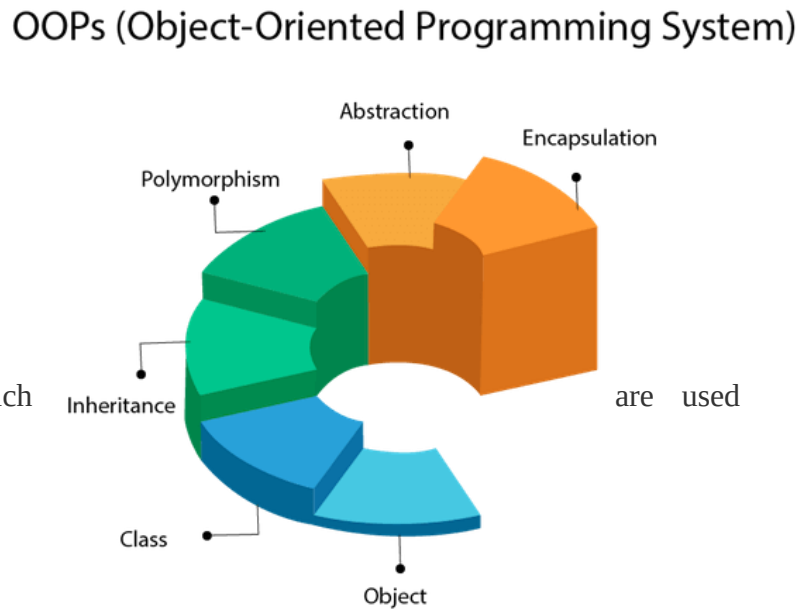
Introduction to Java: Object oriented concepts in Java, History and Features of Java, Java vs. C and C++, Use of Java in internet, Java development kit (JDK), Java Runtime Environment (JRE), Java Virtual Machine (JVM), API in Java, Running Java application and applets, Command line arguments.

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

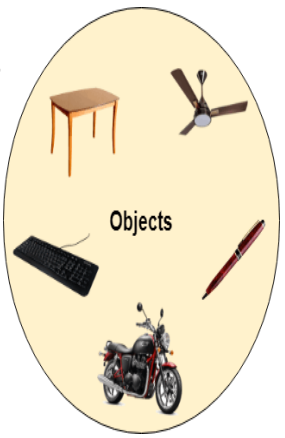
- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation



Apart from these concepts, there are some other terms which are used in Object-Oriented design:

- Coupling
- Cohesion
- Association
- Aggregation
- Composition

Object



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

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

Class

Collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism

If *one task is performed in different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

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

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

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 are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Coupling

Coupling refers to the knowledge or information or dependency of another class. It arises when classes are aware of each other. If a class has the details information of another class, there is strong coupling. In Java, we use private, protected, and public modifiers to display the visibility level of a class, method, and field. You can use interfaces for the weaker coupling because there is no concrete implementation.

Cohesion

Cohesion refers to the level of a component which performs a single well-defined task. A single well-defined task is done by a highly cohesive method. The weakly cohesive method will split the task into separate parts. The java.io package is a highly cohesive package because it has I/O related classes and interface. However, the java.util package is a weakly cohesive package because it has unrelated classes and interfaces.

Association

Association represents the relationship between the objects. Here, one object can be associated with one object or many objects. There can be four types of association between the objects:

- One to One

- One to Many
- Many to One, and
- Many to Many

Let's understand the relationship with real-time examples. For example, One country can have one prime minister (one to one), and a prime minister can have many ministers (one to many). Also, many MP's can have one prime minister (many to one), and many ministers can have many departments (many to many).

Association can be undirectional or bidirectional.

Aggregation

Aggregation is a way to achieve Association. Aggregation represents the relationship where one object contains other objects as a part of its state. It represents the weak relationship between objects. It is also termed as a *has-a* relationship in Java. Like, inheritance represents the *is-a* relationship. It is another way to reuse objects.

Composition

The composition is also a way to achieve Association. The composition represents the relationship where one object contains other objects as a part of its state. There is a strong relationship between the containing object and the dependent object. It is the state where containing objects do not have an independent existence. If you delete the parent object, all the child objects will be deleted automatically.

