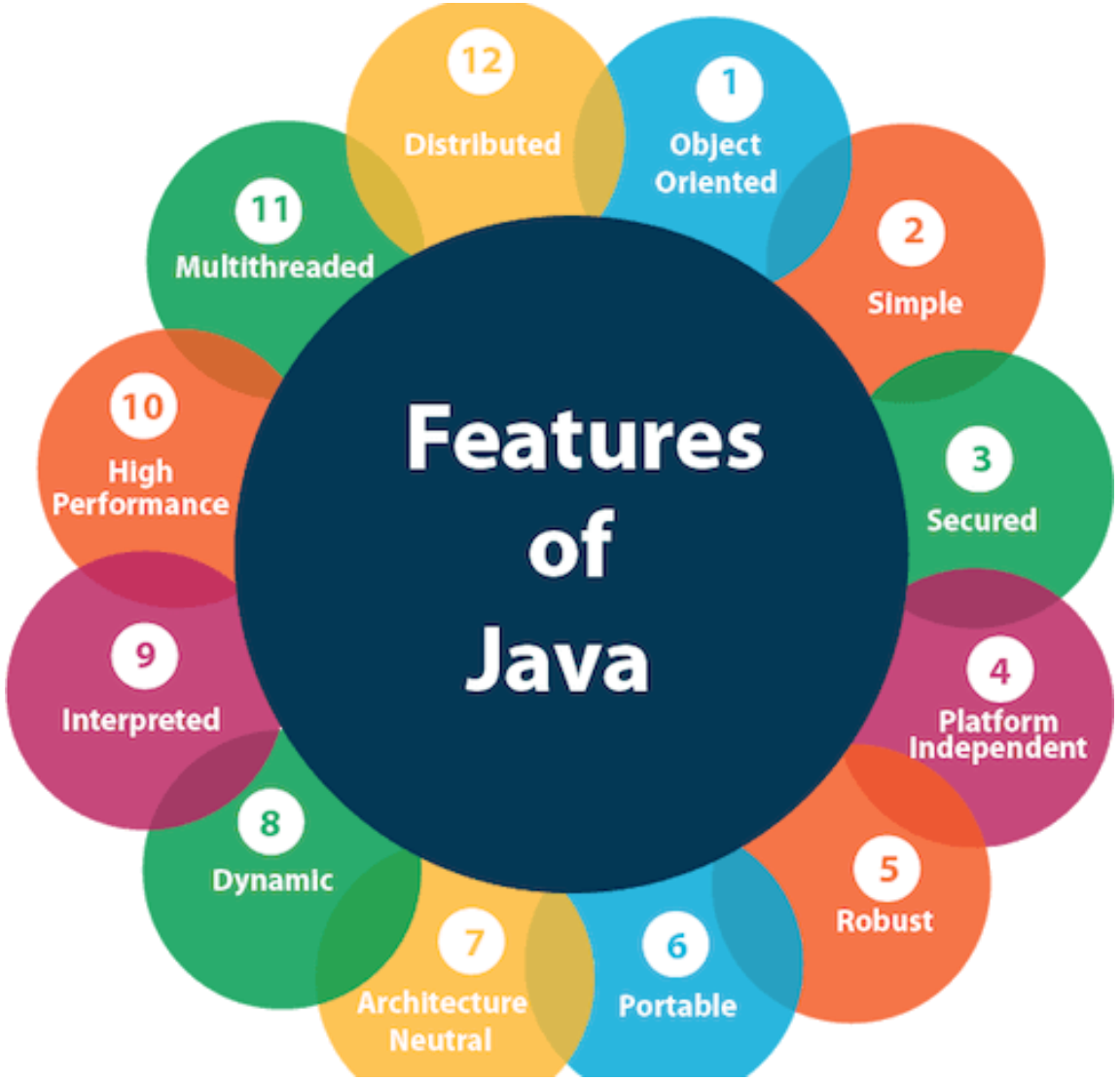
Department of Computer Science
Amrita School of Engineering
Amritapuri Campus

