**Introduction to Object-Oriented Programming**

Everywhere you look in the real world you see objects—people, animals, plants, cars, planes,buildings, computers and so on. Humans think in terms of objects. Telephones, houses, traffic lights, microwave ovens and water coolers are just a few more objects. Computer programs, such as the Java programs, are composed of lots of interacting software objects.

**Object-Oriented Programming (OOP)** is the term used to describe a programming approach based on **objects** and **classes**. The object-oriented paradigm allows us to organise software as a collection of objects that consist of both data and behaviour. This is in contrast to conventional functional programming practice that only loosely connects data and behaviour.

Since the 1980s the word 'object' has appeared in relation to programming languages, with almost all languages developed since 1990 having object-oriented features. Some languages have even had object-oriented features retro-fitted. It is widely accepted that object-oriented programming is the most important and powerful way of creating software.

The object-oriented programming approach encourages:

* Modularisation: where the application can be decomposed into modules.
* Software re-use: where an application can be composed from existing and new modules.

An object-oriented programming language generally supports five main features:

* Classes
* Objects
* Classification
* Polymorphism
* Inheritance

### Simple comparison between Object-oriented Programming and Procedural Programming

A simple way to compare both programming methods is to think of Object-oriented Programming as  learn to read picture book. As children see pictures of simple objects like a house or picture they know that throughout the book when they see a picture of the house it represents the word house. The children can then read through the book as words are substituted throughout the story with pictures.

In Object-oriented Programming the classes could represent the pictures in the learn to read books. A house could represent a class and anything the developer wants to have included to describe that house like the color, size, number of bathrooms etc.

In Procedural Programming the learn to read book would be words on the page without pictures to help guide the young learner through the book. If the story was changed in the beginning of the book it could disrupt or make the story later on in the book not make any sense. Although learning to read the book would make programming with Procedural Programming simple, it would make it difficult for other readers in the case of the book or programmers in the case of Procedural Programming to add to the story.

|  |  |
| --- | --- |
| **Procedural** | **Object-oriented** |
| procedure | Method |
| record | Object |
| module | Class |
| procedure call | Message |

### Areas of Application of OOP concept:

1. Real Time Systems Design
2. Simulation and Modeling System
3. Object Oriented Database
4. Object Oriented Distributed Database
5. Client-Server System

# History of Java

**Java history** is interesting to know. The history of java starts from Green Team. Java team members (also known as **Green Team**), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describes the history of java.

1) **James Gosling**, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.

2) Originally designed for small, embedded systems in electronic appliances like set-top boxes.

3) Firstly, it was called **"Greentalk"** by James Gosling and file extension was .gt.

4) After that, it was called **Oak** and was developed as a part of the Green project.

## Why "Oak" name

5) **Why Oak?** Oak is a symbol of strength and choosen as a national tree of many countries like U.S.A., France, Germany, Romania etc.

6) In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

# JVM (Java Virtual Machine)

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

### What is JVM

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

### What it does

The JVM performs following operation:

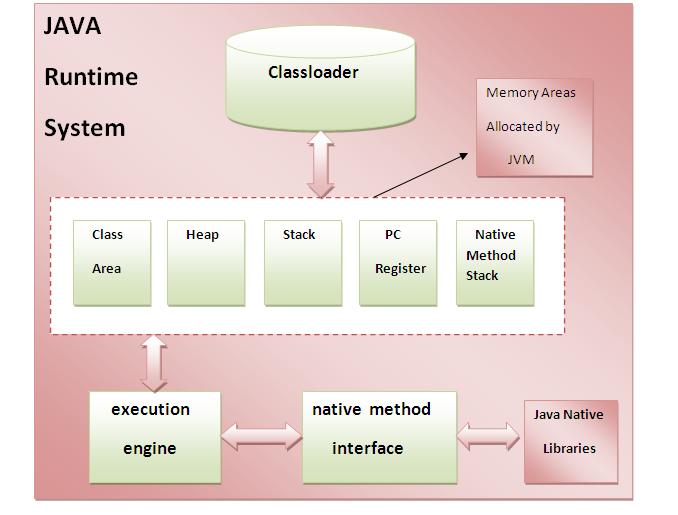
* Loads code
* Verifies code
* Executes code
* Provides runtime environment

JVM provides definitions for the:

* Memory area
* Class file format
* Register set
* Garbage-collected heap
* Fatal error reporting etc.

### Internal Architecture of JVM

|  |
| --- |
| Let's understand the internal architecture of JVM. It contains classloader, memory area, execution engine etc. |



### 1) Classloader

Classloader is a subsystem of JVM that is used to load class files.

### 2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

### 3) Heap

It is the runtime data area in which objects are allocated.

### 4) Stack

|  |
| --- |
| Java Stack stores frames.It holds local variables and partial results, and plays a part in method invocation and return. |
| Each thread has a private JVM stack, created at the same time as thread. |
| A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes. |

### 5) Program Counter Register

PC (program counter) register. It contains the address of the Java virtual machine instruction currently being executed.

### 6) Native Method Stack

It contains all the native methods used in the application.

