GLS UNIVERSITY

SEM – IV

0301301 - CORE JAVA

Dr. Disha Shah
Prof. Vidhi Thakkar

Unit – I
Introduction to Java

Index

- Object Oriented Concepts
- Basics of Java

Object Oriented Concepts

- Introduction
- Principles of OOP
- Java Features
- Java Program Structure
- Java Architecture
- Introduction to Java framework
- Real Time Java Applications

INTRODUCTION

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 (specially Android apps)
- Artificial Intelligence
- Big Data
- Serverless Architecture
- Spring Framework
- Cloud based Models
- Remote Access Solutions

Principles of OOP

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

Principles of OOP Language

- **Classes:** Collection of similar objects
- **Objects:** Instance of class
- **Abstraction:** Hiding the implementation details and showing only functionality to the user.
- **Inheritance:** Adopt the characteristics of one class into another class
- **Encapsulation:** Wrapping of data in single unit
- **Polymorphism:**Perform a single action by different ways

What is JAVA ?

- Java is a popular programming language, created in 1995.
- It is owned by Oracle.
- Java technology is used by more than 6 million developers and runs on more than 5.5 billion devices.
- Java is a high level, robust, secured and object-oriented programming language.
- “Java is a programming language and a platform.”
- Java Applications are called **WORA** (Write Once Run Anywhere)
- **Platform**: Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

Common devices that run Java

- Airplane Systems
- ATMs
- BlackBerry Smartphones
- Blu-ray Disc Players
- Cable Boxes
- Cell Phones
- Computers
- Credit Cards
- CT Scanners
- Government IDs
- Home Security Systems
- Kindle E-Readers
- Livescribe Smartpens
- Lottery Systems
- MRIs
- On-Board Computer Systems
- Parking Meters
- PlayStation Consoles
- Printers
- Public Transportation Passes
- Robots
- Routers
- Smart Grid Meters
- TVs
- Vehicle Diagnostic Systems
- VoIP Phones

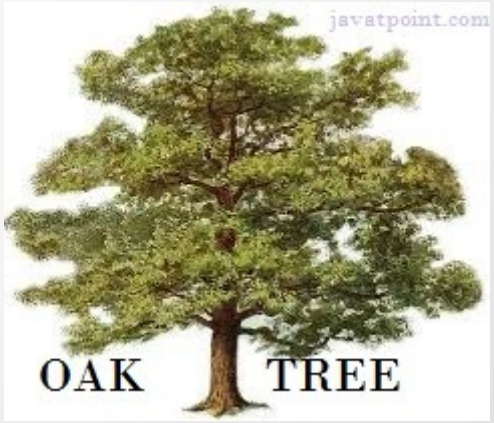
History of Java

- James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991.
- The small team of sun engineers called Green Team.
- Originally designed for small, embedded systems in electronic appliances like set-top boxes.
- Firstly, it was called "Greentalk" by James Gosling and file extension was .gt.
- After that, it was called Oak and was developed as a part of the Green project.

History of Java

Why Sun choosed "Oak" name?

- Why Oak? Oak is a symbol of strength and choosen as a national tree of many countries like U.S.A., France, Germany, Romania etc.
- In 1995, Oak was renamed as "Java" because it was already a trademark by Oak Technologies.



History of Java

Why sun choosed "Java" name?

- 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.
- Java is an island of Indonesia where first coffee was produced (called java coffee).
- JDK 1.0 released in(January 23, 1996).

History of Java

- Java SE 17 (September 2021)
- Java SE 18 (March 2022)

