CORE JAVA

UNIT - I NOTES

Q1. Write a short note on History of Java Language?

- Java is a high-level class-based, Object-Oriented Programming language that is designed to have as few Implementation dependencies as possible.
- Java is like C++ Programming. C++ is an Immediate descendent of C language. Every innovation in language design was driven by the need to take care of crucial issues that the first language couldn't tackle.

- Java was originally developed by James Gosling at ***Sun Microsystems***. It was released in May 1995 as a core component of Sun Microsystems' Java platform.
- James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. The small team of sun engineers called Green Team.

- 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.

- The language was initially called 'Oak' after an oak tree that stood outside Gosling's office. Later the project went by the name Green and was finally renamed Java, from Java coffee, ***a type of coffee from Indonesia***

- Java Derives a lot of its character from C and C++.The Java Creators realized that utilizing the well-known grammar of C and reverberating the Object Oriented Features of C++ would make their Language appealing to C/C++ developers. The difference between the way Java and other programming languages worked was revolutionary.

- By the late 1990s Java had brought multimedia to the Internet and started to grow beyond the Web, powering consumer devices (such as cellular telephones), retail and financial computers, and even the onboard computer of NASA'S Mars exploration rovers.

- **Sun Microsystems released the first public implementation as Java 1.0 in 1996**
- The Java 1.0 compiler was re-written in Java by Arthur van Hoff to comply strictly with the Java 1.0 language specification.

- Because of this popularity, Sun created different varieties of Java for different purposes, including Java SE for home computers, Java ME for embedded devices, and Java EE for Internet servers and supercomputers.

- In 2010 the Oracle Corporation took over the management of Java when it acquired Sun Microsystems.

Q2. Explain Java Language Architecture with its Components?

- Java Architecture is a collection of components, i.e., JVM, JRE, and JDK.
- It integrates the process of interpretation and compilation.
- It defines all the processes involved in creating a Java program. Java Architecture explains each and every step of how a program is compiled and executed.

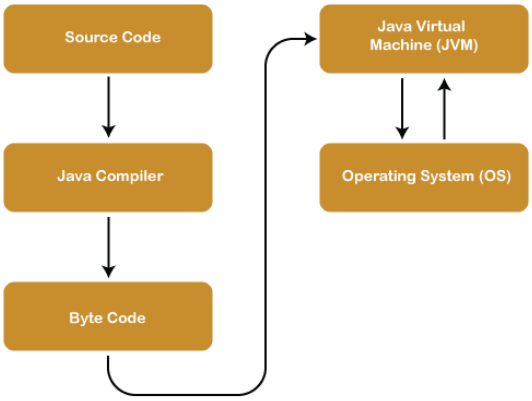
- Java Architecture can be explained by using the following steps:

  1)There is a process of compilation and interpretation in Java.

  2) Java compiler converts the Java code into byte code.

  3) After that, the JVM converts the byte code into machine code. The machine code is then executed by the machine.

- The following figure represents the Java Architecture in which each step is elaborate graphically.



Now let's dive deep to get more knowledge about Java Architecture. As we know that the Java architecture is a collection of components, so we will discuss each and every component into detail.

Components of Java Architecture:

The Java architecture includes the three main components:

- Java Virtual Machine (JVM)
- Java Runtime Environment (JRE)
- Java Development Kit (JDK)

Java Virtual Machine

The main feature of Java is WORA. WORA stands for Write Once Run Anywhere. The feature states that we can write our code once and use it anywhere or on any operating system. Our Java program can run any of the platforms only because of the Java Virtual Machine. It is a Java platform component that gives us an environment to execute java programs. JVM's main task is to convert byte code into machine code.

JVM, first of all, loads the code into memory and verifies it. After that, it executes the code and provides a runtime environment.

Java Runtime Environment

It provides an environment in which Java programs are executed. JRE takes our Java code, integrates it with the required libraries, and then starts the JVM to execute it. To learn more about the Java Runtime Environment, click here.

Java Development Kit

It is a software development environment used in the development of Java applications and applets. Java Development Kit holds JRE, a compiler, an interpreter or loader, and several development tools in it. To learn more about the Java Development Kit, click here

Q3. Java class File.

