Introduction to OOP:

Introduction to OOP: Need of object oriented programming, principles of object oriented languages, OOP languages and Applications, history of JAVA, Java Virtual Machine, Java features.

Programming Constructs: Primitive data types, Identifiers- Naming Conventions, Keywords, Literals, Operators- Binary, Unary and Ternary, Expressions, Primitive Type conversion and casting, Control statements.

1.1 Need of object oriented programming:

There are some drawbacks in procedure oriented programming that�strongs why there is a need of OOP.

Draw backs of POP:

- ✓ The problem is viewed as sequence of things(tasks) to be done. Functions are written to accomplish these tasks (sequence of things).
- ✓ Concentration is given on functions but not data.(Low security). Data is exposed to whole program at once, so there is no security of data available.
- ✓ Does not models real world problems very well.(Since the focus is on the instructions, it is rather difficult to relate to real world objects and in transition some real world problems.)
- ✓ The code re-usability feature is not present in the procedure oriented programming. we have to write the same programming code to many times .
- ✓ We cannot perform Encapsulation , Inheritance etc in the procedure oriented programming.
- ✓ Difficult to create new data types reduces extensibility

OOP Advantages:

- ➢ OOP provides a clear modular structure for programs.(*Code Modularity,* Everything in OOP is an object; these objects can be interchanged or removed to meet the users' needs .)
- ➢ OOP makes it easy to maintain and modify existing code

- OOP provides a good framework for code libraries where supplied software components can be easily adapted and modified by the programmer.

- It is suitable for real word problems and real world works. And it supports code reusability.

- It helps to reduce the complexity and also improves the maintainability of the system. When combined with the concepts of the Encapsulation and Polymorphism, Abstraction gives more power to the Object oriented programming languages.

- *Easier maintenance :*Inheritance usually reduces maintenance because of the 'domino effect it has on derived classes when a change is made in a base class.

- *Design stability:* Once a stable base class has been developed, the new classes that are derived may have fewer less errors and bugs.

- *Improved communication between developers and users* Objects can be broken down into real life entities, hence it is easier and it communicates ideas.

- *Seamless transition from design to implementation* This is mainly because communications are improved.

- *Code reusability:* It means reusing some facilities rather than building it again and again. This is done with the use of a class. We can use it ‗n' number of times as per our need.

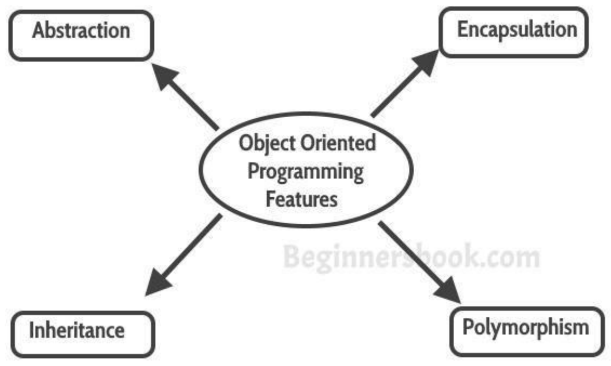
- New objects can be derived from old objects, allowing for improvement and refinement of the code at each stage and also preserving parts of the code for other programs. This is used to develop many class libraries using class codes that have already been written.

1.2 principles of object oriented languages.

Object-oriented programming System(OOPs) is a programming paradigm based on the concept of ―objects‖ that contain data and methods. The primary purpose of object-oriented programming is to increase the flexibility and maintainability of programs. Object oriented programming brings together data and its behaviour(methods) in a single location(object) makes it easier to understand how a program works.

Smalltalk is considered the first truly object-oriented programming language.

Simula is considered the first object-oriented programming language. The programming paradigm where everything is represented as an object is known as a truly object-oriented programming language.



These four features are the main OOPs Concepts/principles that you must learn to understand the Object Oriented Programming in Java

Class and objects

A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties(variables) or methods(functions) that are common to all objects of one type. Class in Java determines how an object will behave and what the object will contain.

Class: it is a blueprint. From it we can derive object.

Object: is a bundle of data(variable) and its behaviour(methods).Objects have two characteristics: They have **states** and **behaviors**.

Examples of states and behaviors Example 1:

Class : House
Object: tejaHouse
State: city, Color.
Behavior: Open door, close door,getdata,display

So if I had to write a class based on states and behaviours of House. I can do it like this: States can be represented as instance variables and behaviours as methods of the class.

```
import java.util.Scanner;

class House{
         String city;
        String color;
        Scanner input = new Scanner(System.in);

   void getdata()
     {
        System.out.print("Enter the city name and color of the house  \n");
city = input.next();             color = input.next();
   }

    void display()
{
      System.out.print("The city name and color name of the house is    \n "
+city+"\n"+color+"\n");
     }

void openDoor()
    {
      System.out.println("The door is opened \n");
   }

 void closeDoor()
 {
      System.out.println("The door is closed \n");
}

public static void main(String arh[])
 {
     House tejaHouse=new House();
     House raviHouse=new House();
    tejaHouse.getdata();
raviHouse.getdata();
```

```
    System.out.println("The tejaHome details
are:");      tejaHouse.display();
tejaHouse.openDoor();
     tejaHouse.closeDoor();

System.out.println("The    raviHome    details
are:");                         raviHouse.display();
raviHouse.openDoor();
raviHouse.closeDoor();  }
}
```

Output:

```
Enter the city name and color of the
house   bhimavaram black
```

```
Enter the city name and color of the house
elur
blue

The tejaHome details are:
The city name and color name of the house is
bhimavaram
black
The door is opened
The door is closed

The raviHome details are:
The city name and color name of the
house is     elur blue
The door is opened
The door is closed
```

In the above program I have created two objects(tejaHome and raviHome) for the class Home. Every object is having their own properties.i.e tejaHome is having its city.color,getdata(),display(),openDoor(),closeDoor() and the object raviHome is also having its own members/properties city.color, getdata(), display(), openDoor(), closeDoor().