Advantage of OOPs over Procedure-oriented programming language

1) OOPs makes development and maintenance easier, whereas, in a procedure-oriented programming language, it is not easy to manage if code grows as project size increases.

2) OOPs provides data hiding, whereas, in a procedure-oriented programming language, global data can be accessed from anywhere.

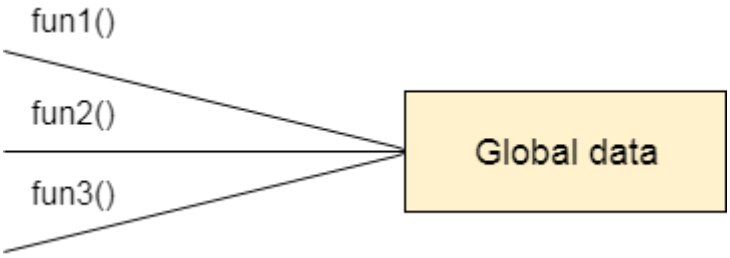


Figure: Data Representation in Procedure-Oriented Programming

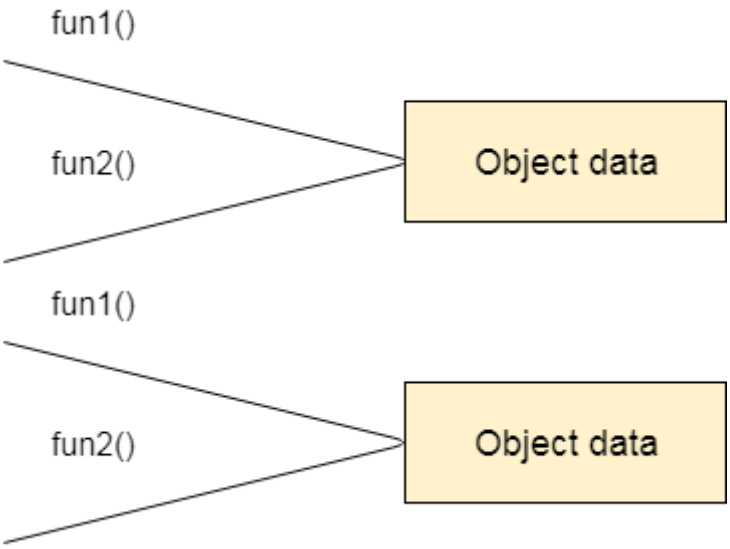
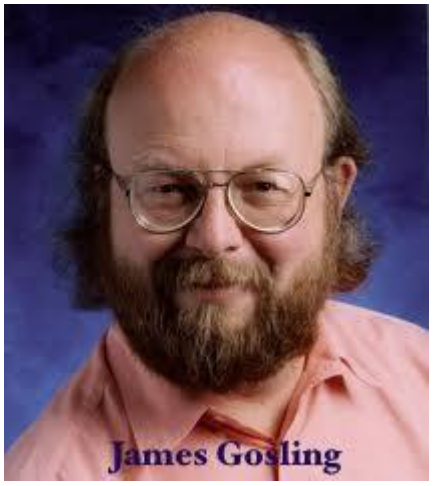


Figure: Data Representation in Object-Oriented Programming

3) OOPs provides the ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

History and Features of Java



The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc. However, it was best suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. Following are given significant points that describe the history of Java.

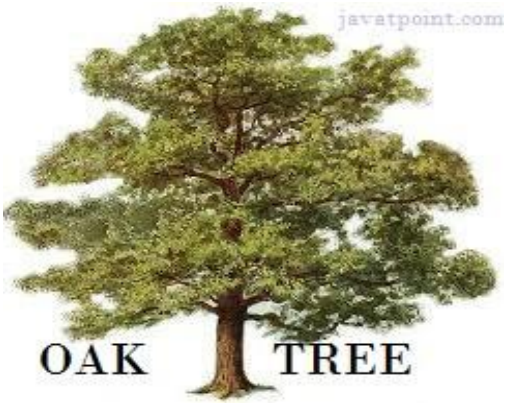
1) **James Gosling, Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.

