Assignment - Fundamental

1. B1. Features of Java. (or) Java buzzwords.

🡪The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

1. Simple
2. Object-oriented
3. Distributed
4. Interpreted
5. Robust
6. Secure
7. Architecture neutral
8. Portable
9. High performance
10. Multithreaded
11. Dynamic

1. Simple

* Java was designed to be easy for a professional programmer to learn and use effectively.
* It’s simple and easy to learn if you already know the basic concepts of Object Oriented Programming.
* Best of all, if you are an experienced C++ programmer, moving to Java will require very little effort. Because Java inherits the C/C++ syntax and many of the object-oriented features of C++, most programmers have little trouble learning Java.
* Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.

2. Object Oriented

* Java is true object-oriented language.
* Almost “Everything is an Object” paradigm. All program code and data reside within objects and classes.
* The object model in Java is simple and easy to extend.
* Java comes with an extensive set of classes, arranged in packages that can be used in our programs through inheritance.
* Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

Basic concepts of OOPs are:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

3. Distributed

* Java is designed for distributed environment of the Internet. Its used for creating applications on networks.
* Java applications can access remote objects on the Internet as easily as they can do in the local system.
* Java enables multiple programmers at multiple remote locations to collaborate and work together on a single project.
* Java is designed for the distributed environment of the Internet because it handles TCP/IP protocols.

4. Compiled and Interpreted

* Usually, a computer language is either compiled or Interpreted. Java combines both this approach and makes it a two-stage system.
* Compiled: Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java Bytecode.
* Interpreted: Bytecode is then interpreted, which generates machine code that can be directly executed by the machine that provides a Java Virtual machine.

5. Robust

* It provides many features that make the program execute reliably in a variety of environments.
* Java is a strictly typed language. It checks code both at compile time and runtime.
* Java takes care of all memory management problems with garbage collection.
* Java, with the help of an exception handling, captures all types of serious errors and eliminates any risk of crashing the system.

6. Secure

* Java provides a “firewall” between a networked application and your computer.
* When a Java Compatible Web browser is used, downloading can be done safely without fear of viral infection or malicious intent.
* Java achieves this protection by confining a Java program to the Java execution environment and not allowing it to access other parts of the computer.

7. Architecture Neutral

* Java language and Java Virtual Machine helped in achieving the goal of “write once; run anywhere, any time, forever.”
* Changes and upgrades in operating systems, processors and system resources will not force any changes in Java Programs.

8. Portable

* Java Provides a way to download programs dynamically to all the various types of platforms connected to the Internet.
* It helps in generating Portable executable code.

9. High Performance

* Java performance is high because of the use of bytecode.
* The bytecode was used so that it was easily translated into native machine code.

10. Multithreaded

* Multithreaded Programs handled multiple tasks simultaneously, which was helpful in creating interactive, networked programs.
* Java run-time system comes with tools that support multiprocess synchronization used to construct smoothly interactive systems.

11. Dynamic

* Java is capable of linking in new class libraries, methods, and objects.
* Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at runtime. This makes it possible to dynamically link code in a safe and expedient manner.

1. B2. Difference between JDK, JRE and JVM.

## JVM

JVM is the abbreviation for Java virtual machine which is basically specification that provides a runtime environment in which Java byte code can be executed i.e it is something which is abstract and its implementation is independent to choose the algorithm and has been provided by Sun and other companies. It is JVM which is responsible for converting Byte code to the machine specific code. It can also run those programs which are written in other languages and compiled to Java bytecode.The JVM performs the mentioned tasks: Loads code, Verifies code, Executes code, Provides runtime environment.

## JRE

JRE is Java runtime environment which is the implementation of JVM i.e the specifications which are defined in JVM are implemented and creates corresponding environment for the execution of code.JRE comprises mainly java binaries and other classes to execute the program alike of JVM it physically exists. Along with Java binaries JRE also consist of various technologies of deployment, user interfaces to interact with code executed, some base libraries for different functionalities and language and util based libraries.