1) Java, being a platform-independent programming language, doesn't work on the one-step compilation. Instead, it involves a two-step execution, first through an OS-independent compiler; and second, in a virtual machine (JVM) which is custom-built for every operating system.

2) Source Code is Compiled by the compiler and we receive a byte code (with .class extension) after that that .class file gets interpret by an interpreter which is present in JVM and at the end we get machine code .

3) A Java class file is a file containing Java bytecode and having .class extension that can be executed by JVM.

4)A Java class file is created by a Java compiler from .java files as a result of successful compilation.

5) A single Java source file (or we can say .java file) may contain one class or more than one class.

6) So if a .java file has more than one class then each class will compile into a separate class files.

7) For Example: Save this below code as Test.java on your system. multiple class files will be generated

```
class Student
{
}
 class Test
```

```
{
    public static void main(String[] args)
    {
       System.out.println("Class File Structure");
    }
}
```

After compilation there will be 2 class files in corresponding folder named as:

Student.class

Test.class

8) A single class file structure contains attributes that describe a class file.

Representation of Class File Structure:

```
ClassFile
{
    magic_number;
    minor_version;
    major_version;
    constant_pool_count;
    constant_pool[];
    access_flags;
    this_class;
    super_class;
    interfaces_count;
    interfaces[];
    fields_count;
    fields[];
    methods_count;
    methods[];
    attributes_count;
```

attributes[];

}

9) The Following are explained in short:

There are 10 basic sections to the Java class file structure:

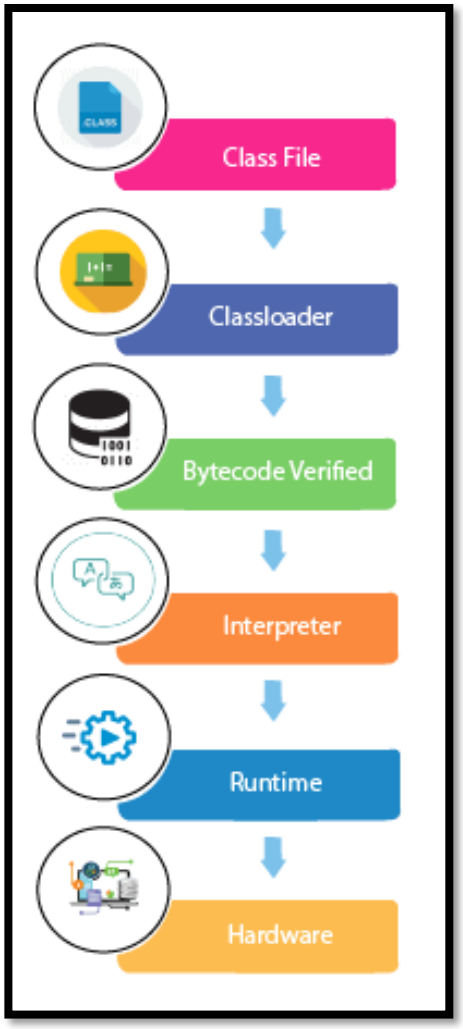
- **Magic Number:** First 4 bytes of a class file JVM uses it to identify whether .class file is generated by valid compiler or not .Value is in Hexadecimal form (0xCAFEBABE).
- **Version of Class File Format:** the minor and major versions of the class file
- **Constant Pool:** Pool of constants for the class
- **Access Flags:** for example whether the class is abstract, static, etc.
- **This Class:** The name of the current class
- **Super Class:** The name of the super class
- **Interfaces:** Any interfaces in the class
- **Fields:** Any fields in the class
- **Methods:** Any methods in the class
- **Attributes:** Any attributes of the class (for example the name of the sourcefile, etc.)

10) After the Class file is Generated ,Java Interpreter Converts that file into machine code:

Classloader: It is the subsystem of JVM that is used to load class files.

Bytecode Verifier: Checks the code fragments for illegal code that can violate access rights to objects.

Interpreter: Read bytecode stream then execute the instructions.



Q4. What is Java Runtime Environment (JRE)?