Chapter 1:
Features of Java

19CSE204
Object Oriented Programming 2-0-3-3

Features of Java

- Simple
- Object-Oriented
- Portable
- Platform independent
- Secured
- Robust
- Architecture neutral
- Interpreted
- High Performance
- Multithreaded
- Distributed
- Dynamic



The Java Buzzwords

Image ref:www.javatpoint.com

Java : Simple and Robust

Simple:

- Coding style is very clean and easy to understand
- **Doesn't** use complex and difficult features of other languages like C and C++, such as
  - Concept of Explicit Pointers
  - Storage classes
  - Preprocessors and header files
  - Multiple Inheritance
  - Operator Overloading
  - Goto Statements
- Automatic Garbage Collection

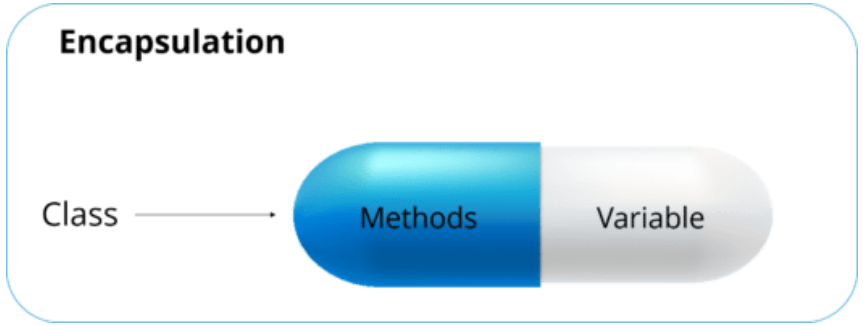
Robust

- It uses strong memory management.
- 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.
- There are exception handling and the type checking mechanism in Java.

Java is Object Oriented

Data is more important than function.

- A mechanism where you bind your data and code together as a single unit. : Encapsulation
- Java is based on declaring classes, creating objects from them and interacting between these objects.
- **Benefits of Object Oriented Programming?**
  - Improved productivity during software development
  - Improved software maintainability
  - Faster development sprints
  - Lower cost of development
  - Higher quality software

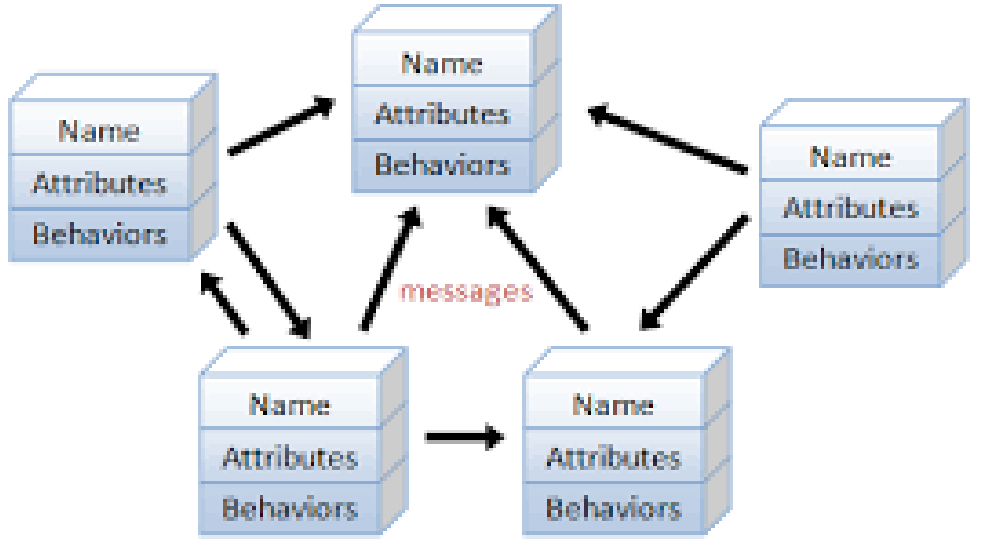



```
public class Car{
    private string _color;
    private string _model;
    private string _makeYear;
    private string _fuelType;

    public void Start(){
        ..
    }

    public void Stop(){
        ..
    }

    public void Accelerate(){
        ..
}
```