C v/s C++ v/s JAVA

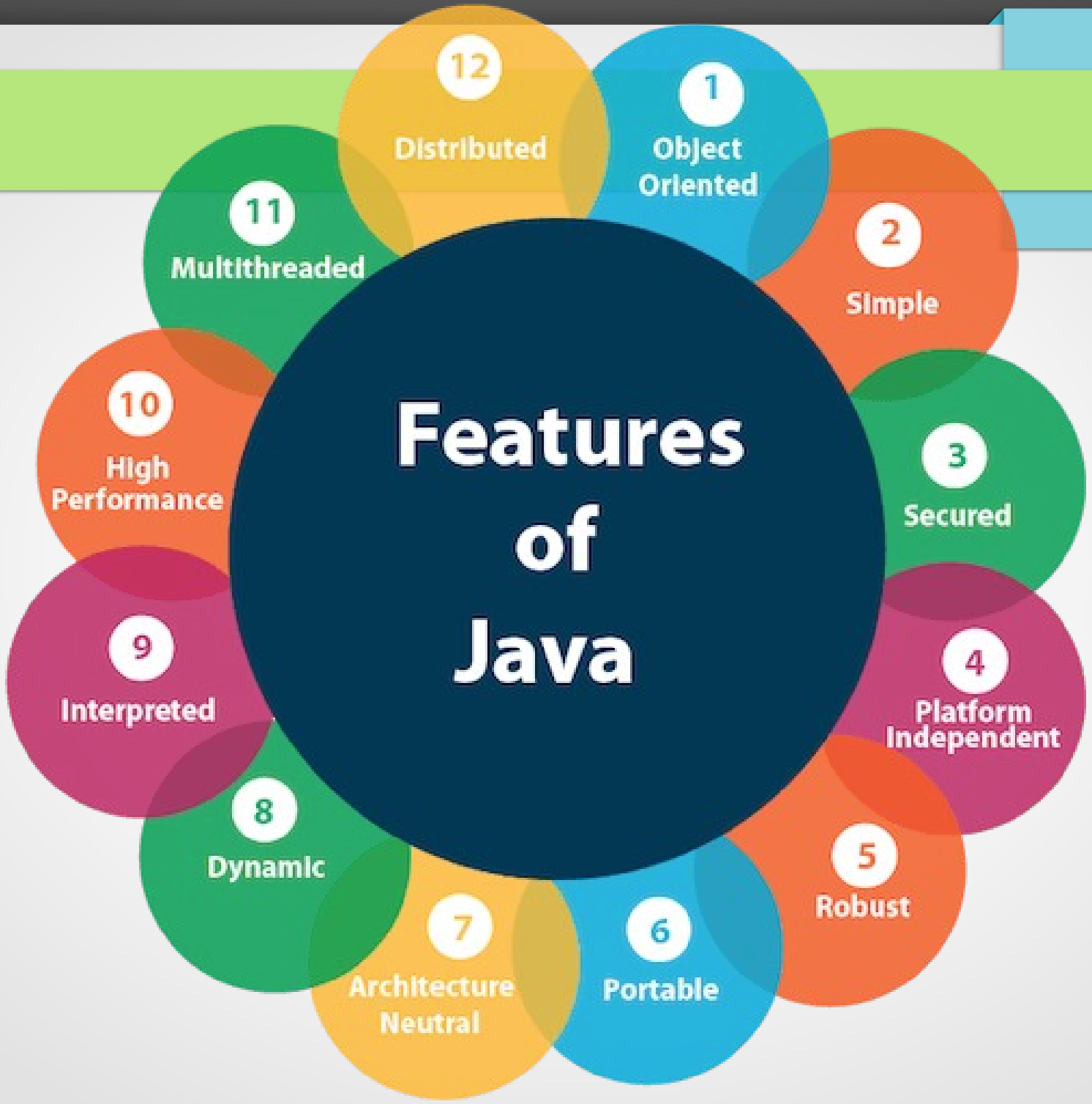
Metrics	C	C++	Java
Programming Paradigm	Procedural language	Object-Oriented Programming (OOP)	Pure Object Oriented Oriented
Origin	Based on Assembly language	Based on C language	Based on C and C++
Developer	Dennis Ritchie in 1972	Bjarne Stroustrup in 1979	James Gosling in 1991
Translator	Compiler only	Compiler only	Interpreted language (**Compiler + interpreter**)
Platform Dependency	Platform Dependent	Platform Dependent	Platform Independent
Code execution	Direct	Direct	Executed by JVM (**Java Virtual Machine**)
Approach	Top-down approach	Bottom-up approach	Bottom-up approach

C v/s C++ v/s JAVA

Metrics	C	C++	Java
File generation	.exe files	.exe files	.class files
Pre-processor directives	Support header files (#include, #define)	Supported (#header, #define)	Use Packages (import)
keywords	Support 32 keywords	Supports 63 keywords	50 defined keywords
Datatypes (union, structure)	Supported	Supported	Not supported
Inheritance	No inheritance	Supported	Supported except Multiple inheritance
Overloading	No overloading	Support Function overloading (Polymorphism)	Operator overloading is not supported
Pointers	Supported	Supported	Not supported

