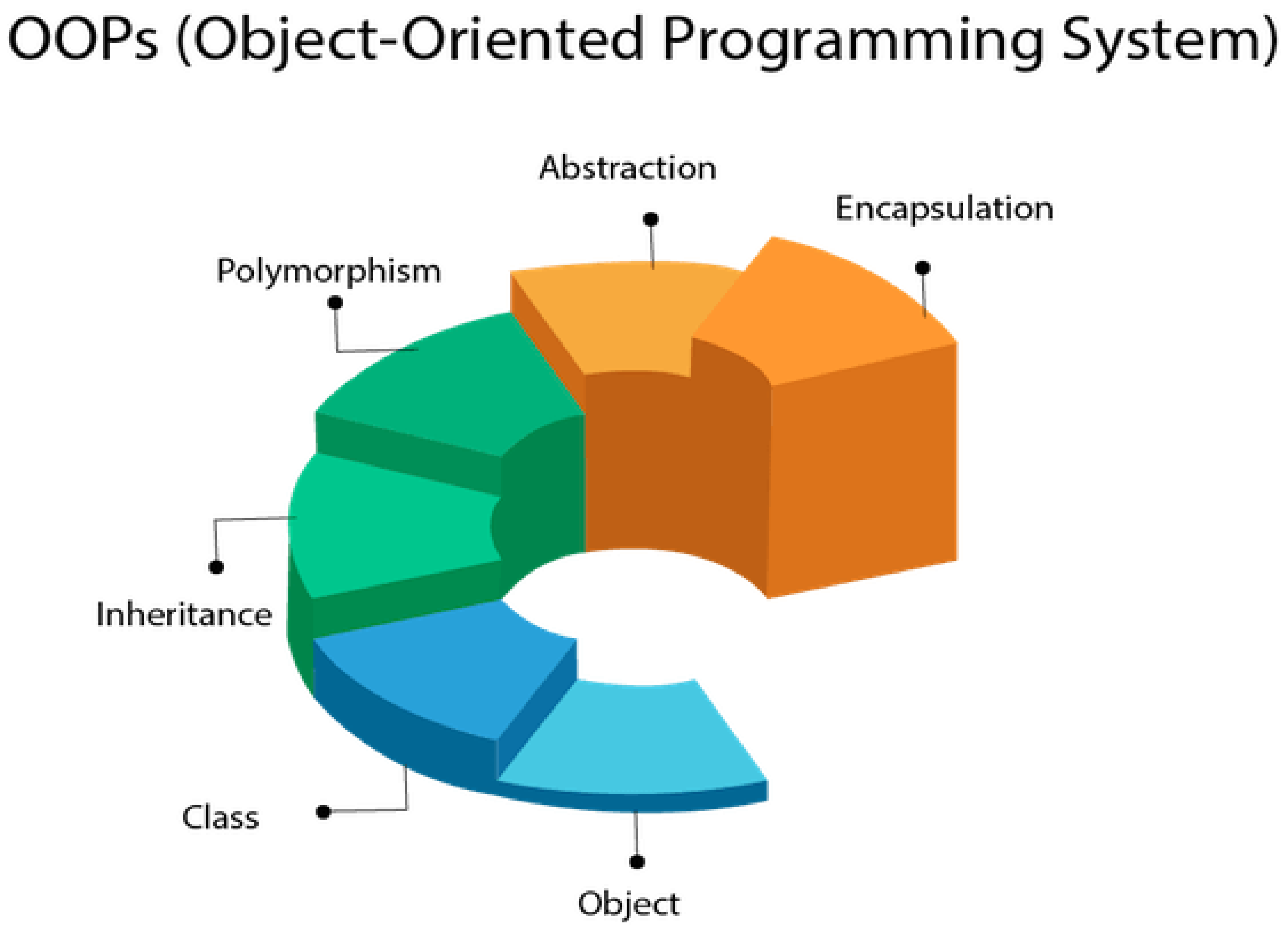
Object Oriented Programming

Unit 1

Object Oriented Concepts

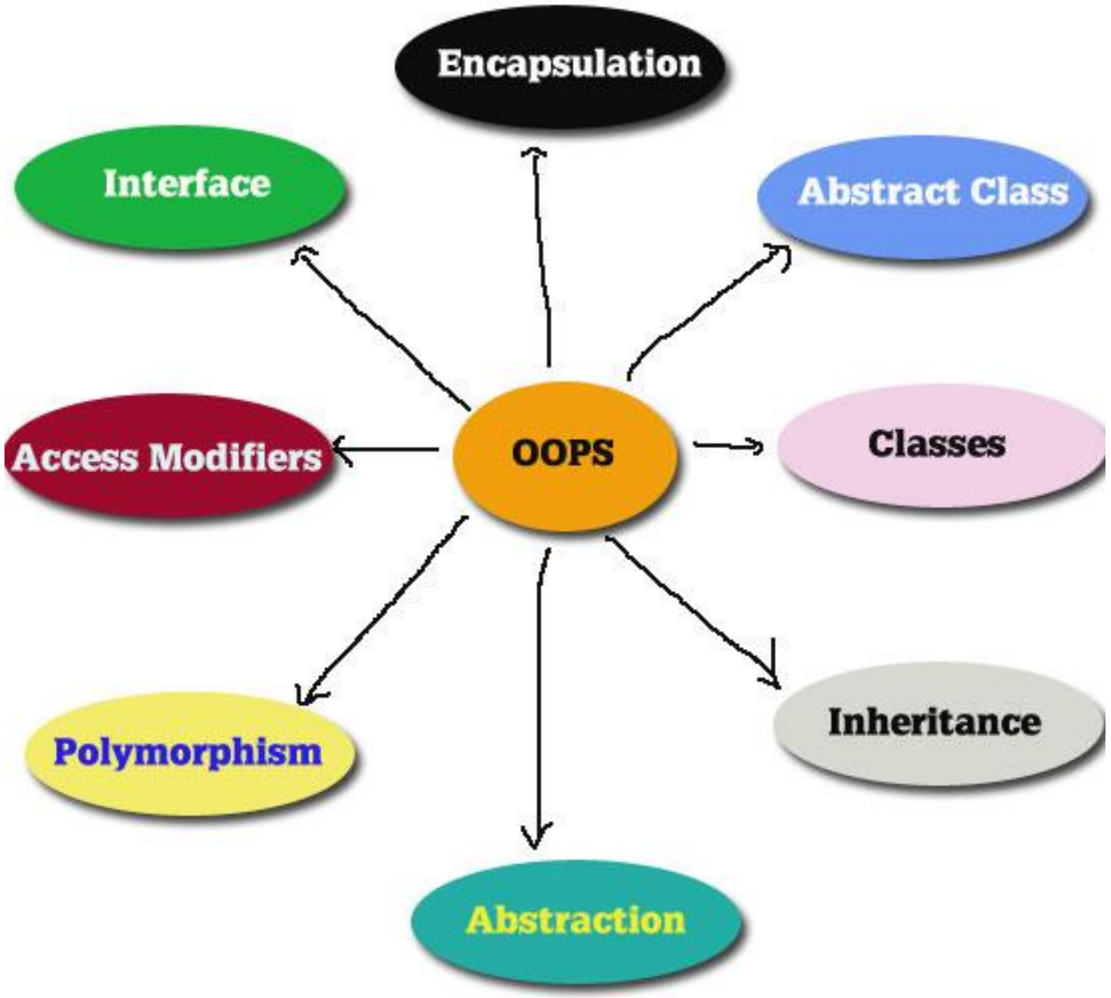
Introduction to OOP Concepts: Objects, Class, Elements of OOP : Inheritance , Encapsulation , Association, Abstraction, Polymorphism

