:**

* It is an entity which binds data member and member methods into single unit.
* Class is a best example of encapsulation
* A class is a blue print or a template for creating different objects which defines its properties and behaviours.

**Methods:**

* Method is just a block of java statement that performs a particular task.
* We can reuse the code and avoid duplicity.

Method can be called using (.) dot operator.

Access-specifier static/non static return type method name ()

{

}

1-public void marks ()

{

}

2- Public static void main marks ()

}

**Objects :**

* It’s an instance of Class.
* An entity that has and behaviour is known as an object.
* Cloning of Class.
* A single Class can Creates any number of unique objects.

**We should define object by “New”**

**What object knows is Variable**

**What object does is Method**

ClassName objectname= new classname ();

* For non-static method can call through object using dot (.)
* For static method can call using class name using dot (.)

**Java Heap:**

It’s nothing but the memory space taken by the **jvm** from **OS.**

**JDK contains Java Compiler + JRE**

**JRE contains JVM + Library Files.**

**Garbage Collection:**

It’s provided by JVM. When an object don’t have reference variable, it goes into garbage.

**Reference variable:**

Prtp obj = new prtp();

Prtp **obj;**

Reference Variable for class.

**Static and Non static :**

**Static:**  When you mention “static” for method or variable is convers as a static method or variable.

Public class addng

{

String name = “BPR”

Static int age = “25”

Public void sum(){

s.o.p(“”)

}

Public static void mul()

{

S.o.p(“”)

}

Public ststic void main()

{

}}

We can call static method / variable by:

* **Direct calling:**

mul(); // method

s.o.p (age) // variable

* **By class Name:**

Addng . mul(); // method

s.o.p(Adding.age).

Scanner validates input data.

**Handling exceptions with try/catch/ blocks:**

Handling exceptions is a mechanism to handle Exceptions.

Common Scenarios:

i) Something divided by Zero

ii) Where Nullpoint exception

Ex:

String S= null; (When you try to find length of String);

iii) Number format Exception

Ex: String s= “abcd”;

Int a = integer.parseint(s);

iv) When Array index out of bounce:

**Try:**

**Final/Finally/Finalize:**

**Final: (Key Word)**

* For constant values(variables)
* To prevent overriding
* To prevent inheritance

**Finally (Block):**

Code in finally block will be executed even if we have exception or no exception

**Finalize(Method):**

It’s related to garbage collector. Its use to clean up memory processing.

**String:**

* It is a Class in java not a data type.

String s = new String ();

* String Concatenation is a way to combine two or more strings into a single string by ‘+’ operator.

Ex:

String s1= “iam”;

String s2= “Prathap”

String s3 =s1+s2;

**String Class:**

* String class has many in-built methods, like comparing, split, concatenate, convert.
* In java Object class is Parent for all classes, so String class inherits object class.

**Constructor:**

AnyMethod created with class name and doesn’t have any return type we can call that as constructor. Constructor executes after object created. It allows us to initialize the object.

***OOPS Concepts:***

*i)* ***Polymorphism*:**  a object oriented programming feature that allows us to perform a single action in different ways.

Static polymorphism: Polymorphism that is resolved during compiler time is known as static polymorphism. Method overloading can be considered as static polymorphism example.

Dynamic Polymorphism: it is also known as Dynamic Method Dispatch. Dynamic polymorphism is a process in which a call to an overridden method is resolved at runtime rather, thats why it is called runtime polymorphism.

Child class object can be referred by parent interface reference variable

a) **Over Loading (Compile Binding):**

* When the method name is same with different arguments or input parameters in same class.
* We can Over load main method also
* We cannot create method in side method
* Avoid Duplicate methods
* Public clss prtp()

{

Public void sum(){

s.o.p();

}

Public void sum(int i)

{

s.o.p();

}

Public void sum(int k, int j)

{

S.o.p();

}

Prtp obj = new Prtp();

Sum.obj();

Sum.obj(int i);

Sum.obj(int k, int j);

}

**b) Overriding:**

When same method presents in parent class as well as in child class, with same name and same number of arguments.

***ii) Inheritance:***

* Child class inherits methods from parent class(super class) by using “Extends”.
* But parent class cannot inherits methods from child class .