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

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

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

Why Java was named as "Oak"?



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

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

Why Java Programming named "Java"?

7) Why had they chose the name Java for Java language? The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA", etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell, and fun to say.

According to James Gosling, "Java was one of the top choices along with **Silk**". Since Java was so unique, most of the team members preferred Java than other names.

8) Java is an island in Indonesia where the first coffee was produced (called Java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having a cup of coffee nearby his office.

9) Notice that Java is just a name, not an acronym.

10) Initially developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

11) In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.

12) JDK 1.0 was released on January 23, 1996. After the first release of Java, there have been many additional features added to the language. Now Java is being used in Windows applications, Web applications, enterprise applications, mobile applications, cards, etc. Each new version adds new features in Java.

Java vs. C and C++:

C vs C++ vs Java

The languages are based on each other but still, they are different in design and philosophy. The following table describes the major differences between C, C++, and Java. It will help you to select which language you have to learn.

S.N.	Basis	C	C++	Java
1	**Origin**	The C language is based on BCPL.	The C++ language is based on the C language.	The Java programming language is based on both C and C++.
2	**Programming Pattern**	It is a procedural language.	It is an object-oriented programming language.	It is a pure object-oriented programming language.
3	**Approach**	It uses the top-down approach.	It uses the bottom-up approach.	It also uses the bottom-up approach.

4	**Dynamic or Static**	It is a static programming language.	It is also a static programming language.	It is a dynamic programming language.
5	**Code Execution**	The code is executed directly.	The code is executed directly.	The code is executed by the JVM.
6	**Platform Dependency**	It is platform dependent.	It is platform dependent.	It is platform-independent because of byte code.
7	**Translator**	It uses a compiler only to translate the code into machine language.	It also uses a compiler only to translate the code into machine language.	Java uses both compiler and interpreter and it is also known as an interpreted language.
8	**File Generation**	It generates the .exe, and .bak, files.	It generates .exe file.	It generates .class file.
9	**Number of Keyword**	There are **32** keywords in the C language.	There are **60** keywords in the C++ language.	There are **52** keywords in the Java language.
10	**Source File Extension**	The source file has a .c extension.	The source file has a .cpp extension.	The source file has a .java extension.
11	**Pointer Concept**	It supports pointer.	It also supports pointer.	Java does not support the pointer concept because of security.
12	**Union and Structure Datatype**	It supports union and structure data types.	It also supports union and structure data types.	It does not support union and structure data types.
13	**Pre-processor Directives**	It uses pre-processor directives such as #include, #define, etc.	It uses pre-processor directives such as #include, #define, #header, etc.	It does not use directives but uses packages.
14	**Constructor/ Destructor**	It does not support constructor and destructor.	It supports both constructor and destructor.	It supports constructors only.
15	**Exception Handling**	It does not support exception handling.	It supports exception handling.	It also supports exception handling.
16	**Memory Management**	It uses the calloc(), malloc(), free(), and realloc() methods to manage the memory.	It uses new and delete operator to manage the memory.	It uses a garbage collector to manage the memory.
17	**Overloading**	It does not support the overloading concept.	Method and operator overloading can be achieved.	Only method overloading can be achieved.
18	**goto Statement**	It supports the goto statement.	It also supports the goto statement.	It does not support the goto statements.

19	**Used for**	It is widely used to develop drivers and operating systems.	It is widely used for system programming.	It is used to develop web applications, mobile applications, and windows applications.
20	**Array Size**	An array should be declared with size. For example, int num[10].	An array should be declared with size.	An array can be declared without declaring the size. For example, int num[].

Importance of Java to the Internet

Java has had a profound effect on the Internet because it allows objects to move freely in Cyberspace. In a network there are two categories of objects that are transmitted between the Server and the Personal computer.

Passive information

Dynamic active programs

The Dynamic Self-executing programs cause serious problems in the areas of Security and probability. But Java addresses those concerns and by doing so has opened the door to an exciting new form of program called the Applet.

Java can be used to create two types of programs

Applications: An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java's ability to create Applets makes it important.