Hence we conclude that every object is having its own members which are declared in the class.

Encapsulation:

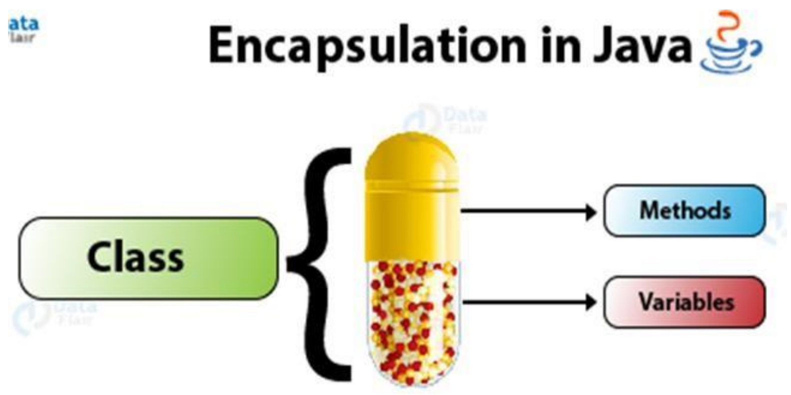
Encapsulation is the action of enclosing something.in Java, data and function are enclosed. In simple terms, Encapsulation is the wrapping up of data & function under a single unit and that unit name is class name.

We can achieve this OOPs feature by declaring all the variables in the class as private and a method which will set and get the value of variables as public.

If you are creating class it means you are doing encapsulation. The variables of a class are always hidden from other classes. It can only be accessed using the methods of their current class.

It basically creates a shield and the code cannot be accessed outside the shield or by any code outside the shield. Java Beans class is the example of a fully encapsulated class.

- The ***variables in Java*** or the method of the class are hidden from any other class and cannot be accessed outside the class.
- We can also call it, as Data-hiding.
- The encapsulated class is easy to test.
- Standard IDE�much s like Eclipse, Netbeans are providing the facility to generate getter setter methods, so it is very easy to create an encapsulated call.
- It can be achieved by declaring the class as private while the methods as public so that the variables can be accessed.



Advantages of Encapsulation in Java: These are benefits of Encapsulation in Java:

Data Hiding – It can provide the programmer to hide the inner classes and the user to give access only to the desired codes. It allows the programmer to not allow the user to know how variables and data store.
Getter and Setter Methods – Private member can only be accessed within the same class. An outside class can not access the data members of that class. If you need to access these variables, you have to use public ―getter‖ and ―setter‖ methods.

- **Flexibility –** With this, we can make the data as read-only or write-only as we require it to be. It also improves the maintainability and flexibility of code.
- **Reusability –** It allows the user to a programmer to use the existing code again and again in an effective way.
- **Testing of the code –** Ease of testing becomes easy. So it is better for Unit testing

The following two programs are same. In program 1 we used two classes and in program 2 we used single class. The following two programs will execute successfully.

Program:1

```
class      Person      {
private int age;

  public    int    getAge()    {
return age;
  }

  public    void    setAge(int    k)    {
age= k;
  }
}

class Main {
  public static void main(String[] args) {
Person      p1      =      new      Person();
p1.setAge(24);
     System.out.println("My age is " +
p1.getAge());
}  }
```

Output:

```
My age is 24
```

Program:2 class

```
Main{      private
int age;

  public    int    getAge()    {
return age;
  }

  public    void    setAge(int    k)    {
age= k;
  }


  public static void main(String[] args) {
Main p1 = new Main();
p1.setAge(24);
     System.out.println("My age is " +
p1.getAge());
}  }
```

Output:

```
My age is 24
```

Execution of above program is as followes

1) Make the instance variables(age) private so that they cannot be accessed directly from outside the class. You can only set and get values of these variables through the methods of the class.
2) Have getter(getAge) and setter (setAge) methods in the class to set and get the values of the fields.

Abstraction

Abstraction is a process where you show only ―relevant‖ data and ―hide‖ unnecessary details of an object from the user.

For example, when you login to your bank account online, you enter your user_id and password and press login, what happens when you press login, how the input data sent to server, how it gets verified is all abstracted away from the you.

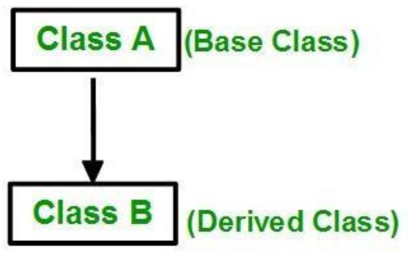
Another example of abstraction: A car in itself is a well-defined object, which is composed of several other smaller objects like a gearing system, steering mechanism, engine, which are again have their own subsystems. But for humans car is a one single object, which can be managed by the help of its subsystems, even if their inner details are unknown. When driver press the accelerator the car move fastly and it will hide the internal details of how accelerator is working. This kind of hiding mechanism is known as abstraction.