Java OOPs Concepts



Java OOPs Concepts

1. **Object**
2. **Classes**
3. **Data Encapsulation**
4. **Data Abstraction**
5. **Polymorphism**
6. **Inheritance**



Basic Concept of OOPs

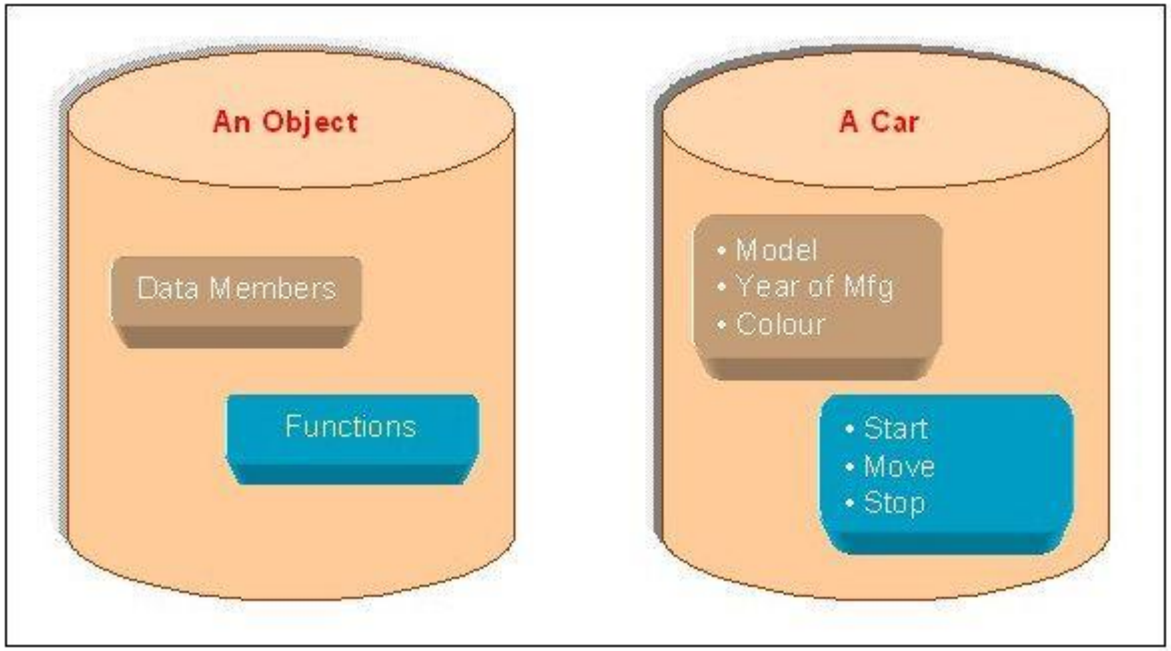
❑ Objects:

➢ Objects are the **basic run-time entities** of an object oriented system.

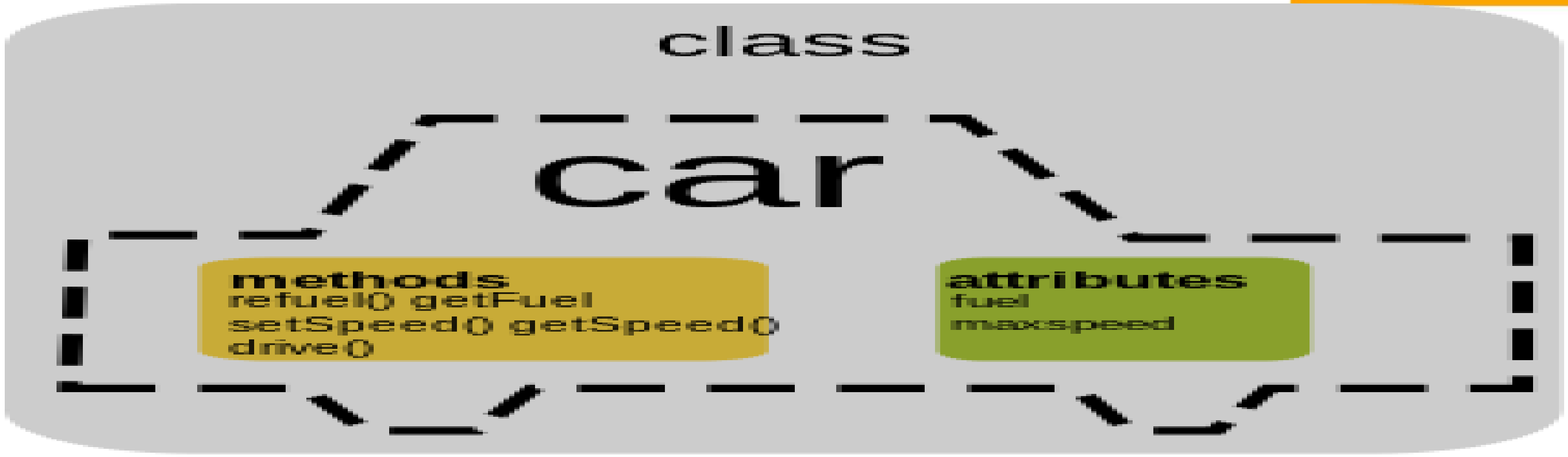
➢ They may represent a person, a place or any item that the program must handle.

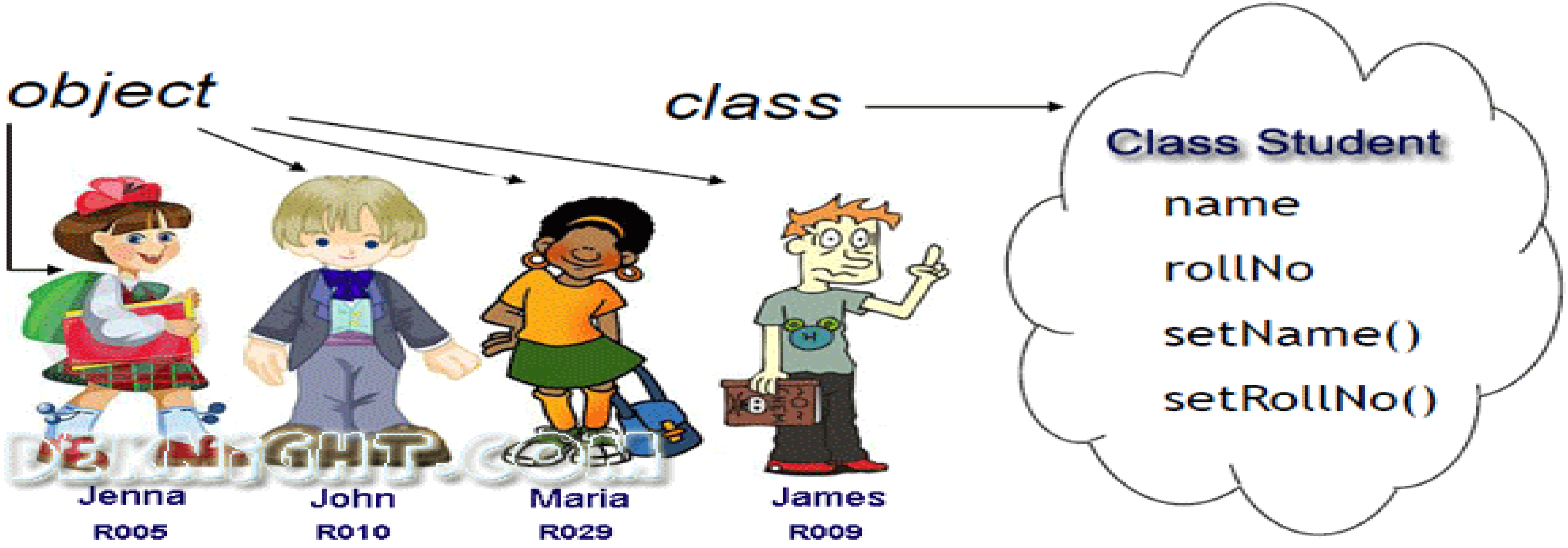
➢ **Syntax: Classname objname = new Classname();**