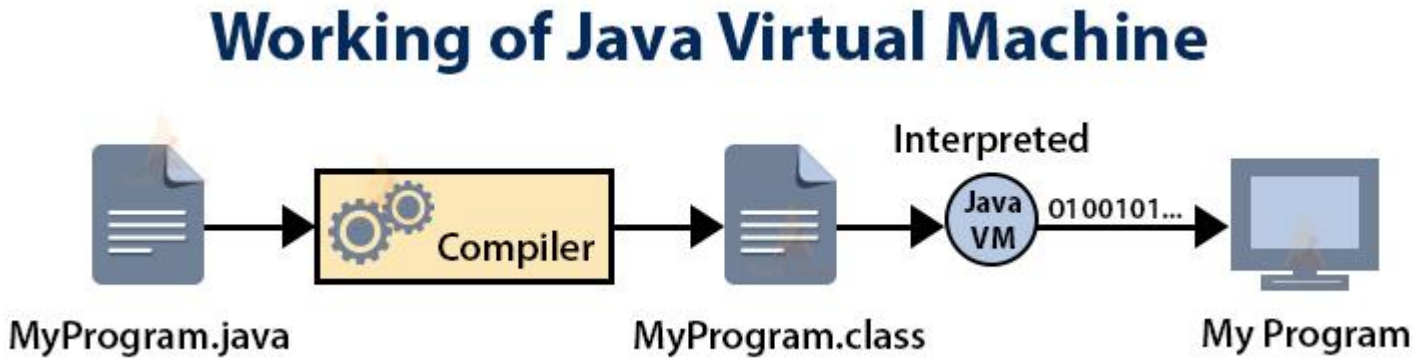


An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

Java: Compiled and Interpreted

Java integrates the power of Compiled Languages with the flexibility of Interpreted Languages.

- **Java compiler (javac)** compiles the java source code into the bytecode.
- **Java Virtual Machine (JVM)** then executes this bytecode which is executable on many operating systems and is portable.
- **Java bytecode** is an intermediate, compact, way of representing a series of operations