Java Abstraction can be achieved by using ***abstract classes and interfaces***.

Inheritance
In general, we can say a child inherits his father�even's properties and his father inherited from his father and this process goes on. The basic principle of inheritance is passing the properties. *Java Inheritance* is one of the most important features of object-oriented programming, that makes it very useful.

It�even's creating a parent-child relationship between two classes.The process by which one class acquires the properties and functionalities of another class is called inheritance.

Inheritance provides the idea of reusability of code and each sub class defines only those features that are unique to it, rest of the features can be inherited from the parent class.



1. Inheritance is a process of defining a new class based on an existing class by extending its common data members and methods.

2. Inheritance allows us to reuse of code, it improves reusability in your java application.
3. The parent class is called the **base class** or **super class**. The child class that extends the base class is called the derived class or **sub class** or **child class**.

Inheritance in Java, allows us to carry the features of the parent class to the subclasses. Inheritance is the mechanism in which derived class acquires the properties of a base class. Here base class is the superclass and derived class is the subclass.

Some of the important terms used in Inheritance are:

- **Class:** Class in Java is basically a group of objects. It is a user-defined data type or blueprint from which objects are created.
- **Super Class** – A superclass in Java is the class from which the features are inherited. It is also called a parent class
- **Sub Class** – A subclass in Java is the class that inherits the features of the superclass. It can have its own features and methods as well. It is a child class.
- **Reusability** – The concept of reusability is supported by inheritance, the codes inside the superclass can be reused by the subclasses.

Syntax of Java Inheritance

In Java, ‗extends‛ keyword is used to inherit a class. Here the BaseClass is the parent class from which the DerivedClass is acquiring the properties and DerivedClass is the derived class. The DerivedClass is inheriting the properties and methods of BaseClass.

```
class BaseClass
{
//methods and variables
}
class DerivedClass extends BaseClass
{
  //methods and variables  }
```

Program:
```
class Staff{
  String designation = "AssistantProfessor";
  String college = "svecw";
  void job(){
        System.out.println("Teaching");
  }
}

public class Javafaculty extends Staff{    String mainSubject = "java";    public static
void main(String args[]){
     Javafculyu obj = new Javafaculty();
     System.out.println(obj.college);
     System.out.println(obj.designation);
System.out.println(obj.mainSubject);
obj.job();
  }
}
```
Output: svecw
```
AssistantProfessor
java Teaching
```

Polymorphism

If *one task is performed in different ways*, it is known as polymorphism. let‛s understand the concept of Polymorphism in Java with a real-time example. Consider a woman can have different characteristics at the same time as she‛s a mother, a wife, an employee, a daughter and so on. So the same person has different behavior in different situations.

Polymorphism is the ability to take different forms. The word —**poly**‖ means *many* and —**morph**‖ means *forms*, so it means having many forms. It is the most important concept of object-oriented programming language.

Java Polymorphism allows us to perform a single task in different ways. It is used when we have multiple classes that are related to each other by inheritance.

```
public class Animal
{
        public void sound()
       {
          System.out.println("Animal is making a sound");
       }
}
```

Now, for example, we have two subclasses of Animal: Frog and Cow that extends Animal class. We can implement the same method like this:

```
public class Frog extends Animal
{
public void sound()
{
System.out.println("Turrrr");
}
}
```

And

```
public class Cow extends Animal
{

public void sound()
{
System.out.println("Moooooo");
}
}
```

In the following example, we have a common method sound but with this method, we are using different ways to do the same action. This is a perfect example of polymorphism. Now the question is how to access this method(all classes are having same method name)

Types of Polymorphism

1) Static Polymorphism
2) Dynamic Polymorphism

Benefits of OOP:

- It is easy to model a real system as real objects are represented by programming objects in OOP. The objects are processed by their member data and functions. It is easy to analyze the user requirements.

- With the help of inheritance, we can reuse the existing class to derive a new class such that the redundant code is eliminated and the use of existing class is extended. This saves time and cost of program.
- In OOP, data can be made private to a class such that only member functions of the class can access the data. This principle of data hiding helps the programmer to build a secure program that can not be invaded by code in other part of the program.
- With the help of polymorphism, the same function or same operator can be used for different purposes. This helps to manage software complexity easily.
- Large problems can be reduced to smaller and more manageable problems. It is easy to partition the work in a project based on objects.
- It is possible to have multiple instances of an object to co-exist without any interference i.e. each object has its own separate member data and function.

1.3 OOP languages and Applications.

JAVA was developed by Sun Microsystems Inc in 1991, later acquired by Oracle Corporation. It was developed by James Gosling and Patrick Naughton. It helps to create modular programs and reusable code.

While Simula is credited as the first object-oriented programming language, the most popular OOP languages are:

- Java.
- JavaScript.
- Python.
- C++
- Visual Basic .NET.
- Ruby.
- Scala.
- Smalltalk.
- PHP etc

Applications of OOP:

1. **Client-Server Systems:**

Object-oriented Client-Server Systems provide the IT infrastructure, creating object-oriented Client-Server Internet (OCSI) applications.