➢ Example: Representation of an object.



Examples of Objects



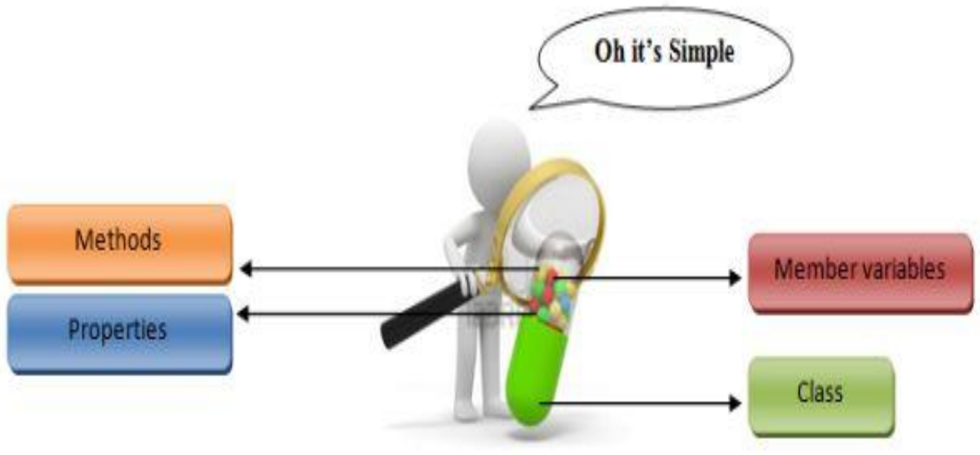


Basic Concept of OOPs

❑ **Classes:**

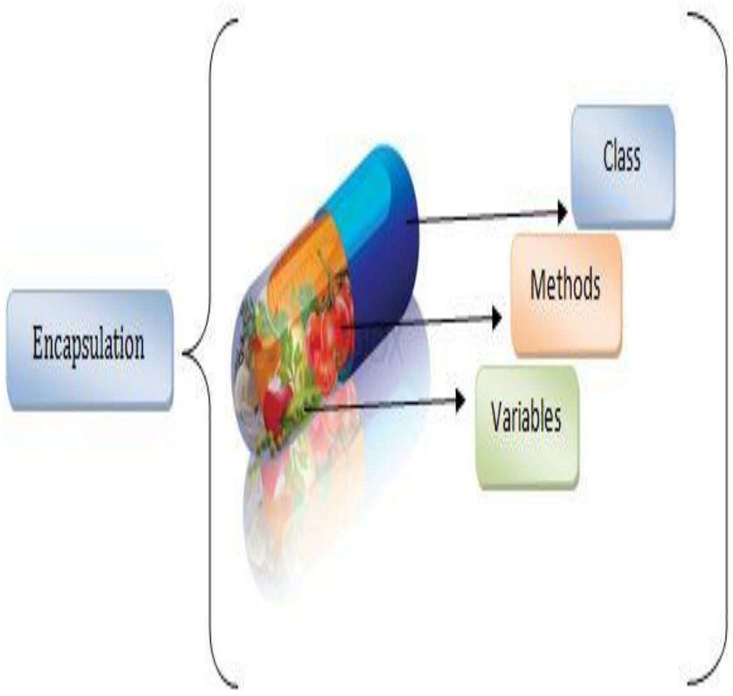
- ➢ Classes are **user-defined data types** and it behaves like built-in types of programming language.
- ➢ Object contains code and data which can be made user defined type using class.
- ➢ **Objects are variables of class.**
- ➢ Once a class has been defined, we can create any number of objects for that class.
- ➢ A **class** is a collection of **objects of similar type**.
- ➢ **Syntax:**

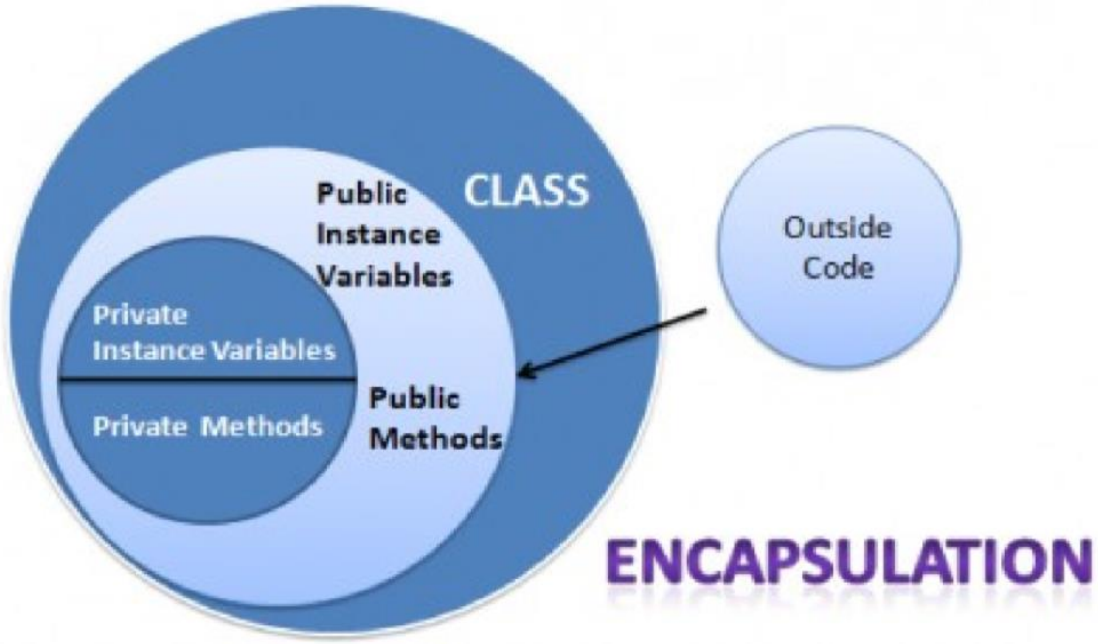
```
public class classname
{
    int x;
}
```



Basic Concept of OOPs: Encapsulation

- **Encapsulation** is the first pillar or principle of object-oriented programming.
- Encapsulation is a **process of binding data members** (variables, properties) and **member functions** (methods) into a single unit.
- Class is the best example of encapsulation.