***iii) Interface:***

* Only Method Declaration
* No method Body, only method proto type
* Here we can use variables, variables are static by default
* We cannot change variable values
* No static method
* No main method
* We cannot create an object of an interface, so interface in abstract in nature.
* When we are declaring interface in other class we use”implements”.

Public interface Prtp

{

Int bal = 100;

Public void credit();

Public void debit();

Public void transfermoney();

}

***iv) Abstarct Class***:

* The method which we declare it calls abstract method.
* A class which has at least one abstract method is abstract class.
* We cannot create object of abstract class.

***vi) Encapsulation:***

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**.

To achieve encapsulation in Java −

* Declare the variables of a class as private.
* Provide public setter and getter methods to modify and view the variables values.

**LOOP:**

Which will allow you to run certain piece of run multiple times?

i) For:

for (initialize; condition; increment/decrement)

{

Code/logic

}

**ii) While**

* It will execute if condition is true

While (condition)

{

//logic

// incrmnt or decrmnt

}

**iii) Do – While:**

* In this loop code/logic will execute at least once.

Do

{

}while (condition)

**Conditional Statements:**

* It gives power to take decisions when to execute and what to be done of condition true.

**i) IF:**

* It check the condition if condition is true then execute the code

If (condition)

{

}

**ii) If Else:**

* If condition is true then execute if block or execute else block

If()

{

}

Else

{

}

iii) **Nested If else:**

**iv) Switch Statement:**

* It allows you match with multiple cases

Switch(expression)

{

Case value:

//code here

Break;

Case value:

//code

Break;

Default:

}

**Points for switch:**

* Data type should be same for comparison
* We can have multiple cases
* We can use only integers string and Enums
* Break is optional but if not provided it will check all cases till the end
* Default is option

**Collection Framework**

* In the form of API which provides architecture to store different data and objects and manipulate data and group of object.
* We can do
  + Searching Data
  + Sorting
  + Insertion Data
  + Delete Data
  + Manipulate Data
* **Interfaces:**
  + Set
  + List
  + Queue
  + D queue
* **Classes**
  + Array List
  + Linked List
  + Vector
  + Tree Set
  + Hash set
  + Priority Queue

What is Collection in java: Collection represents a single unit of objects i.e. a group.

What is framework in java

* provides readymade architecture.
* represents set of classes and interface.
* is optional.

**What is Collection framework**

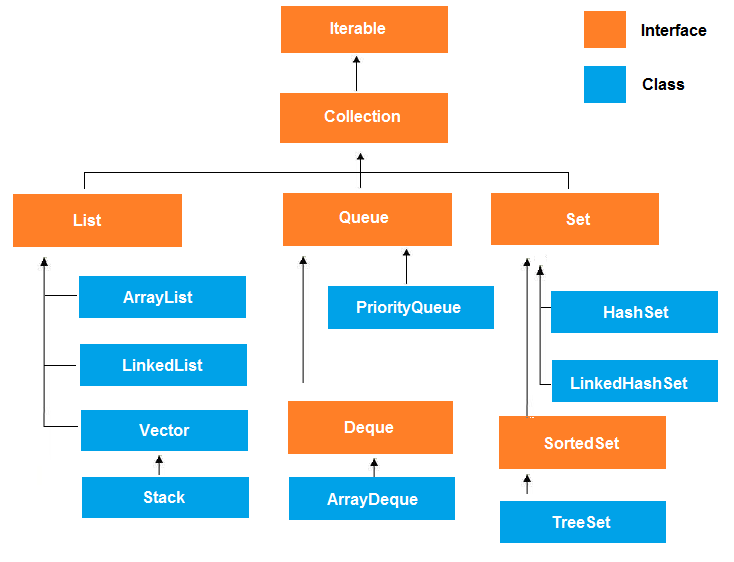
Collection framework represents a unified architecture for storing and manipulating group of objects.

It has:

+Interfaces and its implementations i.e. classes

+Algorithm

These all classes and interfaces is in java.utill package.



**Array List:**

**It is dynamic array**

Properties:

* Contains duplicate values/elements
* Maintain insertion order
* It synchronized
* Allows random access to fetch the elements bcz it stores the values in the basis of index

To print all values from arralist we use “for loop” and “iterator”

**This** Keyword is used to match the global and local variables.