Applet: An Applet is an application designed to be transmitted over the Internet and executed by a Java compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

Features of Java Security

Every time you that you download a program you are risking a viral infection. Prior to Java, most users did not download executable programs frequently and most users were worried about the possibility of infecting their systems with a virus. Java answers both these concerns by providing a "firewall" between a network application and your computer. When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection.

Portability

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed .As you will see, the same mechanism that helps ensure security also helps create portability. Indeed Java's solution to these two problems is both elegant and efficient.

Java Virtual Machine (JVM)

Beyond the language there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct

Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment

Compilation of code

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

Simple

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ programmer, learning Java will be even easier. Because Java inherits the C/C++ syntax and many of the objects oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner

Object-Oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend while simple types such as integers are kept as high-performance non-objects.

Robust

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time. Java virtually eliminates the problems of memory management and de-allocation, which is completely automatic. In a well-written Java program, all run time errors can -and should -be managed by your program.

SERVLETS

Introduction

The Java web server is Java Soft's own web Server. The Java web server is just a part of a larger framework, intended to provide you not just with a web server, but also with tools. To build customized network servers for any Internet or Intranet client/server system. Servlets are to a web server, how applets are to the browser.

Java development kit (JDK):

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop java applications and applets. It physically exists. It contains JRE + development tools.

JDK is an implementation of any one of the below given Java Platforms released by Oracle corporation:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) etc. to complete the development of a Java Application.

Components of JDK

Following is a list of primary components of JDK:

appletviewer:	This tool is used to run and debug Java applets without a web browser.
apt:	It is an annotation-processing tool.
extcheck:	it is a utility that detects JAR file conflicts.
idlj:	An IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.
jabswitch:	It is a Java Access Bridge. Exposes assistive technologies on Microsoft Windows systems.
java:	The loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.
javac:	It specifies the Java compiler, which converts source code into Java bytecode.
javadoc:	The documentation generator, which automatically generates documentation from source code comments
jar:	The specifies the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.
javafxpackager:	It is a tool to package and sign JavaFX applications.
jarsigner:	the jar signing and verification tool.
javah:	the C header and stub generator, used to write native methods.
javap:	the class file disassembler.
javaws:	the Java Web Start launcher for JNLP applications.
JConsole:	Java Monitoring and Management Console.
jdb:	the debugger.
jhat:	Java Heap Analysis Tool (experimental).
jinfo:	This utility gets configuration information from a running Java process or crash dump.
jmap:	Oracle jmap - Memory Map- This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump.
jmc:	Java Mission Control
jps:	Java Virtual Machine Process Status Tool lists the instrumented HotSpot Java Virtual Machines (JVMs) on the target system.
jrunscript:	Java command-line script shell.

jstack:	It is a utility that prints Java stack traces of Java threads (experimental).
jstat:	Java Virtual Machine statistics monitoring tool (experimental).
jstatd:	jstat daemon (experimental).
keytool:	It is a tool for manipulating the keystore.
pack200:	JAR compression tool.
Policytool:	It specifies the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources.
VisualVM:	It is a visual tool integrating several command-line JDK tools and lightweight [clarification needed] performance and memory profiling capabilities
wsimport:	It generates portable JAX-WS artifacts for invoking a web service.
xjc:	It is the part of the Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes.

What is the Java Runtime Environment?

The Java Runtime Environment (JRE) is software that Java programs require to run correctly. Java is a computer language that powers many current web and mobile applications. The JRE is the underlying technology that communicates between the Java program and the operating system. It acts as a translator and facilitator, providing all the resources so that once you write Java software, it runs on any operating system without further modifications.

Why is the JRE important?

A software program needs a runtime environment that provides access to memory and other system resources such as program files and dependencies. In the past, most software used the operating system directly as its runtime environment. However, this meant that developers had to write different code for each operating system that they wanted their applications to run on. The Java Runtime Environment (JRE) technology was created as a solution to this problem.

The JRE is actually one of three Java platform components that are required for any Java program to run successfully. The Java Development Kit (JDK) and Java Virtual Machine (JVM) are the other two components.