### 7) Execution Engine

|  |
| --- |
| It contains: |
| **1) A virtual processor** |
| **2) Interpreter:** Read bytecode stream then execute the instructions. |
| **3) Just-In-Time(JIT) compiler:** It is used to improve the performance.JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.Here the term ?compiler? refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU. Variables and Data Types in Java Variable is a name of memory location. There are three types of variables in java: local, instance and static.  There are two types of data types in java: primitive and non-primitive. Variable **Variable** is name of reserved area allocated in memory. In other words, it is a name of memory location. It is a combination of "vary + able" that means its value can be changed. Types of Variable There are three types of variables in java:   * local variable * instance variable * static variable   types-of-variables.png 1) Local Variable A variable which is declared inside the method is called local variable. 2) Instance Variable A variable which is declared inside the class but outside the method, is called instance variable . It is not declared as static. 3) Static variable A variable that is declared as static is called static variable. It cannot be local. Java Keywords and Identifiers Keywords are predefined, reserved words used in Java programming that have special meanings to the compiler. For example:  int score;  Here, int is a keyword. It indicates that the variable *score* is of integer type (32-bit signed two's complement integer).  You cannot use keywords like int, for, class etc as variable name (or identifiers) as they are part of the Java programming language syntax. Here's the complete list of all keywords in Java programming.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Java Keywords List | | | | | | abstract | assert | boolean | break | byte | | case | catch | char | class | const | | continue | default | do | double | else | | enum | extends | final | finally | float | | for | goto | if | implements | import | | instanceof | int | interface | long | native | | new | package | private | protected | public | | return | short | static | strictfp | super | | switch | synchronized | this | throw | throws | | transient | try | void | volatile | while |   Beside these keywords, you cannot also use true, false and null as identifiers as they are [literals](https://www.programiz.com/java-programming/variables-primitive-data-types#literals). Java identifiers Identifiers are the name given to variables, classes, methods etc. Consider the above code;  int score;  Here, score is a variable (an identifier). You cannot use keywords as variable name. It's because keywords have predefined meaning. For example,  int float;  The above code is wrong. It's because float is a keyword and cannot be used as a variable name. Rules for Naming an Identifier  * Identifier cannot be a keyword. * Identifiers are case-sensitive. * It can have a sequence of letters and digits. However, it must begin with a letter, $ or \_. The first letter of an identifier cannot be a digit. * It's convention to start an identifier with a letter rather and $ or \_. * Whitespaces are not allowed. * Similarly, you cannot use symbols such as @, #, and so on.  Operators in java **Operator** in java is a symbol that is used to perform operations. For example: +, -, \*, / etc.  There are many types of operators in java which are given below:   * Unary Operator, * Arithmetic Operator, * shift Operator, * Relational Operator, * Bitwise Operator, * Logical Operator, * Ternary Operator and * Assignment Operator.  Java Operator Precedence  |  |  |  | | --- | --- | --- | | **Operator Type** | **Category** | **Precedence** | | Unary | postfix | expr++ expr-- | | prefix | ++expr --expr +expr -expr ~ ! | | Arithmetic | multiplicative | \* / % | | additive | + - | | Shift | Shift | << >> >>> | | Relational | comparison | < > <= >= instanceof | | equality | == != | | Bitwise | bitwise AND | & | | bitwise exclusive OR | ^ | | bitwise inclusive OR | | | | Logical | logical AND | && | | logical OR | || | | Ternary | ternary | ? : | | Assignment | assignment | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |  Java Expressions Expressions consist of [variables](https://www.programiz.com/java-programming/variables-primitive-data-types#variables), [operators](https://www.programiz.com/java-programming/operators), [literals](https://www.programiz.com/java-programming/variables-primitive-data-types#literals) and method calls that evaluates to a single value. To learn about method calls, visit *Java methods*.  Let's take an example,  int score;  score = 90;  Here, score = 90 is an expression that returns int. |

Java Data Type Casting Type Conversion

Java supports two types of castings – **primitive data type casting** and [**reference type casting**](http://way2java.com/casting-operations/object-casting/). Reference type casting is nothing but assigning one Java object to another object. It comes with very strict rules and is explained clearly in Object Casting.

1. **Implicit casting**
2. **Explicit casting**
3. **Boolean casting.**

**1. Implicit casting (widening conversion)**

**A data type of lower size (occupying less memory) is assigned to a data type of higher size**. This is done implicitly by the JVM. The lower size is widened to higher size. This is also named as **automatic type conversion**.

**Explicit casting (narrowing conversion)**

A data type of higher size (occupying more memory) cannot be assigned to a data type of lower size. This is not done implicitly by the JVM and requires **explicit casting**; a casting operation to be performed by the programmer. The higher size is narrowed to lower size.

Flow of Control

What is flow of control?

Both C++ and Java support several different kinds of statements designed to alter or control the logical flow of the program, although in some cases, the behavior of those statements differs between Java and C++.

The ability to alter the logical flow of the program is often referred to as Flow of Control.

Statements that support flow of control

The following table lists the statements supported by Java (and by C++ with some exceptions) for controlling the logical flow of the program.

Statement Type

if-else selection

switch-case selection

for loop

while loop

do-while loop

try-catch-finally exception handling

throw exception handling

break miscellaneous

continue miscellaneous

label: miscellaneous

return miscellaneous

goto reserved by Java but not supported