C v/s C++ v/s JAVA

Metrics	C	C++	Java
Allocation	Use malloc, calloc	Use new, delete	Garbage collector
Exception Handling	Not supported	Supported	Supported
Templates	Not supported	Supported	Not supported
Destructors	No constructor neither destructor	Supported	Not supported
Multithreading/ Interfaces	Not supported	Not supported	Supported
Database connectivity	Not supported	Not supported	Supported
Storage Classes	Supported (auto, extern)	Supported (auto, extern)	Not supported


Features of Java
1 Object Oriented
2 Simple
3 Secured
4 Platform Independent
5 Robust
6 Portable
7 Architecture Neutral
8 Dynamic
9 Interpreted
10 High Performance
11 Multithreaded
12 Distributed

Java Features

- **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

Java Features

- **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable.
- **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- **Architecture-neutral -** Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

Java Features

- **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed** − Java is designed for the distributed environment of the internet.
- **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

Java Version

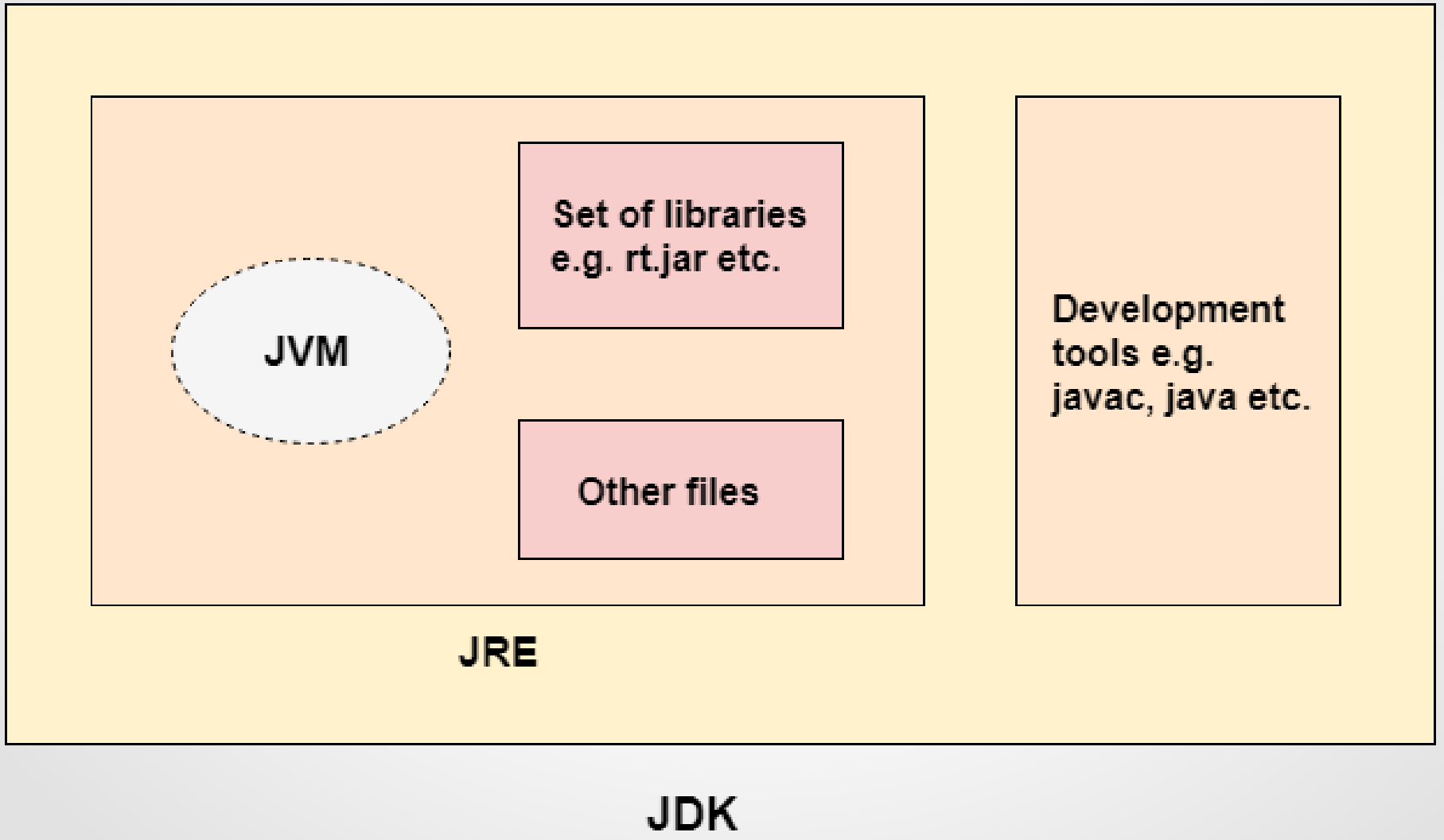
- JDK Alpha and Beta (1995)
- JDK 1.0 (23rd Jan 1996)
- JDK 1.1 (19th Feb 1997)
- J2SE 1.2 (8th Dec 1998)
- J2SE 1.3 (8th May 2000)
- J2SE 1.4 (6th Feb 2002)
- J2SE 5.0 (30th Sep 2004)
- Java SE 6 (11th Dec 2006)
- Java SE 7 (28th July 2011)
- Java SE 8 (18th Mar 2014)
- Java SE 9 (21st Sep 2017)
- Java SE 10 (20th Mar 2018)
- Java SE 11 (September 2018)
- Java SE 12 (March 2019)
- Java SE 13 (September 2019)
- Java SE 14 (Mar 2020)
- Java SE 15 (September 2020)
- Java SE 16 (Mar 2021)
- Java SE 17 (September 2021)
- Java SE 18 (to be released by March 2022)