Java Development Kit

The JDK is a collection of software tools that you can use for developing Java applications. You can set up the JDK in your development environment by downloading and installing it. Select the JDK software version that matches the Java version you want to use. For example, Java Standard Edition, or Java SE, requires the Java SE JDK.

Java Virtual Machine

The JVM is software that runs the Java program line by line. Developers configure the JVM settings to manage program resources when the Java application runs. For example, you can change the JVM memory setting and check how much internal memory your Java applications use at runtime.

Role of the JRE in Java programming language

The JRE combines the Java code that you create by using the JDK with additional built-in code called libraries. It then creates a JVM instance, or local copy, that finally runs the Java programs. JVMs are available for multiple operating systems, and the JRE generates a single copy of your Java code that runs on all types of JVMs. In this way, the JRE facilitates platform independence for Java applications. You can write them once and run them anywhere.

Difference between the JRE, JVM, and JDK

The JDK is a software layer above the JRE that contains a compiler, a debugger, and other tools commonly found in any software development environment. You write code in English-like syntax in the JDK. The JDK compiles it and passes the byte code to the JRE. In contrast, the JRE contains class libraries, supporting files, and the JVM. It uses these software components to run the byte code on any device.

How does the JRE work?

The Java Runtime Environment (JRE) runs on top of the operating system, providing additional Java-specific resources. The Java Development Kit (JDK) and JRE interact to create a sustainable runtime environment that runs Java program files on any machine. The JRE uses three core components to work.

ClassLoader

Java class libraries contain collections of pre-written code that you can call as needed. They simplify the job of Java developers by providing built-in methods for common and non-trivial tasks such as taking input from users, displaying output to users, and more. All Java programs reference several class libraries. The Java ClassLoader dynamically loads all class files necessary into the Java Virtual Machine (JVM) on demand.

Bytecode verifier

The JDK has a compiler that converts the English-like code you write into a machine language version called Java bytecode. They bytecode verifier in the JRE checks the format and accuracy of the Java code before loading it into the JVM. For example, if the code violates system integrity or access rights, the JRE will not load the class file.

Interpreter

After the bytecode successfully loads, the Java interpreter creates the JVM instance that runs the Java program on the underlying machine.

What are the components of the JRE?

Other than the core components, the Java Runtime Environment (JRE) contains several other software components that help run Java programs more efficiently. The following are some examples:

Development tools

The JRE contains development tools such as user interface toolkits that you can use to improve the quality of your applications. The following are some examples:

Java 2D

Java 2D is an application programming interface (API) that you can use to draw two-dimensional graphics and create rich user interfaces, games, animations, and special effects in the Java language.

Swing

Swing is a lightweight graphical user interface (GUI) that offers flexible, user-friendly customizations.

Abstract Window Toolkit

Abstract Window Toolkit (AWT) is a GUI that you can use to create UI objects such as buttons, windows, and scroll bars.

Deployment solutions

The JRE includes technologies that simplify the process of releasing software changes to application users. These technologies also provide advanced support for updates in the application. Deployment technologies such as Java Web Start and Java plugin are included as part of the JRE installation. These technologies simplify the activation of applications and also provide advanced support for future updates to the JRE. The following are some examples:

Java Web Start

With Java Web Start, you can launch full-featured applications with a single click from your web browser.

Java Plugin

Java plugin establishes a connection between popular browsers and the Java platform so that you can run your website applets within a desktop browser.

Language and utility libraries

A collection of Java class files is called a Java package. The JRE includes several Java packages that support versioning, management, and monitoring. The following packages are some examples:

Collections framework

The collections framework is a unified architecture that includes interfaces for improving the storage and processing of application data.

Preferences API

The Preferences API allows multiple users on the same machine to define their own group of application preferences.

Logging

Logging packages produce log reports for troubleshooting incidents such as security failures, performance issues, and configuration errors.

Java Archive

Java Archive (JAR) is a platform-independent file format that lets you bundle multiple files to reduce your application file sizes and significantly improve download speed.