This JVM process is what allows compiled Java to be portable to any computer

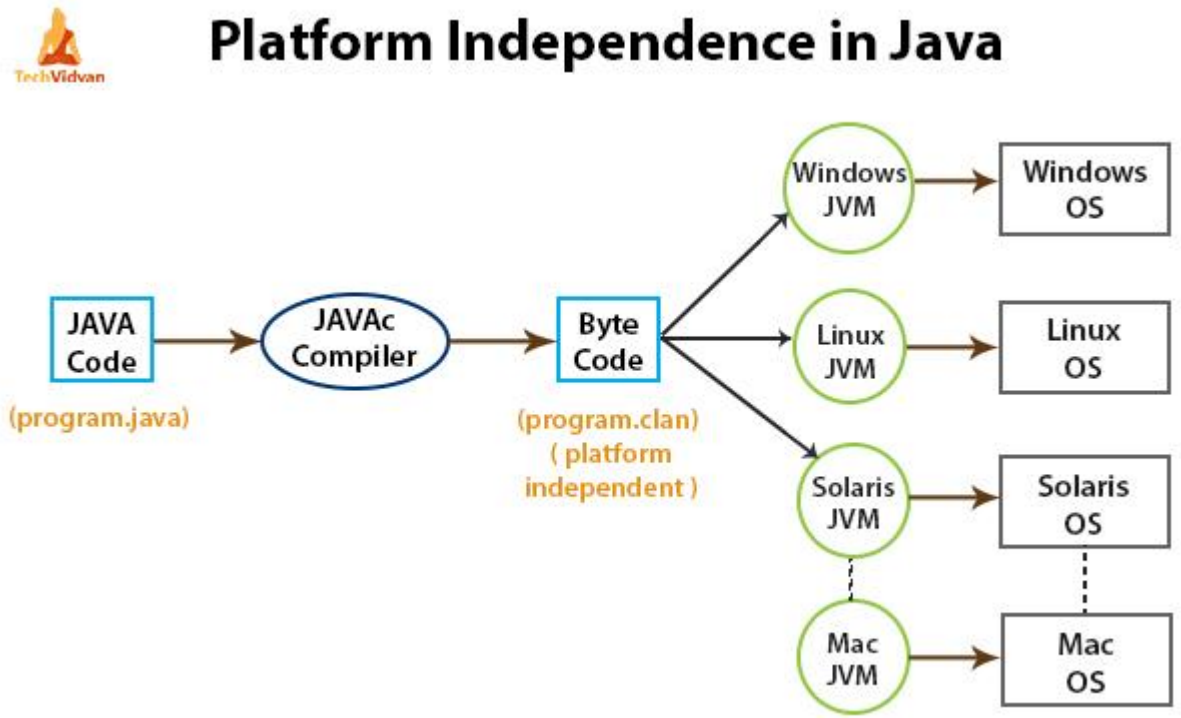
Java: Platform independence

"Write Once, run anywhere" (WORA)- develop applications on one environment (OS) and run on any other environment without doing any modification in the code.

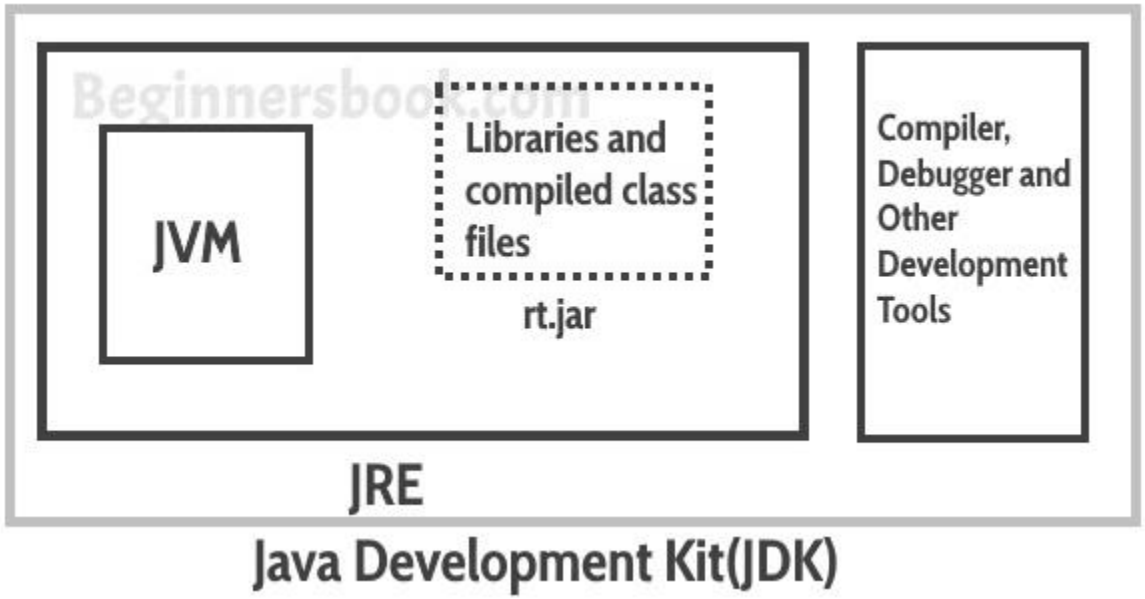
Java Platform:

- 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. Bytecode is a platform-independent code because it can be run on multiple platforms,



Java Development Kit (JDK)



JDK (Java Development Kit) is a software development environment used for making applets and Java applications. DK helps them to code and run Java programs