- JRE was originally developed by Sun Microsystems Inc., a wholly-owned subsidiary of Oracle Corporation.
- The Java Runtime Environment (JRE) is software that Java programs require to run correctly. Java is a computer language that powers many current web and mobile applications.
- The JRE is the underlying technology that communicates between the Java program and the operating system.
- It acts as a translator and facilitator, providing all the resources so that once you write Java software, it runs on any operating system without further modifications.
- A software program needs a runtime environment that provides access to memory and other system resources such as program files and dependencies.

- In the past, most software used the operating system directly as its runtime environment. However, this meant that developers had to write different code for each operating system that they wanted their applications to run on. The Java Runtime Environment (JRE) technology was created as a solution to this problem.
- The JRE is actually one of three Java platform components that are required for any Java program to run successfully. The Java Development Kit (JDK) and Java Virtual Machine (JVM) are the other two components.
- The JRE combines the Java code that you create by using the JDK with additional built-in code called libraries.
- It then creates a JVM instance, or local copy, that finally runs the Java programs.
- JVMs are available for multiple operating systems, and the JRE generates a single copy of your Java code that runs on all types of JVMs. In this way, the JRE facilitates platform independence for Java applications. You can write them once and run them anywhere.
    - ***Difference between the JRE, JVM, and JDK:***

The JDK is a software layer above the JRE that contains a compiler, a debugger, and other tools commonly found in any software development environment. You write code in English-like syntax in the JDK. The JDK compiles it and passes the byte code to the JRE. In contrast, the JRE contains class libraries, supporting files, and the JVM. It uses these software components to run the byte code on any device.

system, providing additional Java-specific resources. The Java Development Kit (JDK) and JRE interact to create a sustainable runtime environment that runs Java program files on any machine.

Q5. Explain Java Virtual Machine?

OR

Q. Explain Java Virtual Machine and Components?

Ans) Java Virtual Machine is important part of the JRE, which actually runs the programs (class files).

- It uses the java class libraries and the run-time libraries to execute these programs.
- Every operating system (OS) or platform will have a different VM. Java Virtual Machine (JVM) is an abstract computing machine.

- Java Runtime Environment (JRE) is an implementation of the JVM.
- Java Development Kit (JDK) contains JRE along with various development tools like Java libraries, Java Runtime Environment (JRE) source compilers, Java debuggers, bundling and deployment tools.
- JVM becomes an instance of JRE at runtime of a java program. It is widely known as a runtime interpreter.
- The Java virtual machine (JVM) is the foundation on
- Top of which the Java technology is built upon. It is the component of the Java technology responsible for its hardware and platform independence.
- JVM largely helps in the abstraction of inner implementation from the programmers who make use of libraries for their programs from IDK.
- Java Executable (java): Java is the application which runs/executes the class files,

  This internally calls the VM for running the programs.
- Just In Time Compiler (JIT): JIT is a module inside the JVM which helps in compiling certain parts of byte code into the machine code for higher performance. Note that only certain parts of byte code will be compiled to the machine code, the other parts are usually interpreted and executed.

Q6. What Is Java API?

Ans)

- API is the abbreviation of the term *Application Programming Interface*.
- The Java API is the group of classes included with the Java Development Environment.
- These classes are written using the Java language and run on the JVM.
- The Java API includes everything from collection classes to GUI classes.

- For every specific programming aspect there is special java API is available.
- For example, JDBC API is the group of java classes especially available for java and database connectivity applications.
- APIs are important software components bundled with the JDK.
- APIs in Java include classes, interfaces, and user Interfaces.
- They enable developers to integrate various applications and websites and offer real-time information.
- Different APIs differ in the level of security and privacy that they offer.
- It is possible to combine multiple web API to form a composite API. It is also known as a collection of data or service APIs.

Q7. What Is Java Platform?

Java Platform provides a java virtual machine (JVM) and an API, and this allows applications written for that platform to run on any compatible system with all the advantages of that platform to run on any compatible system with all the advantages of the Java Programming Language.

1)JAVA SE:

Java SE's API prides 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 technology applications.

2. 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.

3. 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 platform services.

4. JavaFX

JavaFX is a platform for creating rich internet applications using a lightweight user-interface API.

JavaFX applications use hardware-accelerated graphics and media engines to take advantage of higher-performance clients and a modern look-and-feel as well as high-level APIs for connecting to networked data sources.