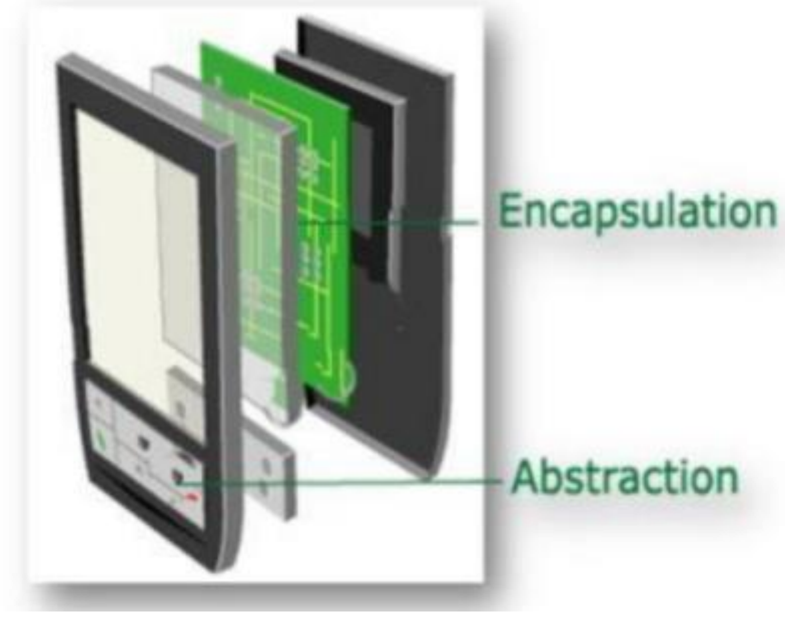
Basic Concept of OOPs

Data Abstraction

Use abstraction
Without Abstraction

Basic Concept of OOPs: Data Abstraction

- **Abstraction refers the representation of necessary features without including more details or explanations.**
- For example, when you press a key on your keyboard the character appears on the screen, you need to know only this, but how exactly it works electronically is not needed. This is called Abstraction.
- Abstraction lets you focus on **what the object does instead of how it does it.**



Basic Concept of OOPs

Polymorphism

"Now Speak!"
"Quack"
"Woof"
"Meow"

In Shopping malls behave like Customer

In Bus behave like Passenger

In School behave like Student

At Home behave like Son Tutorial4us.com

Basic Concept of OOPs

❑ **Polymorphism**

➢ Polymorphism is a Greek term which **means ability to take more than one form.**

➢ For example, + is used to make sum of two numbers as well as it is used to combine two strings.

➢ This is known as **operator overloading** because same operator may behave differently on different instances.

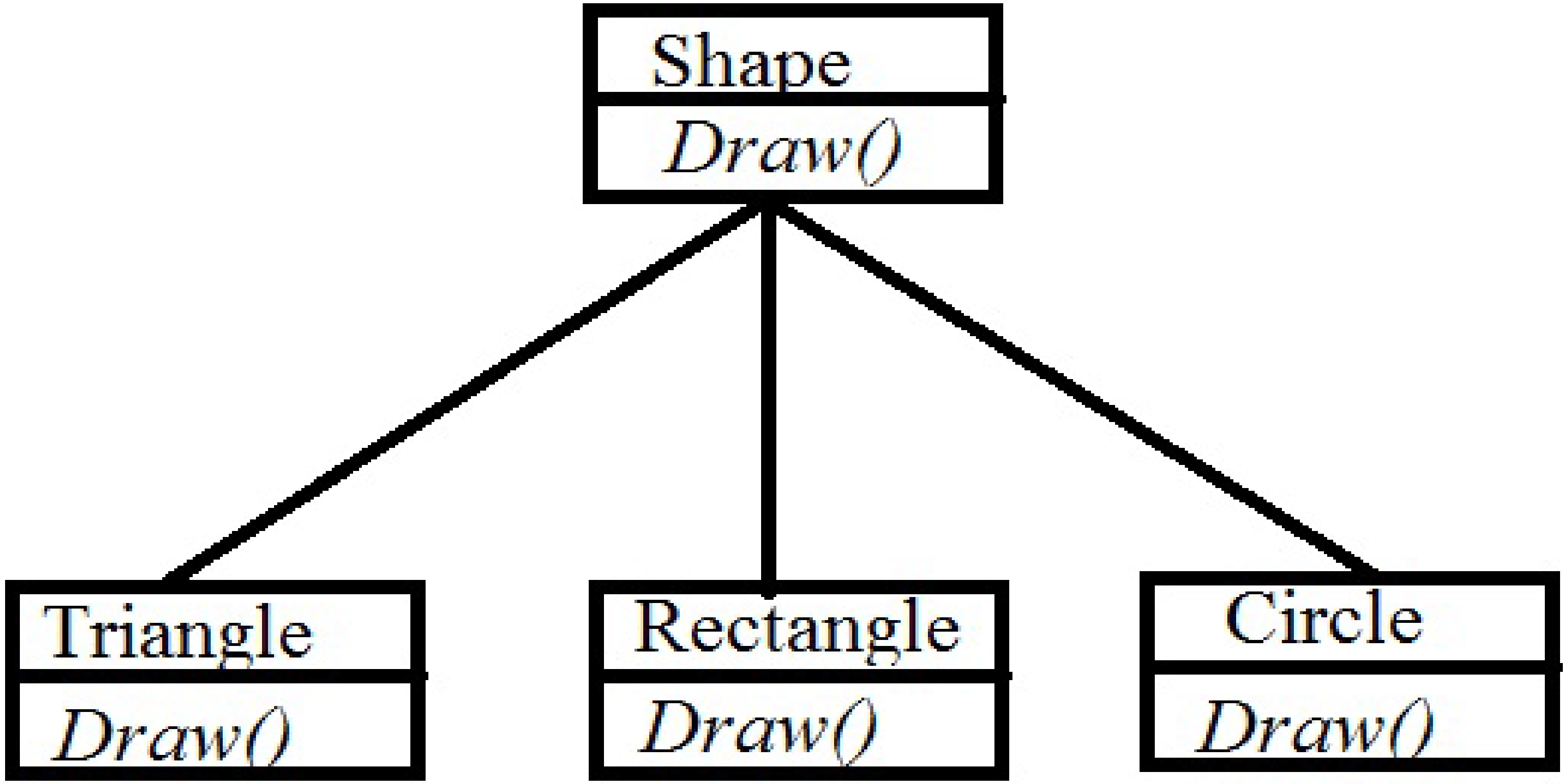
➢ Similarly, **functions can be overloaded**.

➢ For example, sum() function takes two arguments or three arguments, etc. eg: sum (8,4) or sum (4,8,9).

Basic Concept of OOPs

- **Polymorphism**

- Single function Draw() draws different objects.



Basic Concept of OOPs

❑ Inheritance

Son I am Base Class
Papa I am Derive Class
Inheritance

Basic Concept of OOPs

❑ **Inheritance**

➢ The mechanism of **deriving** a **new class** from an **old class** is called **inheritance** or **derivation**.

➢ The **old class** is known as **base class** while the **new class** is known as **derived class** or **sub class**.

➢ The inheritance is the most powerful feature of OOP.

➢ Through effective use of inheritance, we can **save lot of time** in programming and also **reduce the errors**.