Why use Java?

- Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- It is one of the most popular programming language in the world
- It is easy to learn and simple to use
- It is open-source and free
- It is secure, fast and powerful
- It has a huge community support (tens of millions of developers)
- Java is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
- As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa.

Common Terminologies used in Java

- Java Virtual Machine (JVM)
- Bytecode
- Java Development Kit (JDK)
- Java Runtime Environment (JRE)
- Garbage Collector
- ClassPath

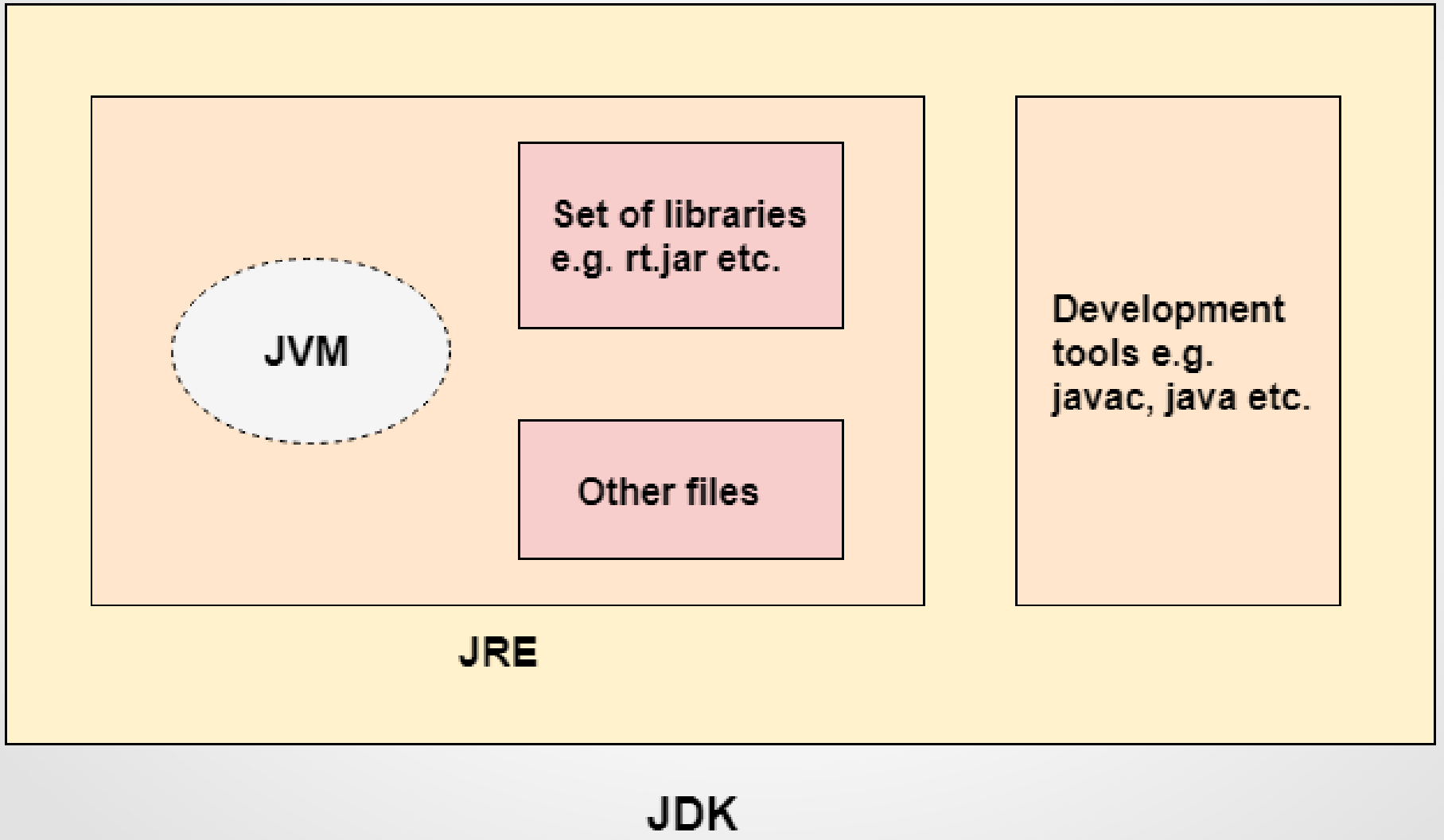

JVM
Set of libraries
e.g. rt.jar etc.
Other files
JRE
Development
tools e.g.
javac, java etc.
JDK

Java Virtual Machine(JVM):

- This is generally referred to as JVM.
- There are three execution phases of a program. They are writting a program, compiling it and running the program.
- Writing a program is done by a java programmer.
- The compilation is done by the JAVAC compiler which is a primary Java compiler included in the Java development kit (JDK). It takes Java program as input and generates bytecode as output.
- In the Running phase of a program, JVM executes the bytecode generated by the compiler.
- The function of Java Virtual Machine is to execute the bytecode produced by the compiler.
- Every Operating System has a different JVM but the output they produce after the execution of bytecode is the same across all the operating systems. Thus Java is known as a platform-independent language.