Here, infrastructure refers to operating systems, networks, and hardware. OSCI consist of three major technologies:

- ➢ The Client Server
- ➢ Object-Oriented Programming ➢ The Internet

2. **Object-Oriented Databases:**

They are also called Object Database Management Systems (ODBMS). These databases store objects instead of data, such as real numbers and integers. Objects consist of the following:

- ✓ **Attributes:** Attributes are data that defines the traits of an object. This data can be as simple as integers and real numbers. It can also be a reference to a complex object.
- ✓ **Methods:** They define the behavior and are also called functions or procedures.

3. **Object Oriented Databases:**

These databases try to maintain a direct correspondence between the realworld and database objects in order to let the object retain their identity and integrity. They can then be identified and operated upon.

4. **Real-Time System Design:**

Real time systems inherit complexities that makes difficult to build them. Object-oriented techniques make it easier to handle those complexities. These techniques present ways of dealing with these complexities by providing an integrated framework which includes schedulability analysis and behavioral specifications.

5. **Simulation And Modelling System:**

It�behaviors difficult to model complex systems due to the varying specification of variables. These are prevalent in medicine and in other areas of natural science, such as ecology, zoology, and agronomic systems. Simulating complex systems requires modelling and understanding interactions explicitly. Object-oriented Programming provides an alternative approach for simplifying these complex modelling systems.

6. **Hypertext And Hypermedia:**

OOP also helps in laying out a framework for Hypertext. Basically, hypertext is similar to regular text as it can be stored, searched, and edited easily. The only difference is that hypertext is text with pointers to other text as well.

Hypermedia, on the other hand, is a superset of hypertext. Documents having hypermedia, not only contain links to other pieces of text and information, but also to numerous other forms of media, ranging from images to sound.

7. **Neural Networking And Parallel Programming:**
It addresses the problem of prediction and approximation of complex timevarying systems. Firstly, the entire time-varying process is split into several time intervals or slots. Then, neural networks are developed in a particular time interval to disperse the load of various networks. OOP simplifies the entire process by simplifying the approximation and prediction ability of networks.

8. **Office Automation Systems:**
These include formal as well as informal electronic systems primarily concerned with information sharing and communication to and from people inside as well as outside the organization. Some examples are:

- Email
- Word processing
- Web calendars
- Desktop publishing

9. **CIM/CAD/CAM Systems:**
OOP can also be used in manufacturing and design applications as it allows people to reduce the effort involved. For instance, it can be used while designing blueprints, flowcharts, etc. OOP makes it possible for the designers and engineers to produce these flowcharts and blueprints accurately.

10. **AI Expert Systems:**
These are computer applications which are developed to solve complex problems pertaining to a specific domain, which is at a level far beyond the reach of a human brain.It has the following characteristics:

- ✓ Reliable
- ✓ Highly responsive
- ✓ Understandable
- ✓ High-performance

1.4 History 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 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, ebusiness solutions, etc. There 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 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 named "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.

7) **Why had they chosen java name 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.

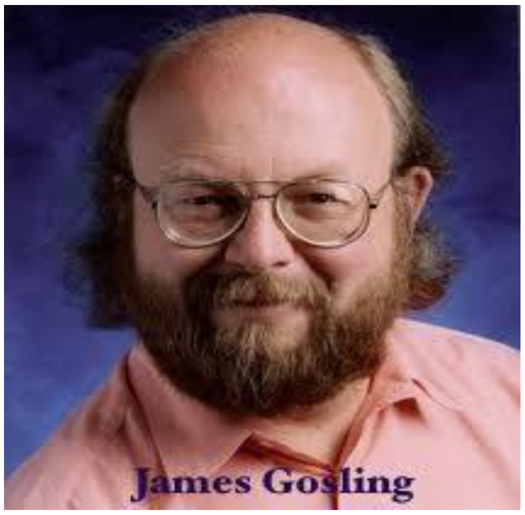
8) Java is an island of Indonesia where the first coffee was produced (called java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having coffee near his office.

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.

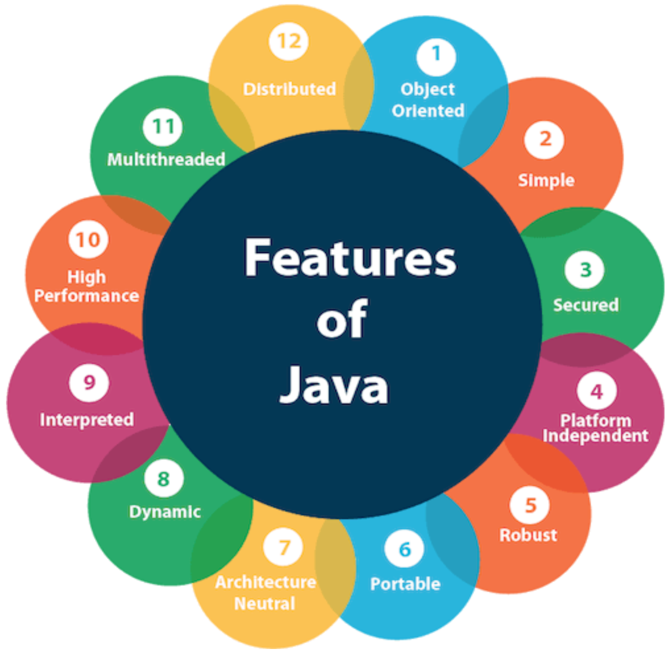
11) In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.

12) JDK 1.0 released in(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 the new features in Java.



Features of Java

The primary objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as java *buzzwords*.
A list of most important features of Java language is given below.