## JDK

JDK is abbreviation for Java Development Kit which includes all the tools, executable and binaries required to compile, debug and execute a Java Program.JDK is platform dependent i.e there is separate installers for Windows, Mac, and Unix systems.JDK includes both JVM and JRE and is entirely responsible for code execution. It is the version of JDK which represent version of Java.

| Sr. No. | Key | JDK | JRE | JVM |
| --- | --- | --- | --- | --- |
| 1 | Definition | JDK (Java Development Kit) is a software development kit to develop applications in Java. In addition to JRE, JDK also contains number of development tools (compilers, JavaDoc, Java Debugger etc.). | JRE (Java Runtime Environment) is the implementation of JVM and is defined as a software package that provides Java class libraries, along with Java Virtual Machine (JVM), and other components to run applications written in Java programming. | JVM (Java Virtual Machine) is an abstract machine that is platform-dependent and has three notions as a specification, a document that describes requirement of JVM implementation, implementation, a computer program that meets JVM requirements, and instance, an implementation that executes Java byte code provides a runtime environment for executing Java byte code. |
| 2 | Prime functionality | JDK is primarily used for code execution and has prime functionality of development. | On other hand JRE is majorly responsible for creating environment for code execution. | JVM on other hand specifies all the implementations and responsible to provide these implementations to JRE. |
| 3 | Platform Independence | JDK is platform dependent i.e for different platforms different JDK required. | Like of JDK JRE is also platform dependent. | JVM is platform independent. |
| 4 | Tools | As JDK is responsible for prime development so it contains tools for developing, debugging and monitoring java application. | On other hand JRE does not contain tools such as compiler or debugger etc. Rather it contains class libraries and other supporting files that JVM requires to run the program. | JVM does not include software development tools. |
| 5 | Implementation | JDK = Java Runtime Environment (JRE) + Development tools | JRE = Java Virtual Machine (JVM) + Libraries to run the application | JVM = Only Runtime environment for executing the Java byte code. |

1. B3. Java is Platform independent.?

* This bytecode is sent to Java virtual machine (JVM) which resides in the RAM of any operating system. JVM recognizes the platform it is on and converts the bytecodes into native machine code. Hence java is called platform independent language.

1. B4. Three flavors of Java

## The Java Programming Language Platforms

There are three platforms of the Java programming language:

* Java Platform, Standard Edition (Java SE)
* Java Platform, Enterprise Edition (Java EE)
* Java Platform, Micro Edition (Java ME)

All Java platforms consist of a Java Virtual Machine (VM) and an application programming interface (API). The Java Virtual Machine is a program, for a particular hardware and software platform, that runs Java applications. An API is a collection of software components that you can use to create other software components or applications. Each Java platform provides a virtual machine and an API, and this allows applications written for that platform to run on any compatible system with all the advantages of the Java programming language: platform-independence, power, stability, ease-of-development, and security.

### Java SE

When most people think of the Java programming language, they think of the Java SE API. Java SE's API provides the core functionality of the Java programming language. It defines everything from the basic types and objects of the Java programming language to high-level classes that are used for networking, security, database access, graphical user interface (GUI) development, and XML parsing.

In addition to the core API, the Java SE platform consists of a virtual machine, development tools, deployment technologies, and other class libraries and toolkits commonly used in Java applications.

### Java EE

The Java EE platform is built on top of the Java SE platform. The Java EE platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications.

### Java ME

The Java ME platform provides an API and a small-footprint virtual machine for running Java programming language applications on small devices, like mobile phones. The API is a subset of the Java SE API, along with special class libraries useful for small device application development. Java ME applications are often clients of Java EE application services.

1. B5. How many types of memory areas are allocated by JVM?

JVM (Java Virtual Machine) is an abstract machine, In other words, it is a program/software which takes Java bytecode and converts the byte code (line by line) into machine understandable code.

[JVM(Java Virtual Machine)](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/) acts as a run-time engine to run Java applications. JVM is the one that actually calls the [main method](https://www.geeksforgeeks.org/understanding-static-in-public-static-void-main-in-java/) present in Java code. JVM is a part of the [JRE(Java Runtime Environment)](https://www.geeksforgeeks.org/differences-jdk-jre-jvm/).