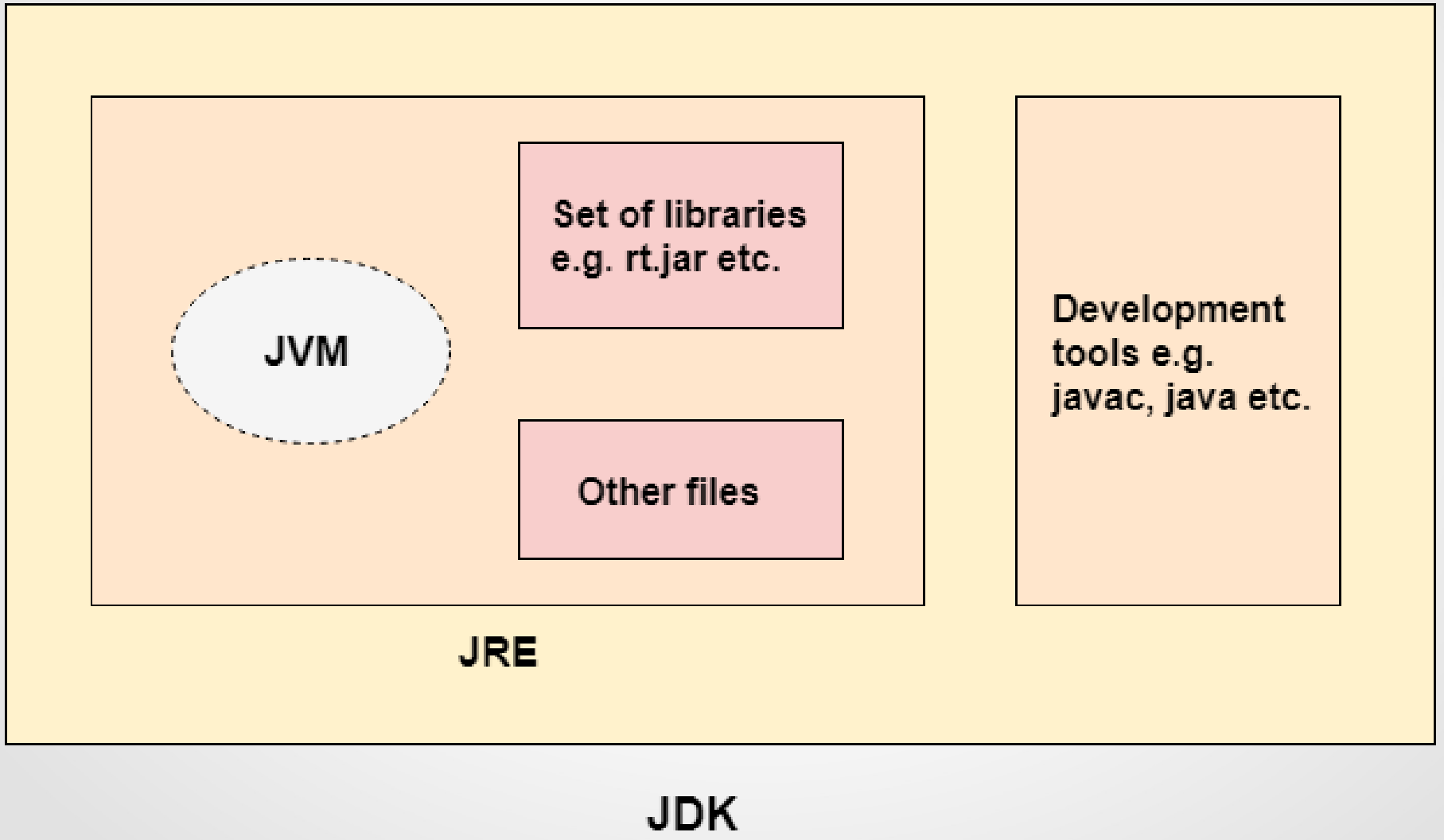
Bytecode

- Bytecode is generated in the Development process:
- The Javac compiler of JDK compiles the java source code into bytecode so that it can be executed by JVM.
- It is saved as .class file by the compiler.
- To view the bytecode, a disassembler like javap can be used.
- The javap tool is used to get the information of any class or interface.
- The javap command (also known as the Java Disassembler) disassembles one or more class files.


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

Java Runtime Environment (JRE):

- JDK includes JRE.
- JRE installation on our computers allows the java program to run, however, we cannot compile it.
- JRE includes a browser, JVM, applet supports, and plugins.
- For running the java program, a computer needs JRE.


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

Java Development Kit(JDK):

- It is a complete Java development kit that includes everything including compiler, Java Runtime Environment (JRE), java debuggers, java docs, etc.
- For the program to execute in java, we need to install JDK on our computer in order to create, compile and run the java program.
- The JDK is a development environment for building applications, applets, and components using the Java programming language.
- The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

How to install JDK?

- https://www.oracle.com/java/technologies/downloads/
  - Use the above link to download and install