1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Interpreted
9. High Performance
10. Multithreaded
11. Distributed
12. Dynamic

Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun, Java language is a simple programming language because: ○ Java syntax is based on C++ (so easier for programmers to learn it after C++).

- ○ Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- ○ There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

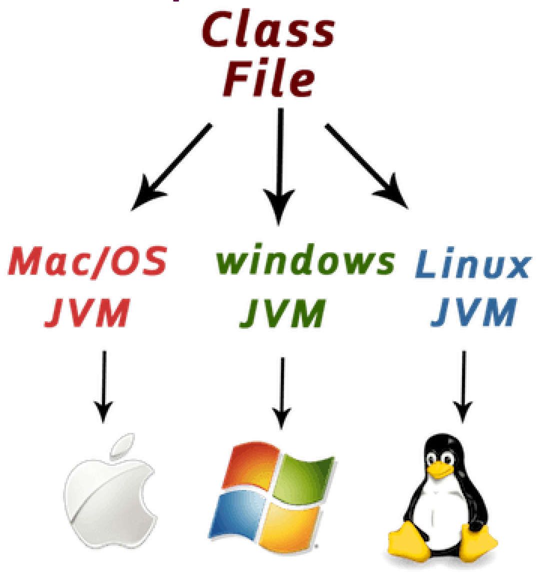
Object-oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.

Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.
Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

Platform Independent



Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

There are two types of platforms software-based and hardware-based. Java provides a software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

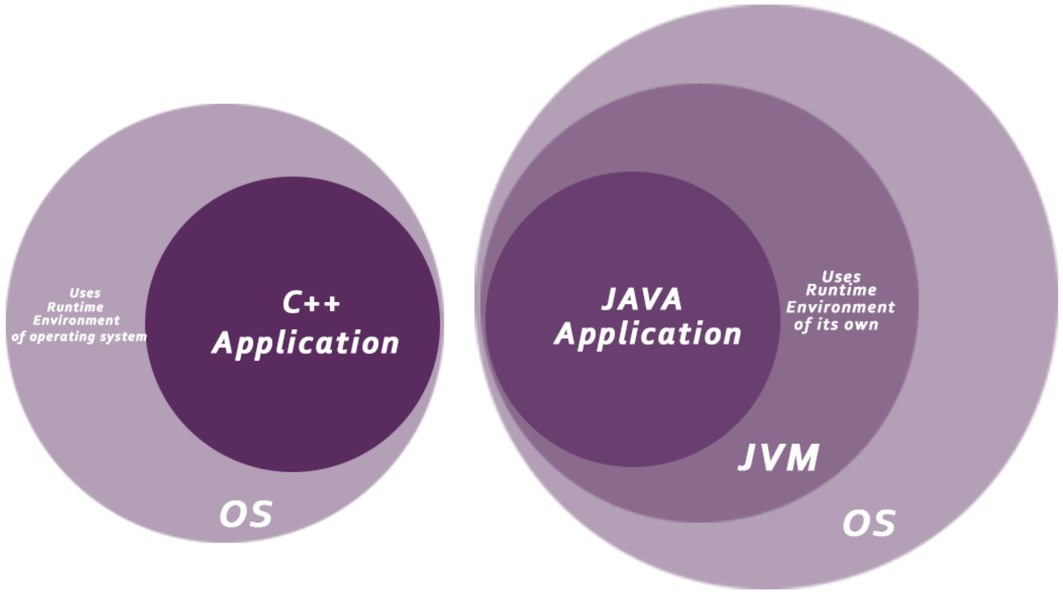
1. Runtime Environment
2. API(Application Programming Interface)

Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere(WORA).

Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

- **No explicit pointer**
- **Java Programs run inside a virtual machine sandbox**


Uses Runtime Environment of operating system
C++ Application
OS
JAVA Application
Uses Runtime Environment of its own
JVM
OS

- **Classloader:** Classloader in Java is a part of the Java Runtime Environment(JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.

Java, unlike other languages, runs all of its programs inside a sandbox of its own called the Java Virtual Machine so that any errors or crashes do not harm the external operating system, thus making it secure and efficient at the same time.

- **Bytecode Verifier:** It checks the code fragments for illegal code that can violate access right to objects.

  Sometimes malicious code can try to access objects outside their permission domains. The bytecode verifier prevents this by obliterating such code.
- **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.

  Java can determine what resources a particular class can access such as reading from a disk and writing data to memory. It does so with the help of a security manager.

Java language provides these securities by default. Some security can also be provided by an application developer explicitly through SSL, JAAS, Cryptography, etc.

Robust
Java is a strong language. Does it mean it can lift mountains? Metaphorically, yes. It does so by implementing some interesting features from which, a few are as follows:

Robust simply means strong. Java is robust because:

- It uses strong memory management.(Java uses strong memory management techniques so that there is no space for improper memory assignment during the running of a program.)
- There is a lack of pointers that avoids security problems. Malicious code can use explicit pointers to access code which is outside their allowance or restricted,sensitive data. Hence Java has no support for pointers.

- 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.

  The constant problem of unreferenced objects clogging the memory is gone. Earlier there was this huge problem of unreferenced objects still being in the memory which led to the wastage of space. But with the advent of Java�that's garbage collector the problem fades away as it looks through the heap memory and discards the objects which are not used or not referenced anymore by the program.

- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

- ―I ran into an error and my compiler closed immediately‖ was a huge problem for developers who were developing applications before the advent of a feature called Error Handling. Java implemented the concept and allowed users to execute and do custom actions when the program crashed or reported an error.

Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.
In C programming or primitive languages were not neutral to the architecture of development environments. For example c had int data type occupies 2 bytes of memory for 32-bit architecture systems and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

Java decided to be neutral to all platforms which dramatically increased collaboration efficiency. Java code does not compile its code to platformspecific byte code but it‛s compiled into platform independent bytecode. This means that the generated class file can now run on different machines running different environments, different operating systems with the only requirement of a Java Virtual Machine to be present in each machine. This makes Java an architecturally robust and flexible programming language.

Portable

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. Native code is the code which is processor specific, ie. it has to be compiled to run with a specific processor, like Intel�much x86 class processor.It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

Microsoft‛s Intermediate language and Java Bytecode has native code for execution of an in-time compiler for faster performance.

Java is faster than a lot of other languages like python, however, this is an abstract concept as python also takes less time to develop due to the syntax and easily fabricated code design

Distributed

Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

Java supports multithreaded programming i.e, it supports multiple operations running at the same time. We can think of a thread as an individual operation or parts of the program using the processor. It increases the performance by decreasing the development time needed for a particular software.

The coding for a particular software becomes streamlined. The maintenance cost drops. However, all of these processes share the same memory slots as individual processes use the memory efficiently.

As a multiprocessor can effectively execute multiple threads at the same time, multiprogramming is a boon to the developer community. Java manages to cover all these points which makes it a super-efficient language.

Dynamic

Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

Java supports dynamic compilation and automatic memory management (garbage collection).

It‟s a requirement based language model. Classes are not loaded all at once. They jump into action only when an invoke operation executes or some data about the class is needed in the memory. Java finalizes invoking instructions during runtime. Ex- Runtime Polymorphism i.e function overriding.

What is JDK?

JDK is a software development environment used for making applets and Java applications. The full form of JDK is Java Development Kit. Java developers can use it on Windows, macOS, Solaris, and Linux. JDK helps them to code and run Java programs.

Why use JDK?

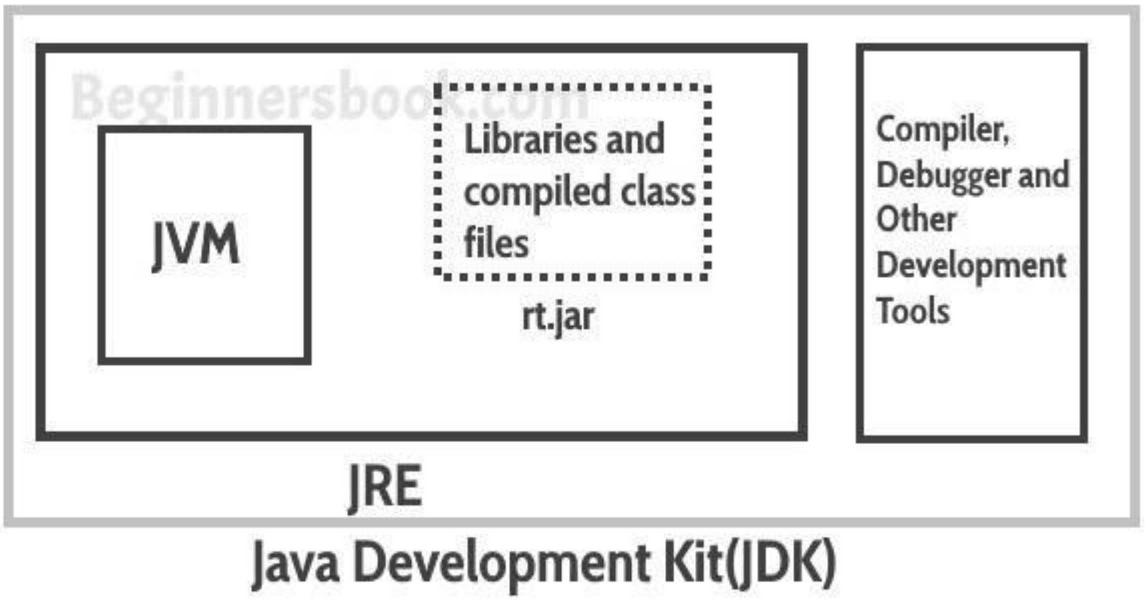
Here are the important reasons of using JDK:

- JDK contains tools required to write Java programs, and JRE to execute them.
- It includes a compiler, Java application launcher, Appletviewer, etc.
- Compiler converts code written in Java into byte code.
- Java application launcher opens a JRE, loads the necessary class, and executes its main method.

Features of JDK

Here are the important features of JDK:

- It enables you to handle multiple extensions in a single catch block.
- JDK includes all features that JRE has.
- It contains development tools such as a compiler, debugger, etc.
- JDK provides the environment to develop and execute Java source code.
- It can be installed on Windows, Unix, and Mac operating systems.
- Diamond operator can be used in specifying a generic type interface instead of writing the exact one.


Beginnersbook.com
JVM
Libraries and compiled class files
rt.jar
Compiler, Debugger and Other Development Tools
JRE
Java Development Kit(JDK)

What is JRE?