Integration libraries

The JRE includes several integration libraries that assist developers in creating seamless data connections between their services and applications. The following are some example libraries:

Java IDL

Java Interface Definition Language (IDL), which is based on the Common Object Request Broker Architecture (CORBA), supports distributed data objects—that is, objects that interact on different platforms across a network. For example, Java IDL allows objects written in Java to interact with those written in another language, such as C, C++, or COBOL.

Java Database Connectivity

Developers use the Java Database Connectivity (JDBC) API to write applications that can access remote databases, spreadsheets, and files.

Java Naming and Directory Interface

Java Naming and Directory Interface (JNDI) is a directory service that lets clients create portable applications that fetch information from external databases using naming rules.

What is AWS SDK for Java?

The AWS Software Development Kit (SDK) for Java simplifies the use of AWS services by providing a set of libraries that are consistent and familiar for Java developers. It supports higher-level abstractions for simplified development. AWS focused open-source Java libraries are available along with code examples and a Java API reference guide.

Java Virtual Machine (JVM):

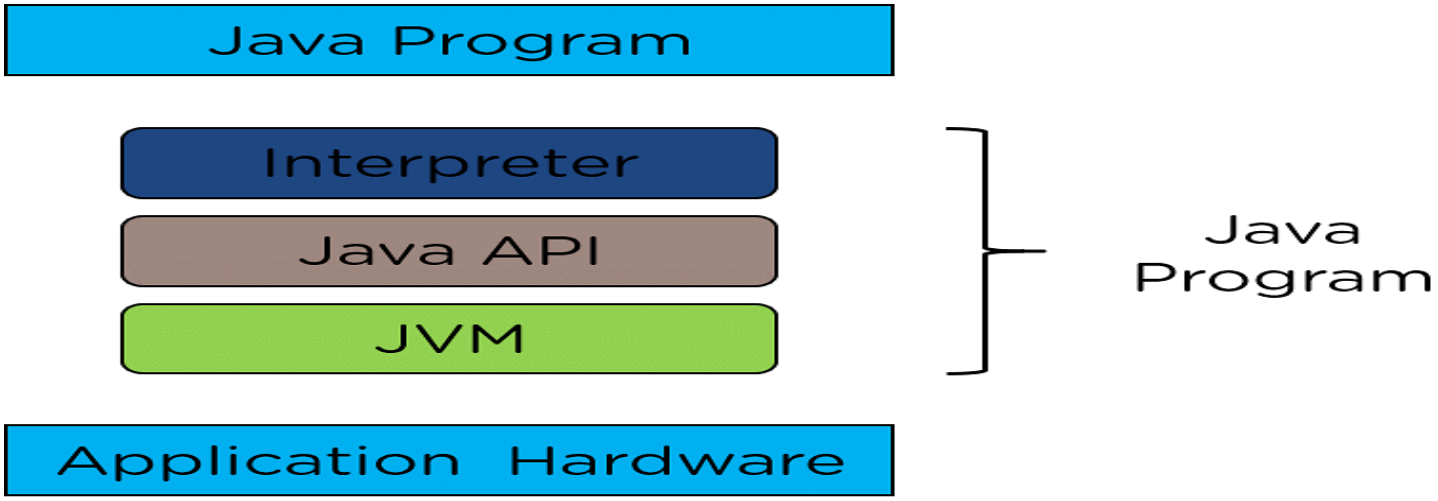
A Java virtual machine (JVM), an implementation of the Java Virtual Machine Specification, interprets compiled Java binary code (called bytecode) for a computer's processor (or "hardware platform") so that it can perform a Java program's instructions. Java was designed to allow application programs to be built that could be run on any platform without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible because it is aware of the specific instruction lengths and other particularities of the platform.

The Java Virtual Machine Specification defines an abstract -- rather than a real -- machine or processor. The Specification specifies an instruction set, a set of registers, a stack, a "garbage heap" and a method area. Once a Java virtual machine has been implemented for a given platform, any Java program (which, after compilation, is called bytecode) can run on that platform. A Java virtual machine can either interpret the bytecode one instruction at a time (mapping it to a real processor instruction) or the bytecode can be compiled further for the real processor using what is called a just-in-time compiler.

What Are Java APIs?