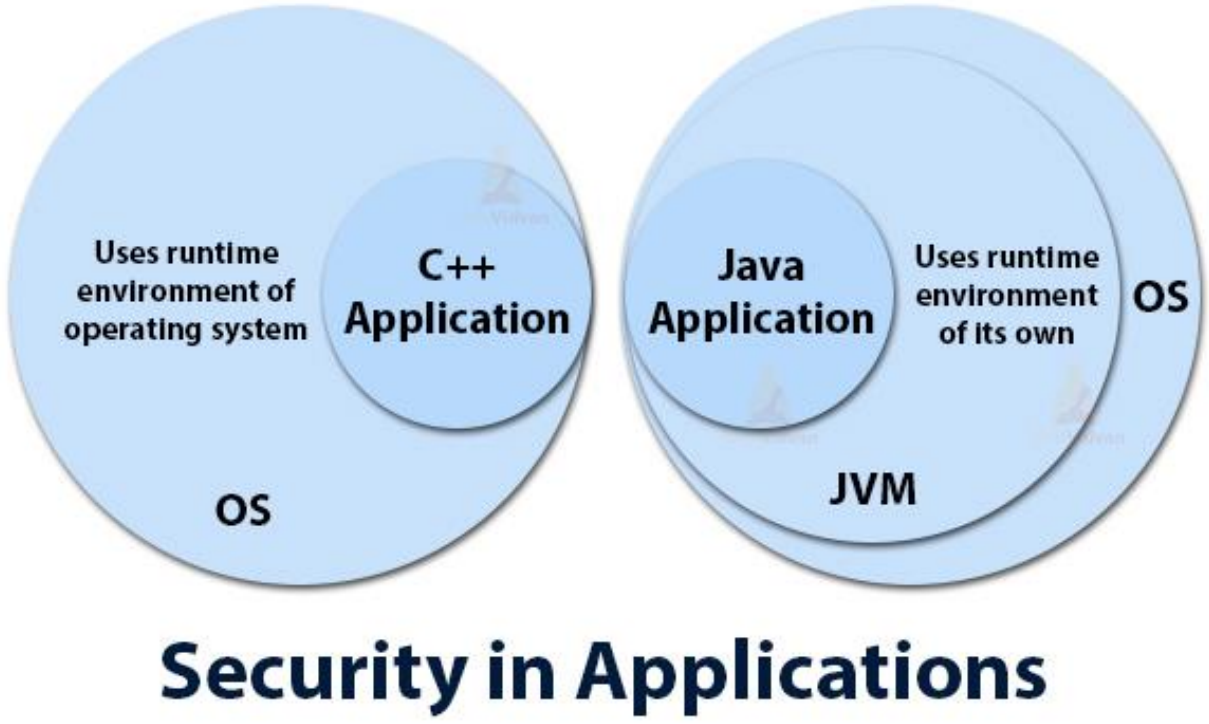
JRE : Java Runtime Environment
Is a piece of a software which is designed to run other software. It contains the class libraries, loader class, and JVM the minimum requirement for executing a java application.

JVM (Java Virtual Machine),
analyzes the bytecode, interprets the code, and executes it.

Java: Secure

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

- **No explicit pointer**
- **virtual machine sandbox**: A separate environment that allows users to execute their applications without affecting the underlying system
- **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.
- **Bytecode Verifier:** It checks the code fragments for illegal code that can violate access right to objects.
- **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.