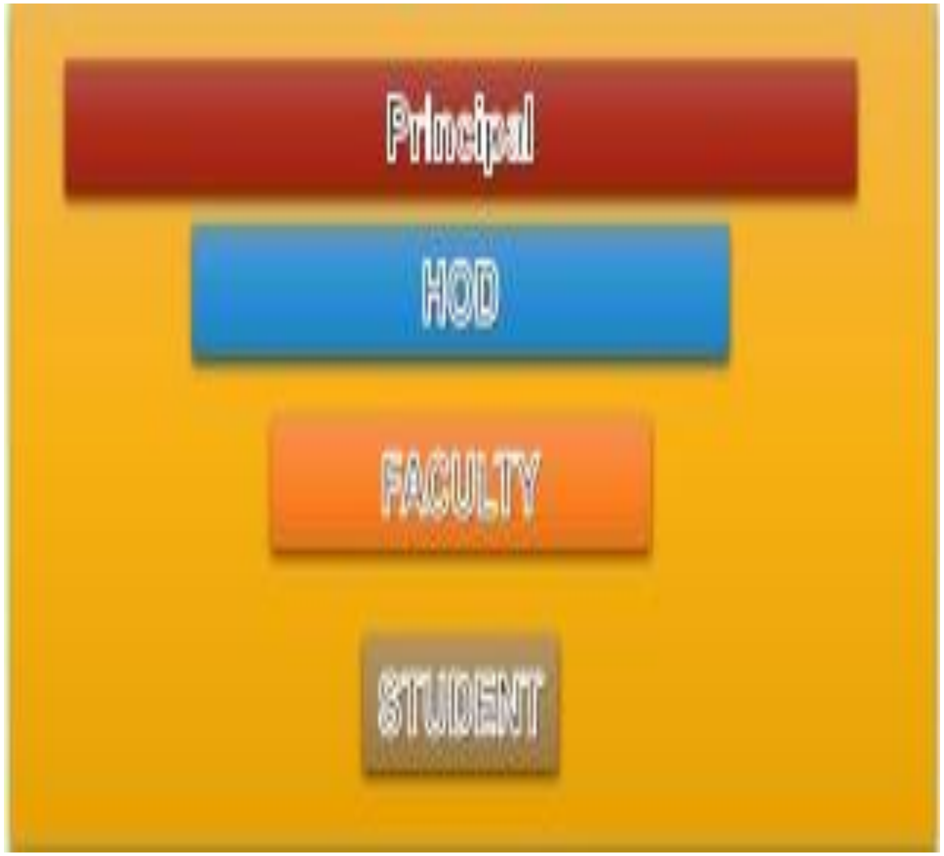
➢ Thus, increasing the **quality of work** and **productivity**.

Basic Concept of OOPs

❑ **Inheritance**

➢ The different types of inheritance are:

1. Single Inheritance
2. Hierarchical Inheritance
3. Multi Level Inheritance



Association

- Association **establishes relationship between two separate classes through their objects.** The relationship can be one to one, One to many, many to one and many to many.
- It refers to how objects are related to each other and how they are using each other's functionality. Composition and aggregation are two types of association.

- **Composition [Has-A relationship]**
  The composition is the strong type of association. An association is said to composition **if an Object owns another object and another object cannot exist without the owner object.** Consider the case of Human having a heart. Here Human object contains the heart and heart cannot exist without human.

- **Aggregation [Has-A relationship]**
  Aggregation is a **weak association**. An association is said to be aggregation if **both Objects can exist independently.** For example, a Team object and a Player object. The team contains multiple players, but a player can exist without a team.

Introduction

- Java is a **programming language** and a **platform**.
- Java is a high level, robust, object-oriented and secure programming language.

History

- In 1991, a group of Sun Microsystem engineers: Patrick Naughton, Mike Sheridan and James Gosling(called Green Team), decided to design a computer programming language that could be used for consumer electronic devices like cable TV, set-top-boxes.
- Initially, James Gosling decided to name this language as **‘Greentalk’**, followed by **‘Oak’**(because of the Oak tree that was outside his window at his office).
- Later realized that, there was an existing programming language called Oak, so renamed it by ***Java***, from Java coffee.

Types of Applications

- By using Java, **four types of applications** can be implemented. They are:
  - **Standalone Applications**
  - Known as Desktop applications or windows based applications.
  - Need to install in each computer systems.
  - Example: Media player, Antivirus, etc.
  - AWT and Swing are used in Java for creating standalone applications.

  - **Web Applications**
  - An application that runs on the server side and creates dynamic page.
  - Currently, Servlet, JSP, Struts, Spring, Hibernate, JSF, etc. technologies are used for creating web applications in Java.

  - **Enterprise Applications**
  - Applications that are distributed in nature.
  - EJB is used for creating enterprise applications.

  - **Mobile Applications**
  - Applications which are created for mobile usages.
  - Android and Java ME are used for creating mobile applications.

Features of Java

- **Simple Small and Familiar:** Java is a **simple language** because it contains many features of other languages like C and C++ and Java removes complexity because it doesn't use pointers, Storage Classes and Go to Statements and java doesn't support Multiple Inheritance.

- **Object-Oriented:** Java is purely **OOP language** because all the code of the java language is written into the classes and objects. Java is most popular language because it also supports Code Reusability, Maintainability etc.

- **High performance:** Java programs are **complied to portable intermediate** form know as **bytecodes**. This architecture means that Java programs are faster than program or scripts written in purely interpreted languages but slower than C and C++ programs that compiled to native machine languages.

Features of Java(Contd.)

- **Architectural Neutral**: One of the key feature of Java that makes it different from other programming languages is **architectural neutral** (or **platform independent**). This means that the programs **written on one platform can run on any other platform** without having to rewrite or recompile them. In other words, it follows '**Write-once-run-anywhere**' approach.