APIs are important software components bundled with the JDK. APIs in Java include classes, interfaces, and user Interfaces. They enable developers to integrate various applications and websites and offer real-time information.

The following image depicts the fundamental components of the Java API.



Now that we know the basics of the Java API and its components, let's explore who uses them.

Who Uses Java APIs?

Three types of developers use Java APIs based on their job or project:

1. Internal developers
2. Partner developers
3. Open developers

Internal Developers

Internal developers use internal APIs for a specific organization. Internal APIs are accessible only by developers within one organization.

Applications that use internal APIs include:

- B2B
- B2C
- A2A
- B2E

Examples include Gmail, Google Cloud VM, and Instagram.

Partner Developers

Organizations that establish communications develop and use partner APIs. These types of APIs are available to partner developers via API keys.

Applications that use partner APIs include:

- B2B
- B2C

Examples include Finextra and Microsoft (MS Open API Initiative),

Open Developers

Some leading companies provide access to their APIs to developers in the open-source format. These businesses provide access to APIs via a key so that the company can ensure that the API is not used illegally.

The application type that uses internal APIs is:

- B2C

Examples include Twitter and Telnyx.

The next section explores the importance of Java APIs.

The Need for Java APIs

Java developers use APIs to:

Streamline Operating Procedures

Social media applications like Twitter, Facebook, LinkedIn, and Instagram provide users with multiple options on one screen. Java APIs make this functionality possible.

Improve Business Techniques

Introducing APIs to the public leads many companies to release private data to generate new ideas, fix existing bugs, and receive new ways to improve operations. The Twitter developer account is an example of an API that gives programmers private API keys to access Twitter data and develop applications.

Create Powerful Applications

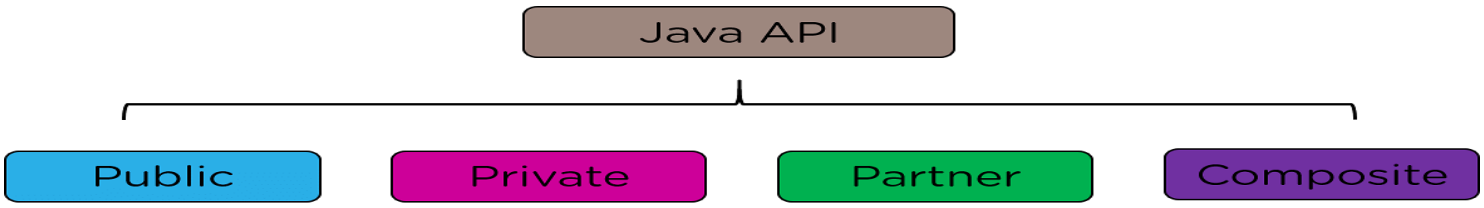
Online banking has changed the industry forever, and APIs offer customers the ability to manage their finances digitally with complete simplicity.

Below we discuss the various types of Java APIs.

Types of Java APIs

There are four types of APIs in Java:

- Public
- Private
- Partner
- Composite



Public

Public (or open) APIs are Java APIs that come with the JDK. They do not have strict restrictions about how developers use them.

Private

Private (or internal) APIs are developed by a specific organization and are accessible to only employees who work for that organization.

Partner

Partner APIs are considered to be third-party APIs and are developed by organizations for strategic business operations.

Composite

Composite APIs are microservices, and developers build them by combining several service APIs.

Now that we've covered the types of Java APIs, let's discuss the categorization of Java APIs based on the services that different varieties of APIs in Java provide.

Data and API Services

Data and API services are another way to categorize Java APIs other than public, private, partner, and composite. APIs are also classified based on their data-manipulation capabilities and the variety of services they offer, including:

- Internal API services
- External API services
- CRUD
- User interface services

Internal API Services

Internal API services are developed to offer organizations services specific to that organization. These services include only complex data operations and internal processes.

External API Services

External APIs are open-source APIs that developers integrate into an existing application or website.

CRUD

CRUD APIs provide data manipulation operations over various data storage units such as software as a service (SaaS) and relational database management systems (RDBMS), using standard storage-unit connecting tools like Java Database Connectivity (JDBC).