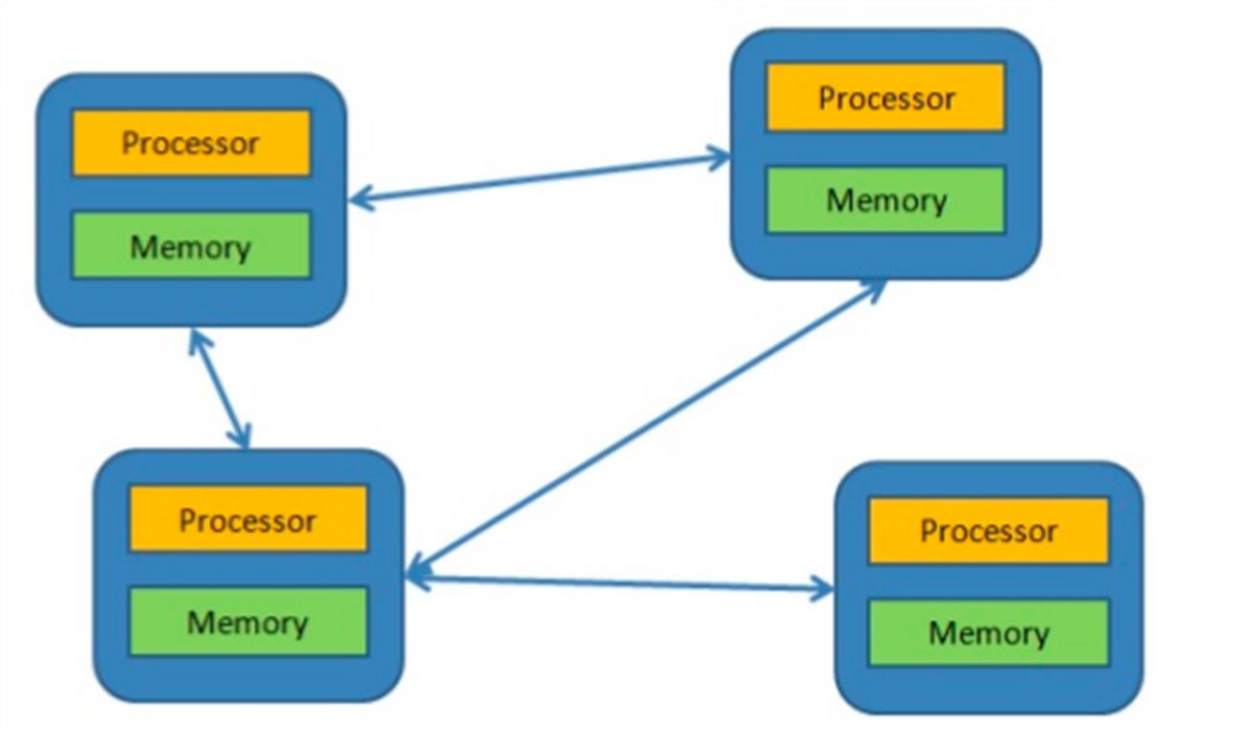
Java provides these securities by default

Java: Distributed

Java is designed to make distributed computing Applications

- Java is designed to make distributed computing easy with networking capabilities that is inherently integrated
- In Java, we can **split a program into many parts** and store these parts on different computers. A Java programmer sitting on a machine can access another program running on the other machine.

- Java helps us to achieve this by providing the concept of **RMI (Remote Method Invocation)** and **EJB (Enterprise JavaBeans)**.
- Java comes with an extensive library of classes for interacting, using **TCP/IP protocols such as HTTP and FTP,** which makes creating network connections much easier than in C/C++.
- It also enables multiple **programmers at many locations to work together** on a single project.



Writing network programs in java is like sending and receiving files to and from file

Java: Portable , Architectural neutral

Architectural Neutral

- Program written on one platform or OS is **independent** of other platforms or environments
- It can run on any other Operating System without recompiling them.
- In other words, it is based on the **'Write-once-run-anywhere' (WORA)** or 'Write-once-run-everywhere' (WORE) approach.
- Byte-code is not dependent on any machine architecture and Java Virtual Machine (JVM) can easily translate bytecode into a machine-specific code

Portable

- In C/C++, the source code may run slightly differently on different hardware platforms,
- Java simplifies it. We can run Java bytecode on **any hardware** that has a compliant JVM which can convert the bytecode according to that particular machine.
- The portability comes from **architecture-neutrality**.