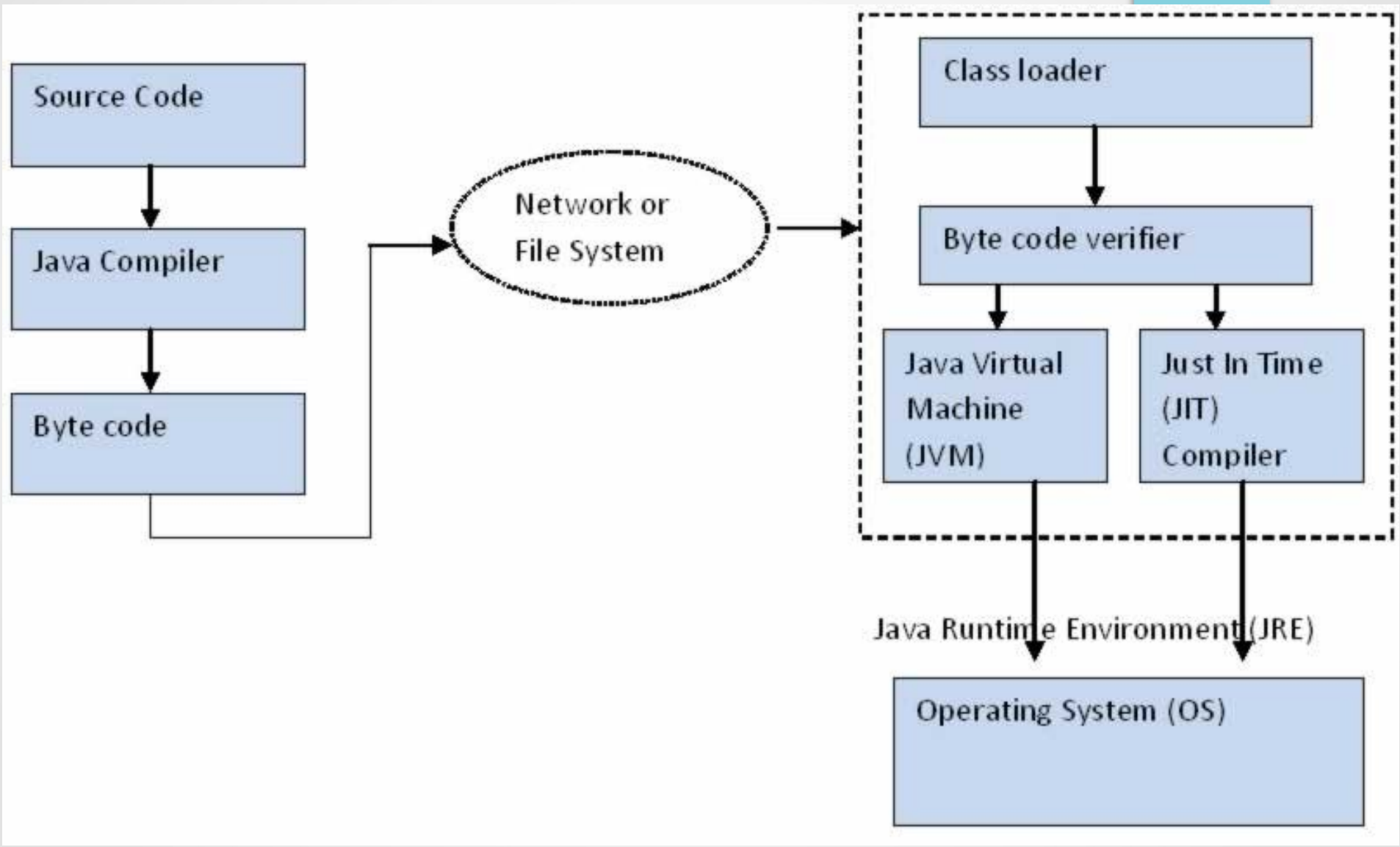
Garbage Collector:

- In Java, programmers can’t delete the objects.
- To delete or recollect that memory JVM has a program called Garbage Collector.
- Garbage Collectors can recollect the of objects that are not referenced.

ClassPath:

- The classpath is the file path where the java runtime and Java compiler look for .class files to load.
- By default, JDK provides many libraries.
- If you want to include external libraries they should be added to the classpath.
- Classpath is a parameter in the Java Virtual Machine or the Java compiler that specifies the location of user-defined classes and packages. The parameter may be set either on the command-line, or through an environment variable

Java Architecture



Java Architecture

- In Java, there is a process of compilation and interpretation.
- The code written in Java, is converted into byte codes which is done by the **Java Compiler**.
- The **Just-In-Time (JIT) compiler** is a an essential part of the JRE i.e. **Java Runtime Environment,** that is responsible for performance optimization of java based applications at run time.
- The byte codes, then are converted into machine code by the **JVM**.
- The Machine code is executed directly by the machine.

Java Architecture

- **Java Virtual Machine (JVM)** is a engine that provides runtime environment to drive the Java Code or applications.
- It converts Java bytecode into machines language. JVM is a part of Java Run Environment (JRE).
- JVM is responsible for allocating memory space.