JRE is a piece of a software which is designed to run other software. It contains the class libraries, loader class, and JVM. In simple terms, if you want to run Java program you need JRE. If you are not a programmer, you don't need to install JDK, but just JRE to run Java programs. Though, all JDK versions comes bundled with Java Runtime Environment, so you do not need to download and install the JRE separately in your PC. The full form of JRE is Java Runtime Environment.

Why use JRE?

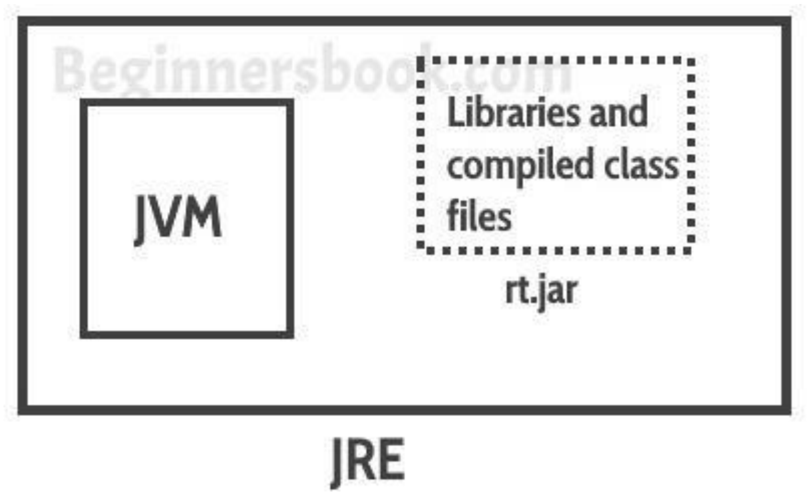
Here are the important reasons of using JRE:

- JRE contains class libraries, JVM, and other supporting files. It does not contain any tool for Java development like a debugger, compiler, etc.
- It uses important package classes like math, swingetc, util, lang, awt, and runtime libraries.
- If you have to run Java applets, then JRE must be installed in your system.

Features of JRE

Here are the important features of JRE:

- Java Runtime Environment is a set of tools using which the JVM actually runs.
- JRE contains deployment technology, including Java Web Start and Java Plug-in.
- Developers can easily run the source code in JRE, but he/she cannot write and compile the Java program.
- It includes integration libraries like Java Database Connectivity (JDBC), Remote Method Invocation (RMI), Java Naming and Directory Interface (JNDI), and more.
- JRE has JVM and Java HotSpot virtual machine client.



What is JVM?

JVM is an engine that provides a runtime environment to drive the Java Code or applications. It converts Java bytecode into machine language. JVM is a part of Java Run Environment (JRE). It cannot be separately downloaded and installed. To install JVM, you need to install JRE. The full form of JVM is Java Virtual Machine.

In many other programming languages, the compiler produces machine code for a specific system. However, Java compiler produces code for a virtual machine which is called as JVM.

Why JVM?

Here are the important reasons of using JVM:

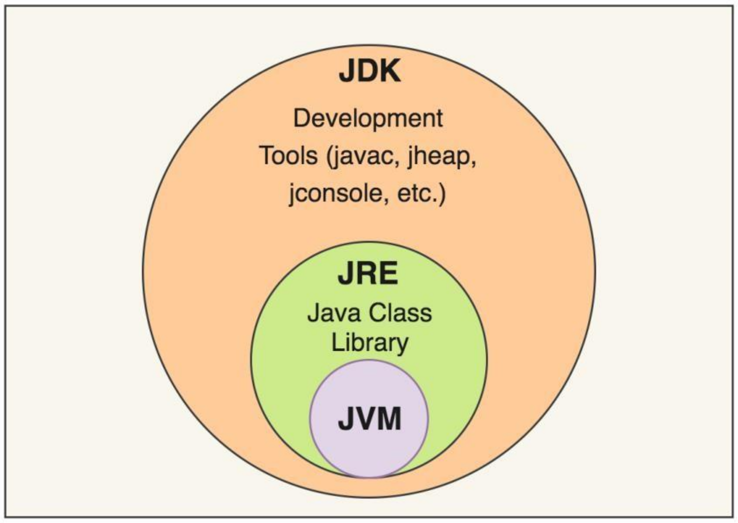
- JVM provides a platform-independent way of executing Java source code.
- It has numerous libraries, tools, and frameworks.

- Once you run Java program, you can run on any platform and save lots of time.
- JVM comes with JIT(Just-in-Time) compiler that converts Java source code into low-level machine language. Hence, it runs more faster as a regular application.

Features of JVM

Here are the important features of JVM:

- It enables you to run applications in a cloud environment or in your device.
- Java Virtual Machine converts byte code to the machine-specific code.
- It provides basic java functions like memory management, security, garbage collection, and more.
- JVM runs the program by using libraries and files given by Java Runtime Environment.
- JDK and JRE both contain Java Virtual Machine.
- It can execute the java program line by line hence it is also called as interpreter.
- JVM is easily customizable for example, you can allocate minimum and maximum memory to it.
- It is independent from hardware and the operating system. So, you can write a java program once and run anywhere.