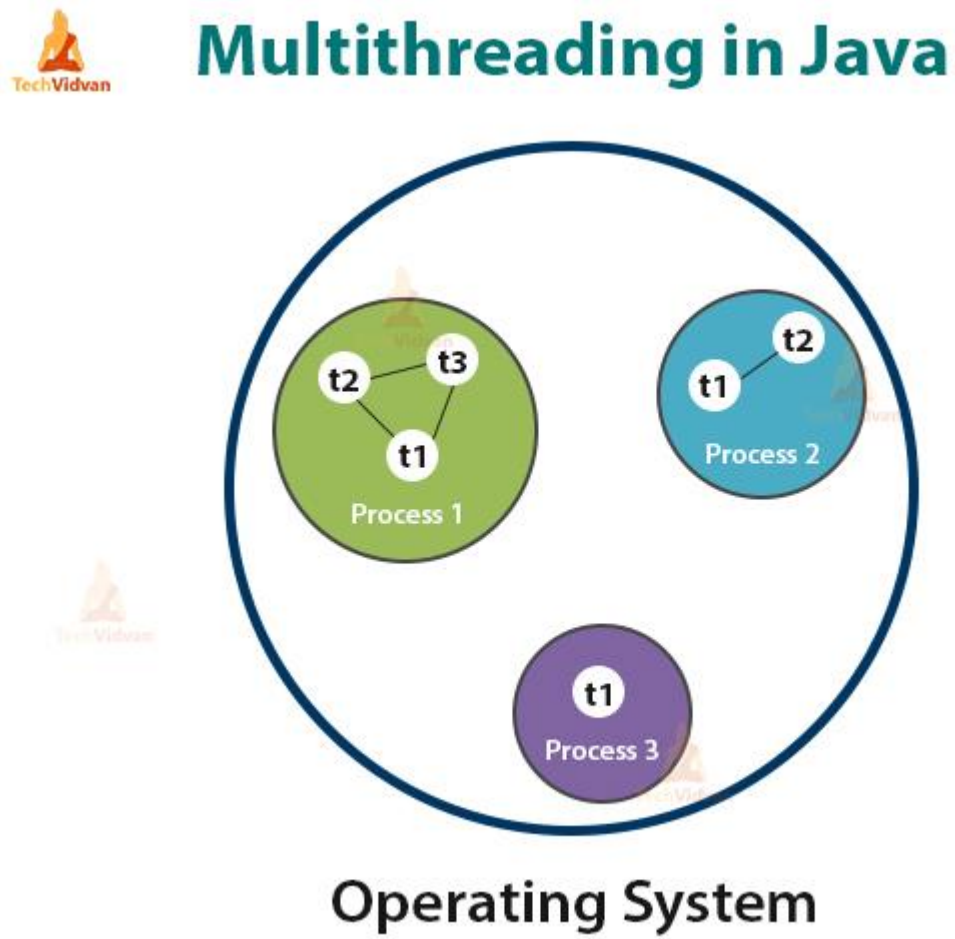
Java: Multithreading

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU

- **A thread is an independent path of execution** within a program, executing concurrently.
- **Multithreaded** means handling multiple tasks simultaneously or executing multiple portions (functions) of the same program in parallel.
- The **code of java is divided into smaller parts** and Java executes them in a **sequential** and **timely** manner