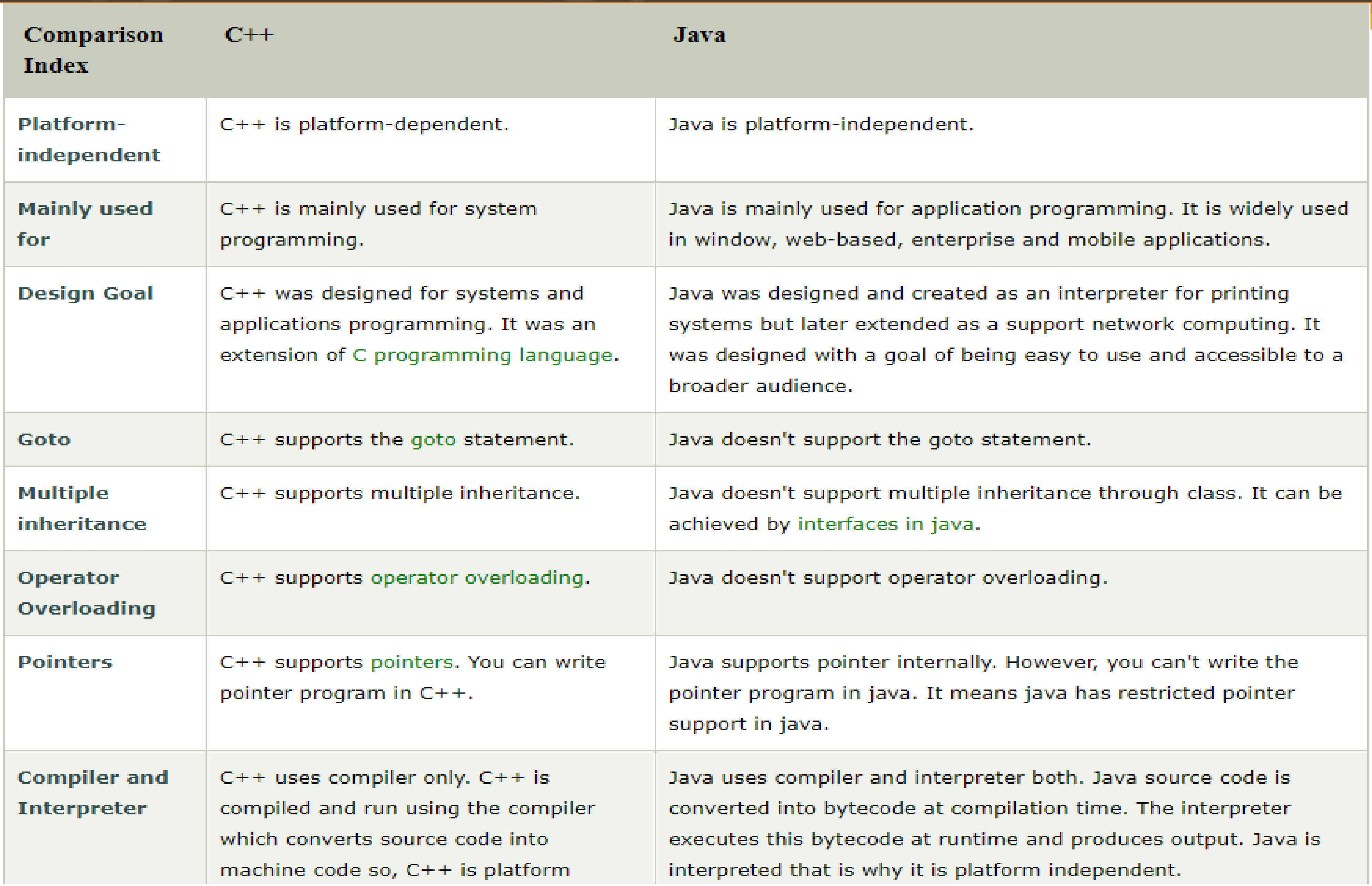
- **Multithreaded and Interactive:** Java uses **Multithreaded Techniques:** For Execution means that the code of java is divided into the smaller parts and are executed by java in Sequence and Timing Manner this is Called as **Multithreading.**

- **Interpreted and Compiled**: Unlike most of the programming languages which are either compiled or interpreted, Java is both compiled and interpreted. The **Java compiler** translates a **java source file to bytecodes** and the **Java interpreter executes the translated byte codes** directly on the system that implements the Java Virtual Machine. These two steps of compilation and interpretation allow extensive code checking and improved security .

Features of Java(Contd.)

- **Robust and Secure:** Java has to be consistent on a variety of systems. Java helps us find our mistakes early in program development. It is a strongly typed language. There is a **lack of pointers** that avoids security problems. There is **automatic garbage collection** in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- **Portable :** The portability actually comes from architecture-neutrality. In C/C++, source code may **run slightly differently on different hardware** platforms. In Java, it has been simplified.
- **Dynamic and Extensible Code:** Java has dynamic and extensible code means with the help of OOPS, java provides inheritance which helps **reuse the code** that is pre-defined and also uses all the built in functions of java and classes.
- **Distributed:** Java is a **distributed language** which means that the program can be design to **run on computer networks**. Java provides an extensive library of classes for communicating ,using TCP/IP protocols such as HTTP and FTP. This makes creating network connections much easier than in C/C++.

Difference between C++ and JAVA

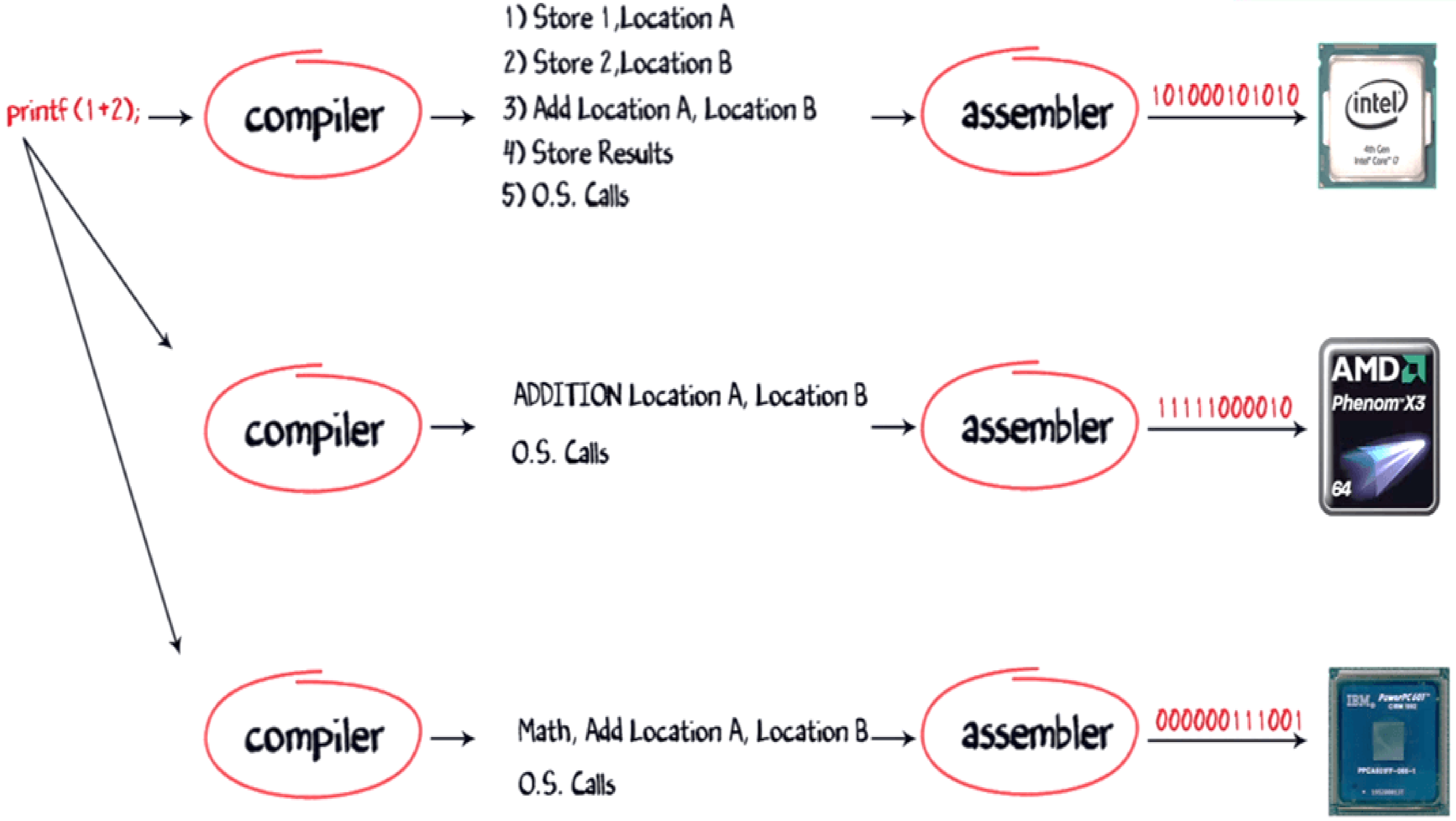


Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Design Goal	C++ was designed for systems and applications programming. It was an extension of C programming language.	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform	Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.