User Interface Services

User interface service APIs are open-source APIs that allow developers to build user interfaces for mobile devices, computers, and other electronics.

Next, let’s examine the rules and protocols that Java APIs follow.

API Service Protocols

The rules and protocols guide the functionality of the Java API. Different APIs have different service protocols. Let’s consider an example of RESTful API service protocol as an example.

For a typical RESTful API, developers must follow these rules:

- Stateless
- Uniform interface
- Client-server
- Cache
- Layered

Stateless

A RESTful API follows client-server architecture so it must be stateless.

Uniform Interface

The entities in a RESTful API are the server and clients. Applications that run on a global scale need a uniform client and server interface through the Hypertext Transfer Protocol (HTTP). Uniform Resource Identifiers (URIs) allocate the required resources.

Client-Server

The client-server model used in the RESTful API should be fault-tolerant. Both the client and server are expected to operate independently. The changes made at the client end should not affect the server end and vice versa.

Cache

Including a cache memory allows the application to record intermediate responses and run faster in real-time. A RESTful API also includes the cache memory.

Layered

A RESTful API is built using layers. Layers in the API are loosely coupled, or independent, from each other. Each layer contributes to a different level of hierarchy and also supports encapsulation.

Next we'll go over the most frequently used Java APIs.

Java application and applets:

Java Application

Java application is basically a Java program (collection of instructions) that runs stand alone in a client or server, and works on an underlying OS (operating system) that receives virtual machine support. Graphical User Interface (GUI) is not required for execution of a Java application.

Java Application Features

- It resembles Java programs strongly.
- We do not require any web browser for the execution of these programs. Execution can be easily done via the local system.
- The implementation of these applications requires main( ) function.
- The local file systems and networks are fully accessed by these applications.

Java Applet

The Java applet works on the client side, and runs on the web browser. It is a Java application that the user can easily embed on a web page.

Java Applet Features

- Java applet is a small and easy-to-write Java program.
- One can easily install Java applet along with various HTML documents.
- One needs a web browser (Java based) to use applets.
- Applet do not have access to the network or local disk and can only access browser-specific services.

- It cannot perform system operations on local machines.
- Java applet cannot establish access to local system.

Java Application Vs. Java Applet

Parameters	Java Application	Java Applet
Meaning	A Java Application also known as application program is a type of program that independently executes on the computer.	The Java applet works on the client side, and runs on the browser and makes use of another application program so that we can execute it.
Requirement of main( ) method	Its execution starts with the main( ) method only. The use of the main( ) is mandatory.	It does not require the use of any main() method. Java applet initializes through init( ) method.
Execution	It cannot run independently, but requires JRE to run.	It cannot start independently but requires APIs for use (Example. APIs like Web API).
Installation	We need to install the Java application first and obviously on the local computer.	Java applet does not need to be pre-installed.
Connectivity with server	It is possible to establish connections with other servers.	It cannot establish connection to other servers.
Operation	It performs read and write tasks on a variety of files located on a local computer.	It cannot run the applications on any local computer.
File access	It can easily access a file or data available on a computer system or device.	It cannot access the file or data found on any local system or computer.
Security	Java applications are pretty trusted, and thus, come with no security concerns.	Java applets are less reliable. So, they need to be safe.

Java Command Line Arguments

1. Command Line Argument
2. Simple example of command-line argument
3. Example of command-line argument that prints all the values

The java command-line argument is an argument i.e. passed at the time of running the java program.

The arguments passed from the console can be received in the java program and it can be used as an input.

So, it provides a convenient way to check the behavior of the program for the different values. You can pass **N** (1,2,3 and so on) numbers of arguments from the command prompt.

Simple example of command-line argument in java

In this example, we are receiving only one argument and printing it. To run this java program, you must pass at least one argument from the command prompt.

```
1. class CommandLineExample{
```

```
2.  public static void main(String args[]){
3.  System.out.println("Your first argument is: "+args[0]);
4.  }
5.  }
```

1. compile by > javac CommandLineExample.java
2. run by > java CommandLineExample sonoo

```
Output: Your first argument is: sonoo
```