Advantages of Multithreading

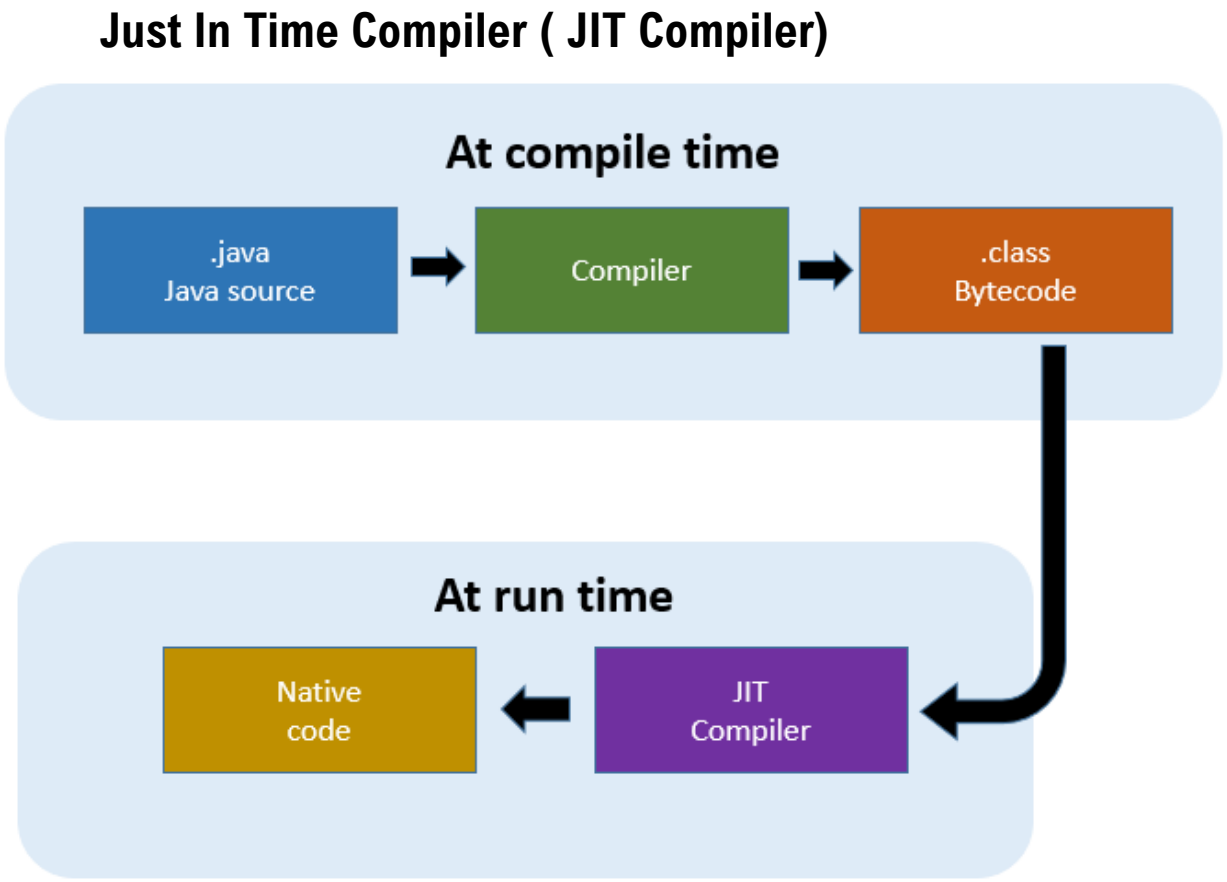
•the maximum utilization of resources is possible.
•It doesn't occupy memory for each thread. It shares a common memory area.
•There is no need to wait for the application to finish one task before beginning another one.
•There is a decreased cost of maintenance. Also, It is time-saving.
•It improves the performance of complex applications.



Java : High Performance

The performance of Java is impressive for an interpreted language because of its intermediate bytecode.

- Java provides high performance with the use of **"JIT – Just In Time compiler"**, in which the compiler compiles the code on-demand basis, that is, it compiles only that method which is being called. This saves time and makes it more efficient.
- **Java architecture** is also designed in such a way that it reduces overheads during runtime.
- The inclusion of **multithreading** enhances the overall execution speed of Java programs.
- Bytecodes generated by the Java compiler are **highly optimized**, so Java Virtual Machine can execute them much faster.



NB:Java is faster than other traditional interpreted programming languages because of Java bytecode It is still a little bit slower than a compiled language (e.g., C++) as it is interpreted language.

Java: Dynamic and Extensible

Java is dynamic and extensible due to OOPS, dynamically linking new libraries, supports functions written in other languages

- **With OOPS** we can add classes and add new methods to classes, creating new classes through subclasses. This makes it easier for us to **expand** our own classes and even **modify** them.

- Java gives the **facility of dynamically linking new class libraries**, methods, and objects. It is highly dynamic as it can adapt to its evolving environment.

- Java even **supports functions written in other languages** such as C and C++ to be written in Java programs. These functions are called “native methods”. These methods are dynamically linked at runtime.

Next Session- OOPS

In the next sessions we get into Object Oriented Programming Concepts

Namah Shivaya!