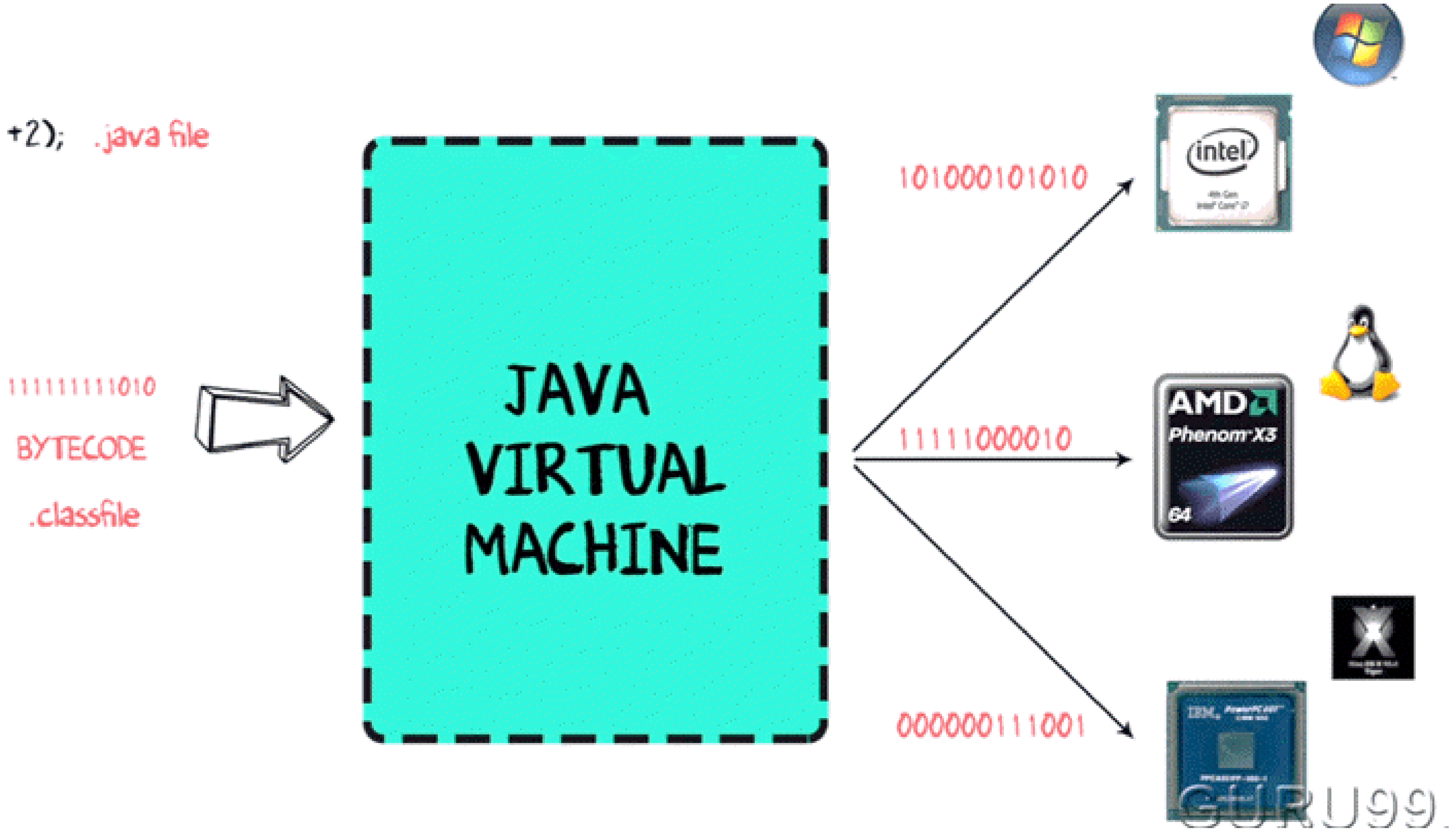
Java Runtime Environment(JRE)

- The JRE is the **smallest set of executables and files** that constitute the standard **java platform**.
- The **Java Runtime Environment (JRE)** contains the class libraries, the Java Virtual Machine, and other supporting components.
- The Java source code is compiled into bytecode by Java Compiler.
- This **bytecode** will be stored in **class files**.
- During runtime, this bytecode will be loaded, verified and JVM interprets the bytecode into machine code which will be executed in the machine in which the Java program runs.
- A JRE performs the following tasks:
  - **Loads the class(done by class loader)**
  - **Verifies the bytecode(this is done by bytecode verifier)**
  - **Interprets the bytecode(This is done by JVM)**

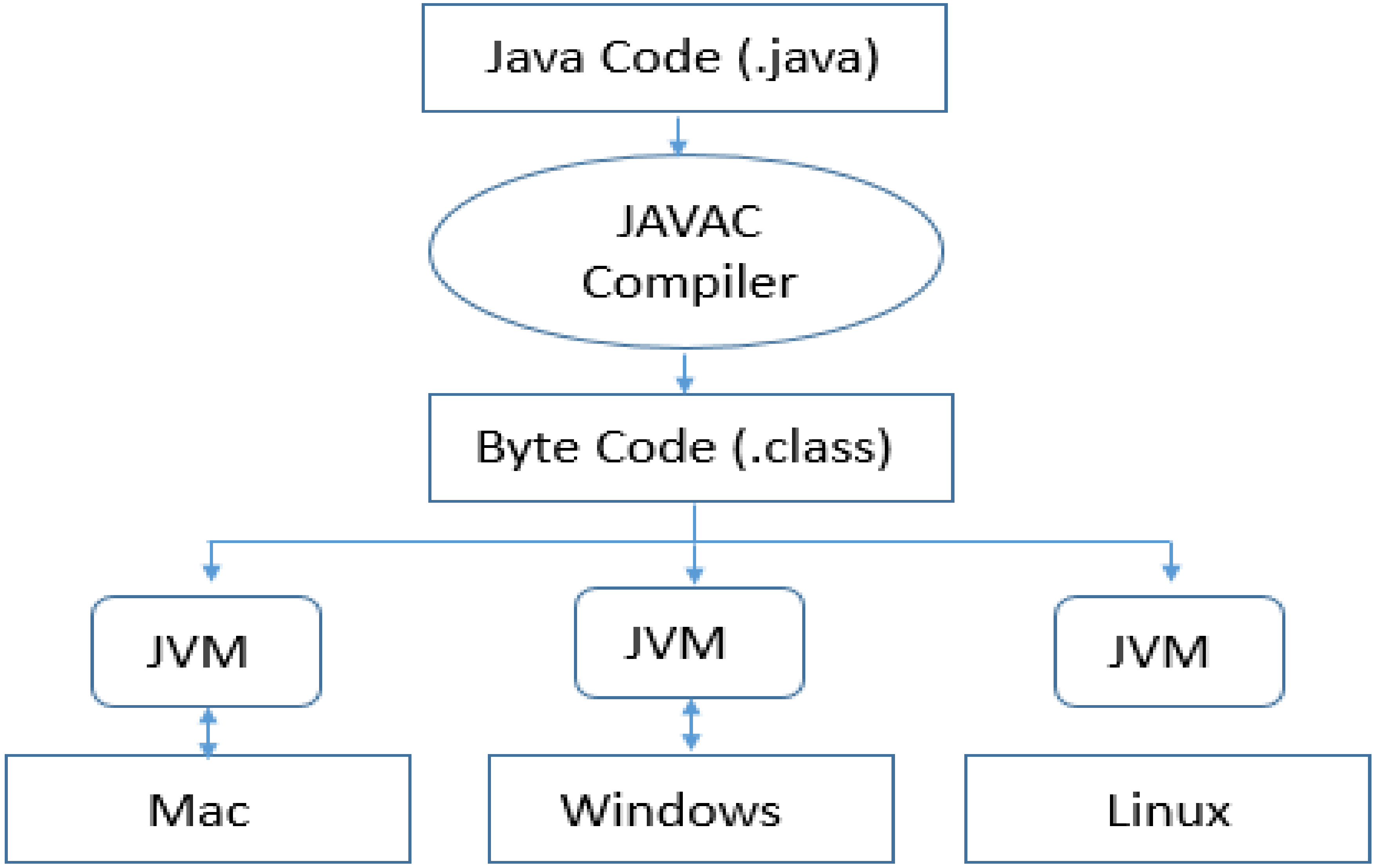
Without JVM(Reference Slide)



