## **What is Java?**

Java is a popular programming language, created in 1995.

It is owned by Oracle, and more than **3 billion** devices run Java.

It is used for:

* Mobile applications (especially Android apps)
* Desktop applications
* Web applications
* Web servers and application servers
* Games
* Database connection
* And much, much more!

**OOPS Concepts**

## **(Object-Oriented Programming System)**

**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.

## 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.

A class is a group of objects that have mutual methods. It can be considered as the blueprint using which objects are created.

Classes being passive do not communicate with each other but are used to instantiate objects that interact with each other.

Example:

## Object

Any entity that has state and behaviour 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.

An object stores its information in attributes and discloses its behaviour through methods.

Example:

Pen is an object. Its name is Reynolds; colour is white, known as its state. It is used to write, so writing is its behaviour.

### Inheritance

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

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

* **subclass** (child) - the class that inherits from another class
* **superclass** (parent) - the class being inherited from

To inherit from a class, use the extends keyword.

In the example below, the Car class (subclass) inherits the attributes and methods from the Vehicle class (superclass):

Example:



Programmer is the subclass and Employee is the superclass. The relationship between the two classes is **Programmer IS-A Employee**. It means that Programmer is a type of Employee.

### 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.

Polymorphism is the process of using same method name by multiple classes and redefines methods for the derived classes.

Example:

Bank is a class that provides a method to get the rate of interest. However, the rate of interest may differ according to banks. For example, SBI, ICICI, and AXIS banks are providing 8.4%, 7.3%, and 9.7% rate of interest.

#### Abstraction

Hiding internal details and showing functionality is known as abstraction. For example, phone call, we don't know the internal processing.

Example:

Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

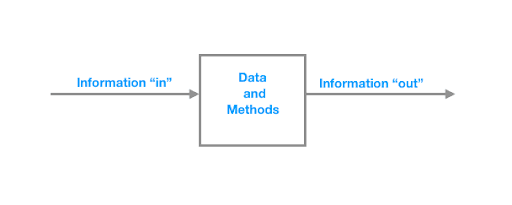
### Encapsulation

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.

Hiding the implementation details of the class from the user through an object’s methods is known as data encapsulation. In object-oriented programming, it binds the code and the data together and keeps them safe from outside interference.

Example:



Abstraction is the act of reducing programming complexity by representing essential features without including the background explanations or details. Classes are the concept of abstraction and are defined as the list of abstract attributes such as size, weight, cost, and methods that operate on these attributes. Classes wrap or encapsulate all the essential properties of the objects that are to be created.

## **Interface**

The point where the software entities interact with each other either in a single computer or in a network is known as a public interface. This help in data security. Other objects can change the state of an object in an interaction by using only those methods that are exposed to the outer world through a public interface.

**Difference Between JRE and JVM**

|  |  |  |
| --- | --- | --- |
| JRE | JVM | |
| 1. JRE is the Java Runtime Environment. | JVM is Java Virtual Machine. | |
| 1. Java Runtime Environment is a part of Java Development Kit (JDK) that comprises of a JVM, core classes and support libraries. Its main function is to provide a runtime environment to run java code. | Java Virtual Machine (JVM) is like a virtual computer that executes a set of per-compiled statements. When a Java compiler compiles source code, it generates a set of instructions called byte-code (.class files). The Java Virtual Machine interprets byte-code instructions and converts them into machine code for its execution. | |
| 1. JRE acts as a container that contains JVM, libraries and other files. Moreover, JRE doesn’t contain development tools like compiler and debugger. | | JVM is a part of JRE. It can be considered as a subset of JRE. |
| 1. It is not used for development purposes. | It is used for development purposes. | |
| 1. It is used for running java code. | It is not used for running java code. | |
| 1. JRE is composed of JVM + Libraries to run applications. | It contains only a runtime environment for executing Java byte code. | |
| 1. It contains class libraries that are required to run programs. | It does not contain any tools. | |

## **System Software**

[Software](https://www.toppr.com/guides/computer-aptitude-and-knowledge/basics-of-computers/hardware-and-software/" \t "_blank) is a set of programs that are used to perform a specific task. Similarly, **system software** is a set of programs that handles all the basic internal working of a computer.

**The most important tasks performed by the operating system are:**

1. **[Memory Management](https://www.geeksforgeeks.org/operating-system-overlays-memory-management/)**: The OS keeps track of the primary memory and allocates the memory when a process requests it.

2. **[Processor Management](https://www.geeksforgeeks.org/gate-notes-operating-system-process-management-introduction/)**: Allocates the main memory (RAM) to a process and de-allocates it when it is no longer required.

3. **[File Management:](https://www.geeksforgeeks.org/file-system-operating-systems/)** Allocates and de-allocates the resources and decides who gets the resources.

4. **Security:** Prevents unauthorized access to programs and data by means of passwords.

5. **Error-detecting Aids:** Production of dumps, traces, error messages, and other debugging and error-detecting methods.

6. **Scheduling:** The OS schedules process through its scheduling algorithms.

**Steps of Software System**

**Compiler:**A compiler is a software that translates the code written in one language to some other language without changing the meaning of the program. The compiler is also said to make the target code efficient and optimized in terms of time and space.

A compiler performs almost all of the following operations during

compilation: per-processing, lexical analysis, parsing, semantic analysis (syntax-directed translation), conversion of input programs to an intermediate representation, code optimization and code generation. Examples of compiler may include GCC (C compiler), g++ (C++ Compiler), javac (Java Compiler) etc.

**Interpreter:** An interpreter is a computer program that directly executes, i.e., it performs instructions written in a programming or scripting language. Interpreter do not require the program to be previously compiled into a machine language program. An interpreter translates high-level instructions into an intermediate form, which is then executes.

Interpreters are fast as it does not need to go through the compilation stage during which machine instructions are generated. Interpreter continuously translates the program until the first error is met. If an error comes it stops executing

**Assembler:**An assembler is a program that converts assembly language into machine code. It takes the basic commands and operations and converts them into binary code specific to a type of processor.

Assemblers produce executable code that similar to compilers. However, assemblers are more simplistic since they only convert low-level code (assembly language) to machine code. Since each assembly language is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code. On the other hand, compilers must convert generic high-level source code into machine code for a specific processor.