159.234
Object-Oriented Programming

Lecture 01 - Introduction to Java

Objective

Understand design goals and features of Java as a pure object-oriented language

Understand that Java technology is both a programming language and a platform.

Introduction to Java

Java was originally developed by Sun Microsystems in the early 90s and gained a lot of popularity for its portability, especially due to the ability to execute Java programs on the web as Java applets.

Java itself consists of the Java language specification and the Java API.

Introduction to Java

The Java language specification is a technical definition of the Java programming language's syntax and semantics.

The design of Java syntax was based heavily on C and C++ to provide programmers with a familiar programs.

However, Java provides fewer low-level features than C or C++.

Introduction to Java

Java is much closer to being a pure object-oriented language than C++. It is not possible to write a Java program without using classes.

Even the simplest example must use a class.

Introduction to Java

The major reason Java is not considered to be a pure object-oriented language is because it provides types such as int, float, char etc. that are not objects.

This was an intentional decision that was made when Java was created based on performance considerations.

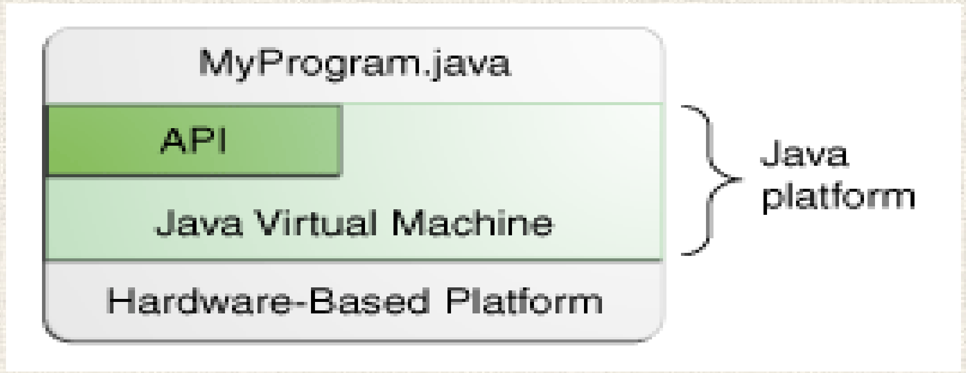
Java Compared to C++

C++	Java
Compatible with C	Not compatible with C
Compiled to native machine language	Compiled to bytecode
Allow calls to native system libraries	Calls to native functions go through JNI
Requires porting to other platforms	Write once , run anywhere
Exposes low-level system functions	Runs in a protected virtual machine (JVM)
Explicit memory management	Managed memory access
Multiple inheritance	Limited to single inheritance

The Java API

The Java application program interface (API) is a large collection of ready-made software components that provide many useful capabilities.

It is grouped into libraries of related classes and interfaces; these libraries are known as packages.



Java Platform API Specification

The Java Platform API Specification (https://docs.oracle.com/javase/8/docs/api/index.html) (https://docs.oracle.com/en/java/javase/11/docs/api/) contains the complete listing for all packages, interfaces, classes, fields, and methods supplied by the Java SE platform.

Load the page in your browser and bookmark it. As a programmer, it will become your single most important piece of reference documentation.

Design Goals of Java

Simple, Object-Oriented and Familiar

Java was designed that the fundamental concepts of Java are simple to understand, designed as an Object-Oriented language from the beginning and shares a lot of C/C++ syntax to make it is familiar to many programmers.

Design Goals of Java

Robust and Secure

Java performs a great deal more compile-time and run-time checking than C++. This involves both checking of the user-defined code as well as security features to try to prevent security exploits.

Design Goals of Java

Architecture Neutral and Portable

One of the major factors contributing to the success of Java is its portability. Java code is compiled into an architecture neutral format that can be transported and executed on multiple different hardware and software platforms.

Design Goals of Java

Interpreted, Threaded and Dynamic

Java is an interpreted language where a Java interpreter executes Java programs on the machine it is installed on. The Java platform supports multi-threaded within the language with synchronization primitives and thread-safe libraries.

Portability

Programs written in languages such as C or C++ are compiled directly into machine instructions specific to one architecture.

The source-code for these programs must be recompiled for each different architecture. This involves distributing the source-code of the program.

Portability

Java programs are instead compiled into an intermediate and machine-independent form called Java bytecode.

Java bytecode is the instruction set of the Java system and is not specific to any one machine.

Portability

Java bytecode programs are not executed directly on a machine and instead are run on a JVM or Java Virtual Machine.

A JVM is responsible for loading a Java program, interpreting the bytecode instructions and mapping them onto the instructions available on the machine (Just-In-Time or JIT Compilation).