With JVM(Reference Slide)

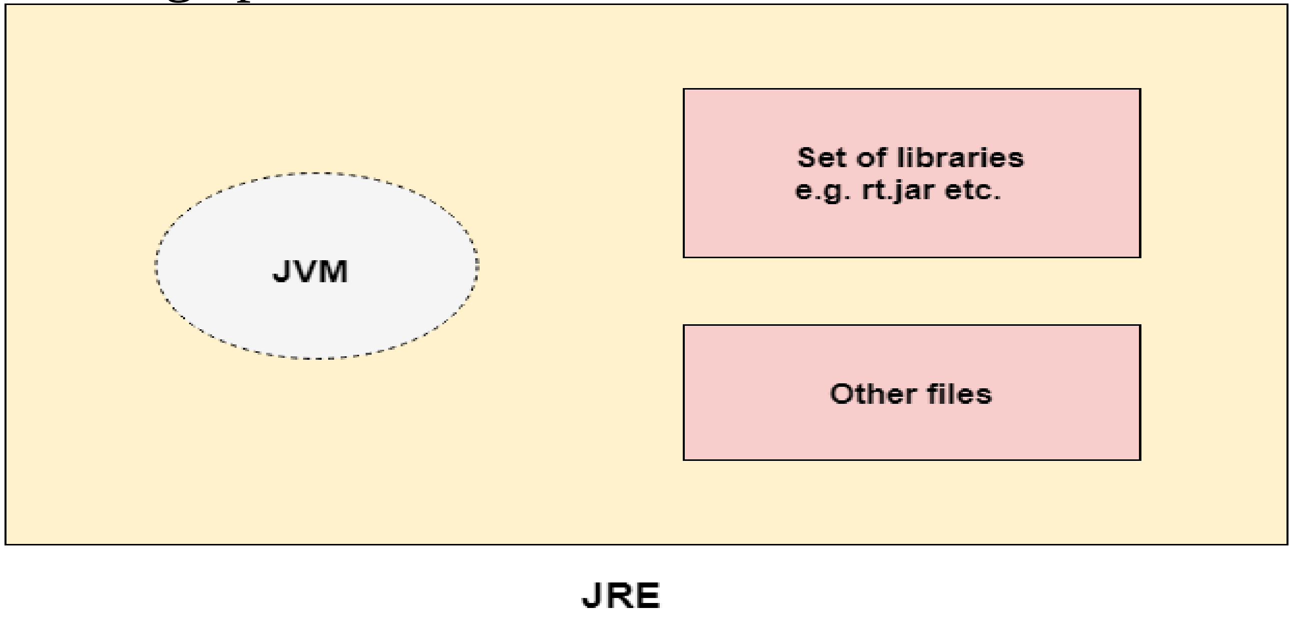


Java Virtual Machine(JVM) or Java Bytecode or Java Class Files



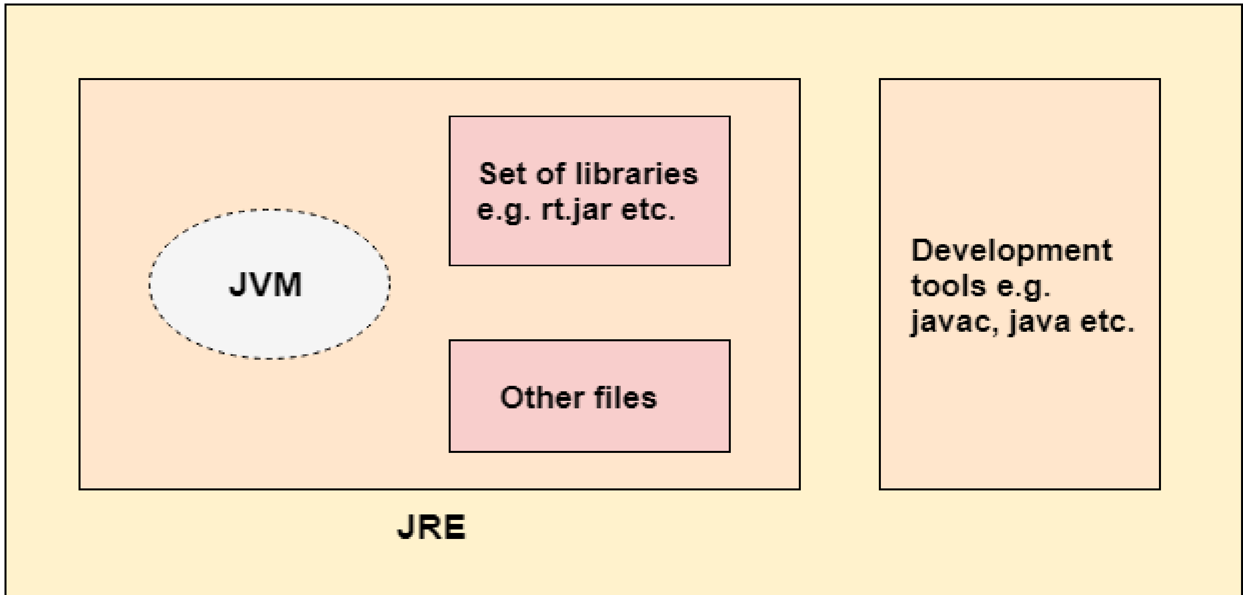
Java Virtual Machine(JVM) or Java Bytecode or Java Class Files

- Java Virtual Machine (JVM) is a specification that provides runtime environment in which java **bytecode** can be executed. As the name implies, the JVM acts as a “virtual” machine or processor.
- Java's platform independence consists mostly of its **Java Virtual Machine (JVM)** . JVM makes this possible because it is aware of the specific instruction lengths and other particularities of the platform.
- The JVM performs following operation:
  - **Loads code**
  - **Verifies code**
  - **Executes code**



Java Development Kit

- 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
  - JavaFX


JVM
Set of libraries e.g. rt.jar etc.
Other files
Development tools e.g. javac, java etc.
JRE