**JVM** perform some particular **types of operations**:

1. Loading of code
2. Verification of code
3. Executing the code
4. It provides a run-time environment to the users

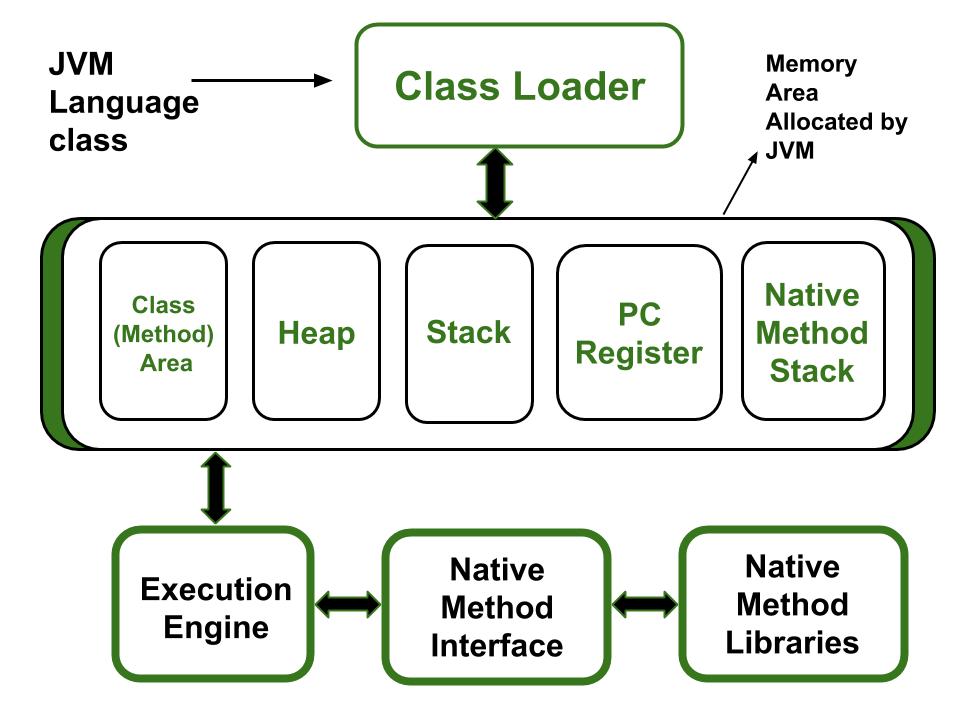
#### ClassLoader

It is a subsystem of JVM which is used to load class files. It is mainly responsible for three activities.

* Loading
* Linking
* Initialization

### Types of Memory Areas Allocated By the JVM:

All these functions take different forms of memory structure. The **memory in the JVM is** **divided into 5 different parts**:



1. Class(Method) Area
2. Heap
3. Stack
4. Program Counter Register
5. Native Method Stack

Let’s see about them in brief:

#### 1. Class (Method) Area

The class method area is the memory block that stores the class code, variable code(static variable, runtime constant), method code, and the constructor of a Java program. (Here method means the function which is written inside the class). It stores class-level data of every class such as the runtime constant pool, field and method data, the code for methods.

#### 2. Heap

The Heap area is the memory block where objects are created or objects are stored. Heap memory allocates memory for class interfaces and arrays (an array is an object). It is used to allocate memory to objects at run time

#### 3. Stack

Each thread has a private JVM stack, created at the same time as the thread. It is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.

Java Stack stores frames and a new frame is created each time at every invocation of the method. A frame is destroyed when its method invocation completes

#### 4. Program Counter Register:

Each JVM thread that carries out the task of a specific method has a program counter register associated with it. The non-native method has a PC that stores the address of the available JVM instruction whereas, in a native method, the value of the program counter is undefined. PC register is capable of storing the return address or a native pointer on some specific platform.