Java Architecture

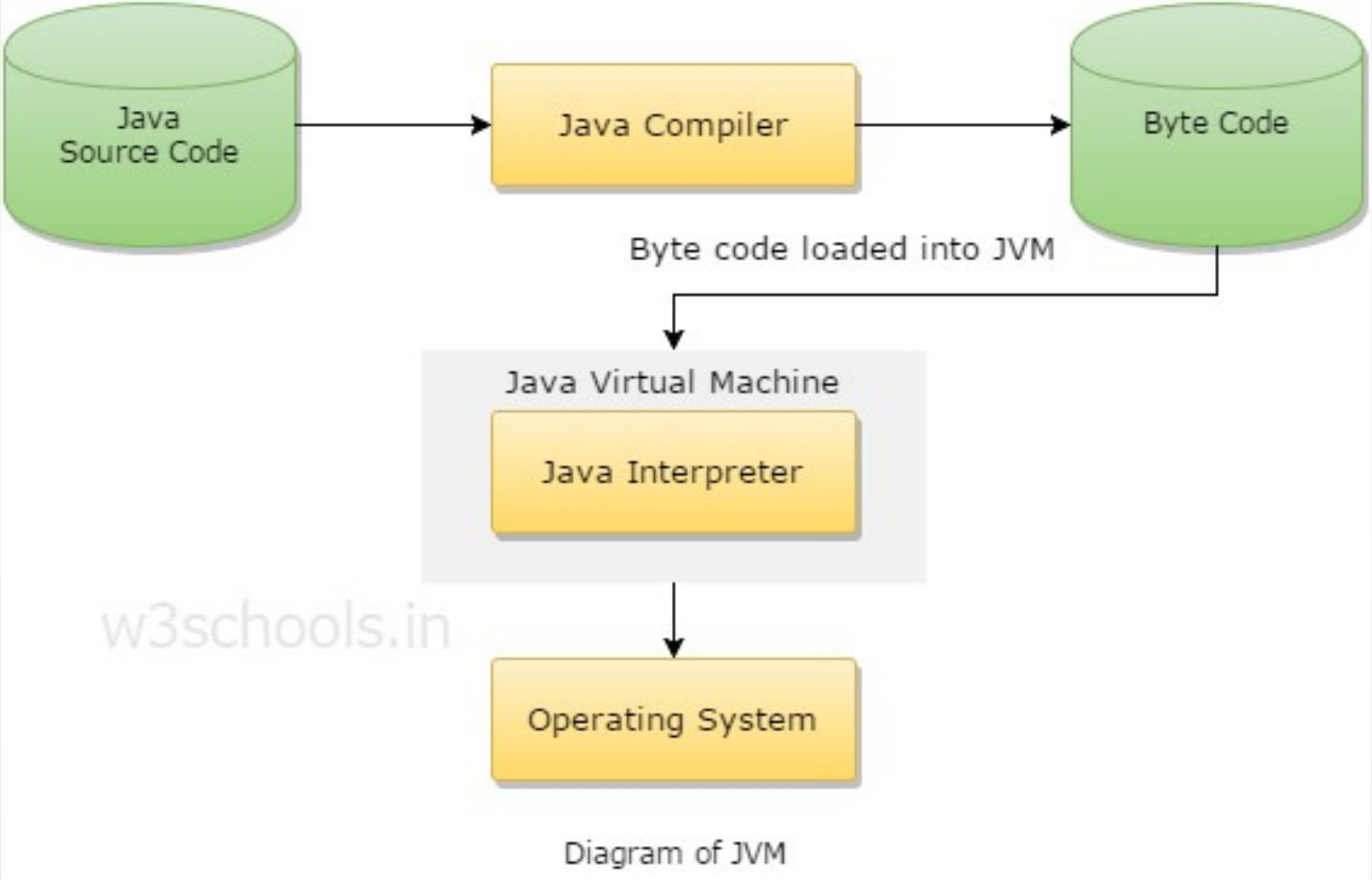


Diagram of JVM

Java Architecture

- There are three main components of Java language:
  - JVM, JRE, and JDK.
- Java Virtual Machine, Java Runtime Environment and Java Development Kit respectively.

Java Programming Structure

```
class Simple
{
    public static void main(String args[])
    {
    System.out.println("Hello Java");
     }
}
```

To compile: javac Simple.java

To execute: java Simple

Java Programming Structure

- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility, it means it is visible to all.
- **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create object to invoke the main method. So it saves memory.
- **void** is the return type of the method, it means it doesn't return any value.
- **main** represents startup of the program.
- **String[] args** is used for command line argument. We will learn it later.
- **System.out.println()** is used print statement. We will learn about the internal working of System.out.println statement later.

Introduction to Java framework

- Frameworks are large bodies of pre-written code to which you add your own code in order to solve a problem.
- You make use of a framework by calling its methods, inheritance, and supplying callbacks, listeners, or other implementations of the patterns.
- A framework will often dictate the structure of an application.
- Some frameworks even supply so much code that you have to do very little to write your application.

Top Java Frameworks used

- Spring
- Hibernate
- JavaServer Faces [JSF]
- Struts
- Google web toolkit [GWT]
- Grails
- Vaadin
- Blade
- Dropwizard
- Play

Real Time Java Applications

1) Android Apps
2) Server Apps at Financial Services Industry
3) Java Web applications
4) Software Tools
5) Trading Application
6) J2ME Apps
7) Embedded Space
8) Big Data technologies
9) High Frequency Trading Space
10) Scientific Applications