KEY DIFFERENCES

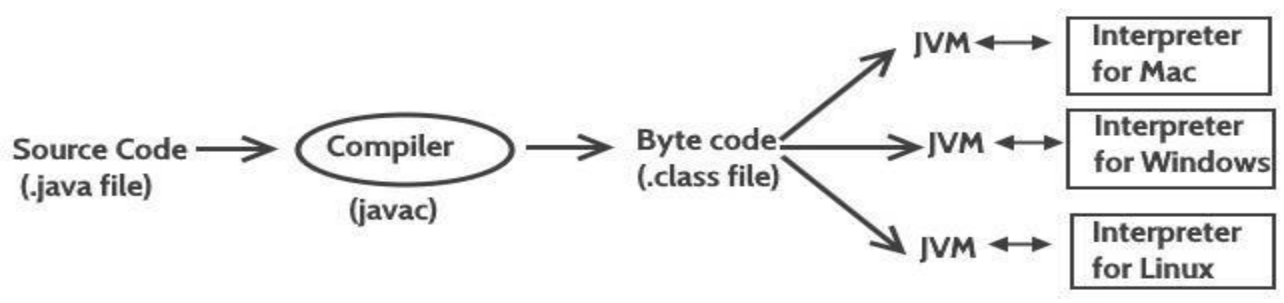
- JDK is a software development kit whereas JRE is a software bundle that allows Java program to run, whereas JVM is an environment for executing bytecode.
- The full form of JDK is Java Development Kit, while the full form of JRE is Java Runtime Environment, while the full form of JVM is Java Virtual Machine.
- JDK is platform dependent, JRE is also platform dependent, but JVM is platform independent.
- JDK contains tools for developing, debugging, etc. JRE contains class libraries and other supporting files, whereas software development tools are not included in JVM.
- JDK comes with the installer, on the other hand, JRE only contains the environment to execute source code whereas JVM bundled in both software JDK and JRE.

Java Virtual Machine (JVM)

Java is a high level programming language. A program written in high level language cannot be run on any machine directly. First, it needs to be translated into that particular machine language. The **javac compiler** does this thing, it takes java program (.java file containing source code) and translates it into machine code (referred as byte code or .class file).

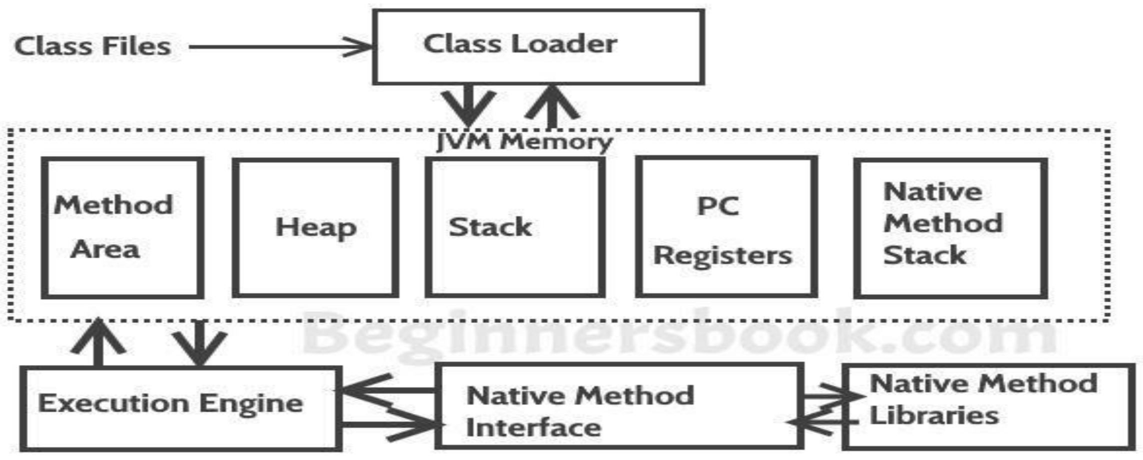
Java Virtual Machine (JVM) is a virtual machine that resides in the real machine (your computer) and the **machine language for JVM is byte code**. This makes it easier for compiler as it has to generate byte code for JVM rather than different machine code for each type of machine. JVM executes the byte code generated by compiler and produce output. **JVM is the one that makes java platform independent**.

So, now we understood that the primary function of JVM is to execute the byte code produced by compiler. **Each operating system has different JVM, however the output they produce after execution of byte code is same across all operating systems.** Which means that the byte code generated on Windows can be run on Mac OS and vice versa. That is why we call java as platform independent language. The same thing can be seen in the diagram below:



So to summarise everything: The Java Virtual machine (JVM) is the virtual machine that runs on actual machine (your computer) and executes Java byte code. The JVM doesn‘t understand Java source code, that‘s why we need to have javac compiler that compiles *.java files to obtain *.class files that contain the byte codes understood by the JVM. JVM makes java portable (write once, run anywhere). Each operating system has different JVM, however the output they produce after execution of byte code is same across all operating systems.

JVM Architecture



Lets see how JVM works:
Class Loader: The class loader reads the .class file and save the byte code in the **method area**.

Method Area: There is only one method area in a JVM which is shared among all the classes. This holds the class level information of each .class file.

Heap: Heap is a part of JVM memory where objects are allocated. JVM creates a Class object for each .class file.

Stack: Stack is a also a part of JVM memory but unlike Heap, it is used for storing temporary variables.
PC Registers: This keeps the track of which instruction has been executed and which one is going to be executed. Since instructions are executed by threads, each thread has a separate PC register.

Native Method stack: A native method can access the runtime data areas of the virtual machine.

Native Method interface: It enables java code to call or be called by native applications. Native applications are programs that are specific to the hardware and OS of a system.

Garbage collection: A class instance is explicitly created by the java code and after use it is automatically destroyed by garbage collection for memory