#### 5. Native method Stacks:

Also called C stacks, native method stacks are not written in Java language. This memory is allocated for each thread when it’s created And it can be of a fixed or dynamic nature.

1. B6. What is the latest version of Java?

### What is the latest Java version?

As of September 2021, [Java 17](https://openjdk.java.net/projects/jdk/17/) is the latest released Java version. It is also the next long-term support version (LTS) after Java 11.

1. B7. What is Write Once, Run Anywhere (WORA)?

Java applications are called **WORA (Write Once Run Anywhere)**. This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.

1. B8. Is Java a pure/fully object oriented language?

Java supports property 1, 2, 3, 4 and 6 but fails to support property 5 and 7 given above. Java language is not a Pure Object Oriented Language as it contain these properties:

1. B9. What is bytecode?

Java bytecode is the instruction set for the Java Virtual Machine. It acts similar to an assembler which is an alias representation of a C++ code. As soon as a java program is compiled, java bytecode is generated. In more apt terms, java bytecode is the machine code in the form of a .class file. With the help of java bytecode we achieve platform independence in java.

1. B10. What is Heap space in Java?

Heap space is used for the dynamic memory allocation of Java objects and JRE classes at runtime. New objects are always created in heap space, and the references to these objects are stored in stack memory.

These objects have global access and we can access them from anywhere in the application.

We can break this memory model down into smaller parts, called generations, which are:

1. Young Generation – this is where all new objects are allocated and aged. A minor Garbage collection occurs when this fills up.
2. Old or Tenured Generation – this is where long surviving objects are stored. When objects are stored in the Young Generation, a threshold for the object's age is set, and when that threshold is reached, the object is moved to the old generation.
3. Permanent Generation – this consists of JVM metadata for the runtime classes and application methods.
4. B11. Difference between EAR, JAR and WAR file in J2EE.

### **Definition**

A JAR file is a file with Java classes, associated metadata, and resources such as text and images aggregated into one file. A WAR file is a file that is used to distribute a collection of JAR files, JSP, Servlet, XML files, static web pages like HTML and other resources that constitute a web application. An EAR file is a standard JAR file that represents the modules of the application, and a metadata directory called META-INT which contains one or more deployment descriptors. Thus, this is the main difference between JAR WAR and EAR files.

### **Long Form**

JAR file stands for Java Archive while WAR file stands for Web Application Resource or Web Application Archive, and EAR stands for Enterprise Application Archive.

1. B12. Explain memory leak in Java.

**Java** provides out-of-box **memory management**. When we create an object using the new keyword, the JVM automatically allocates memory for that object. If the object is no longer is used by the application, the **garbage collector** automatically removes that object and free up space for other applications. Therefore, the programmer need not to manage memory manually like other procedural programming languages (C, and C++). Nevertheless, there is a chance of **memory leak** in a Java application. In this section, we will understand **what is a memory leak in Java**, its causes, detect and fixing of **memory leaks**.

## What is memory leak in Java?

In [Java](https://www.javatpoint.com/java-tutorial), the **memory leak** is a situation when the garbage collector does not recognize the unused objects and they remain in the memory indefinitely that reduces the amount of memory allocated to the application. Because the unused objects still being

referenced that may lead to **OutOfMemoryError.** It also affects the reliability of the application. The following figure represents the memory leak.

1. B13. How Garbage collection works in Java?

In **Java, garbage collection** is the process of managing memory, automatically. It finds the unused objects (that are no longer used by the program) and delete or remove them to free up the memory. The **garbage collection** mechanism uses several GC algorithms. The most popular algorithm that is used is **Mark and Sweep**.

In this section, we will learn **when an object becomes eligible to garbage collection, types of garbage collection**, and **Mark and Sweep algorithm**.

1. B14. Does Java garbage collector clean both heap and stack memory?

Garbage Collector In Java works only on Heap memory and not on stack memory , because of the main principal that stack works on which is ( Last In First Out).That explains it all in itself.

1. B15. Why garbage collection is required in Java?

When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.