What is Java?

Java is an [object-oriented](https://www.javatpoint.com/java-oops-concepts), class-based, concurrent, secured and general-purpose computer-programming language. It is a widely used robust technology.

Object-Oriented Programming is a paradigm that provides many concepts, such as **inheritance**, **data binding**, **polymorphism**, etc.

Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995.

Java Example

Let's have a quick look at Java programming example. A detailed description of Hello Java example is available in next page.

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

**Platform**: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

## Java Example

Let's have a quick look at Java programming example. A detailed description of Hello Java example is available in next page.

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

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=Simple)

## Application

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**

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are 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 a dynamic page is called a web application. Currently, [Servlet](https://www.javatpoint.com/servlet-tutorial), [JSP](https://www.javatpoint.com/jsp-tutorial), [Struts](https://www.javatpoint.com/struts-2-tutorial), [Spring](https://www.javatpoint.com/spring-tutorial), [Hibernate](https://www.javatpoint.com/hibernate-tutorial), [JSF](https://www.javatpoint.com/jsf-tutorial), 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. is called enterprise application. It has advantages of the high-level security, load balancing, and clustering. In Java, [EJB](https://www.javatpoint.com/ejb-tutorial) is used for creating enterprise applications.

#### **4) Mobile Application**

An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

# **Features of Java**

The primary objective of [Java programming](https://www.javatpoint.com/java-tutorial) language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as java buzzwords.

A list of most important features of Java language is given below.



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

### **Object-oriented**

Java is an [object-oriented](https://www.javatpoint.com/java-oops-concepts) programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.

Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

Basic concepts of OOPs are:

1. [Object](https://www.javatpoint.com/object-and-class-in-java)
2. Class
3. [Inheritance](https://www.javatpoint.com/inheritance-in-java)
4. [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
5. [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
6. [Encapsulation](https://www.javatpoint.com/encapsulation)

**Classloader:** Classloader in Java is a part of the Java Runtime Environment(JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.

**Bytecode Verifier:** It checks the code fragments for illegal code that can violate access right to objects.

## What happens at compile time?

At compile time, java file is compiled by Java Compiler (It does not interact with OS) and converts the java code into bytecode.



## What happens at runtime?

At runtime, following steps are performed:



|  |
| --- |
| **Classloader:**is the subsystem of JVM that is used to load class files. |
| **Bytecode Verifier:**checks the code fragments for illegal code that can violate access right to objects. |
| **Interpreter:**read bytecode stream then execute the instructions. |

**Date Types**

--------------

### **Understanding Java Access Modifiers**

Let's understand the access modifiers in Java by a simple table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

### **1) Private**

The private access modifier is accessible only within the class.

**Simple example of private access modifier**

In this example, we have created two classes A and Simple. A class contains private data member and private method. We are accessing these private members from outside the class, so there is a compile-time error.

1. **class** A{
2. **private** **int** data=40;
3. **private** **void** msg(){System.out.println("Hello java");}
4. }
6. **public** **class** Simple{
7. **public** **static** **void** main(String args[]){
8. A obj=**new** A();
9. System.out.println(obj.data);//Compile Time Error
10. obj.msg();//Compile Time Error
11. }
12. }

### **Role of Private Constructor**

If you make any class constructor private, you cannot create the instance of that class from outside the class. For example:

1. **class** A{
2. **private** A(){}//private constructor
3. **void** msg(){System.out.println("Hello java");}
4. }
5. **public** **class** Simple{
6. **public** **static** **void** main(String args[]){
7. A obj=**new** A();//Compile Time Error
8. }
9. }

#### **Note: A class cannot be private or protected except nested class.**

### **2) Default**

If you don't use any modifier, it is treated as **default** by default. The default modifier is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than private. But, it is more restrictive than protected, and public.

**Example of default access modifier**

In this example, we have created two packages pack and mypack. We are accessing the A class from outside its package, since A class is not public, so it cannot be accessed from outside the package.

1. //save by A.java
2. **package** pack;
3. **class** A{
4. **void** msg(){System.out.println("Hello");}
5. }
6. //save by B.java
7. **package** mypack;
8. **import** pack.\*;
9. **class** B{
10. **public** **static** **void** main(String args[]){
11. A obj = **new** A();//Compile Time Error
12. obj.msg();//Compile Time Error
13. }
14. }

In the above example, the scope of class A and its method msg() is default so it cannot be accessed from outside the 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.

It provides more accessibility than the default modifer.

**Example of protected access modifier**

In this example, we have created the two packages pack and mypack. The A class of pack package is public, so can be accessed from outside the package. But msg method of this package is declared as protected, so it can be accessed from outside the class only through inheritance.

1. //save by A.java
2. **package** pack;
3. **public** **class** A{
4. **protected** **void** msg(){System.out.println("Hello");}
5. }
6. //save by B.java
7. **package** mypack;
8. **import** pack.\*;
10. **class** B **extends** A{
11. **public** **static** **void** main(String args[]){
12. B obj = **new** B();
13. obj.msg();
14. }
15. }

Output:Hello

### **4) Public**

The **public access modifier** is accessible everywhere. It has the widest scope among all other modifiers.

**Example of public access modifier**

1. //save by A.java
3. **package** pack;
4. **public** **class** A{
5. **public** **void** msg(){System.out.println("Hello");}
6. }
7. //save by B.java
9. **package** mypack;
10. **import** pack.\*;
12. **class** B{
13. **public** **static** **void** main(String args[]){
14. A obj = **new** A();
15. obj.msg();
16. }
17. }

Output:Hello

### **Java Access Modifiers with Method Overriding**

If you are overriding any method, overridden method (i.e. declared in subclass) must not be more restrictive.

1. **class** A{
2. **protected** **void** msg(){System.out.println("Hello java");}
3. }
5. **public** **class** Simple **extends** A{
6. **void** msg(){System.out.println("Hello java");}//C.T.Error
7. **public** **static** **void** main(String args[]){
8. Simple obj=**new** Simple();
9. obj.msg();
10. }
11. }

The default modifier is more restrictive than protected. That is why, there is a compile-time error.

Next Topic[Encapsulation in Java](https://www.javatpoint.com/encapsulation)

[← prev](https://www.javatpoint.com/package)[next →](https://www.javatpoint.com/encapsulation)