Java FX applications may be clients of Java EE platform services.

Q8. What Is Java Development Kit?

OR

What Is JDK? Explain In Detail?

Ans)

- A Java Development Kit (DK) is a program development environment for writing Java applets and applications.
- It consists of a runtime environment that "sits on top" of the operating system layer as well as the tools and programming that developers need to compile, debug, and run applets and applications written in the Java language.
- It includes the Java Runtime Environment (RE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.
- The JDK has a private Java Virtual Machine (JVM) and a few other resources necessary for the development of a Java Application.
- The Java Runtime Environment in JDK is usually called Private Runtime because it is separated from the regular JRE and has extra content.

- The Private Runtime in JDK contains a JVM and all the class libraries present in the production environment, as well as additional libraries useful to developers, e.g., internationalization libraries and the IDL libraries.
- 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

Q9. What is Lambda Expressions ?

- Lambda expression is a new and important feature of Java which was included in Java SE 8.
- It provides a clear and concise way to represent one method interface using an expression.
- It is very useful in collection library. It helps to iterate, filter and extract data from collection.
- The Lambda expression is used to provide the implementation of an interface which has functional interface.
- It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.
- Java lambda expression is treated as a function, so compiler does not create .class file.
- Lambda expression provides implementation of functional interface. An interface which has only one abstract method is called functional interface.
- Java provides an anotation @FunctionalInterface, which is used to declare an interface as functional interface.
- Why use Lambda Expression
- To provide the implementation of Functional interface.
- Less coding.
- Java Lambda Expression Syntax
- (argument-list) -> {body}
- Java lambda expression is consisted of three components.

- 1) Argument-list: It can be empty or non-empty as well.
- 2) Arrow-token: It is used to link arguments-list and body of expression.
- 3) Body: It contains expressions and statements for lambda expression.
    - No Parameter Syntax

```
() -> {
//Body of no parameter lambda
}
```

    - One Parameter Syntax

```
(p1) -> {
//Body of single parameter lambda
}
```

    - Two Parameter Syntax

```
(p1,p2) -> {
//Body of multiple parameter lambda
}
```

Q11.What is Java Type Annotations ?

- Java Annotation is a kind of a tag that represents the metadata or information attached with class, interface, methods, or fields to show some additional information that Java compiler and JVM can use.
- There is no direct effect of Annotations on the operation of the code they annotate; they do not affect the execution of the program. Annotations provide supplemental information about a program.

Some points about Annotations are:

- They start with '@'.
- They do not change the action or execution of a compiled program.
- Annotations help to associate metadata or information to the elements of the program like classes, instance variables, interfaces, constructors, methods, etc.
- We cannot consider Annotations as pure comments as they can change the way a compiler treats a program

- ♦ The type annotations are applicable to any place where there is a use of a type. For example, if we want to annotate the return type of a method, we can declare these annotations with @Target annotation.
- ♦ These annotations can be categorized as:
- ♦ 1. Predefined annotations
  - • @Deprecated
  - • @Override
  - • @SuppressWarnings
  - • @SafeVarargs
  - • @FunctionalInterface

  2. Custom annotations

  3. Meta-annotations
  - • @Retention
  - • @Documented
  - • @Target
  - • @Inherited
  - • @Repeatable

Write a short note on Java Method References?

- Java provides a new feature called method reference in Java 8.
- Method reference is used to refer method of functional interface.
- It is compact and easy form of lambda expression.
- Each time when you are using lambda expression to just referring a method, you can replace your lambda expression with method reference.
- There are following types of method references in java:
  - Reference to a static method.
  - Reference to an instance method.
  - Reference to a constructor.

1) Reference to a Static Method

You can refer to static method defined in the class. Following is the syntax and example which describe the process of referring static method in Java.

Syntax

ContainingClass::staticMethodName

2) Reference to an Instance Method

like static methods, you can refer instance methods also. In the following example, we are describing the process of referring the instance method.

Syntax

containingObject::instanceMethodName

3) Reference to a Constructor

You can refer a constructor by using the new keyword. Here, we are referring constructor with the help of functional interface.

Syntax

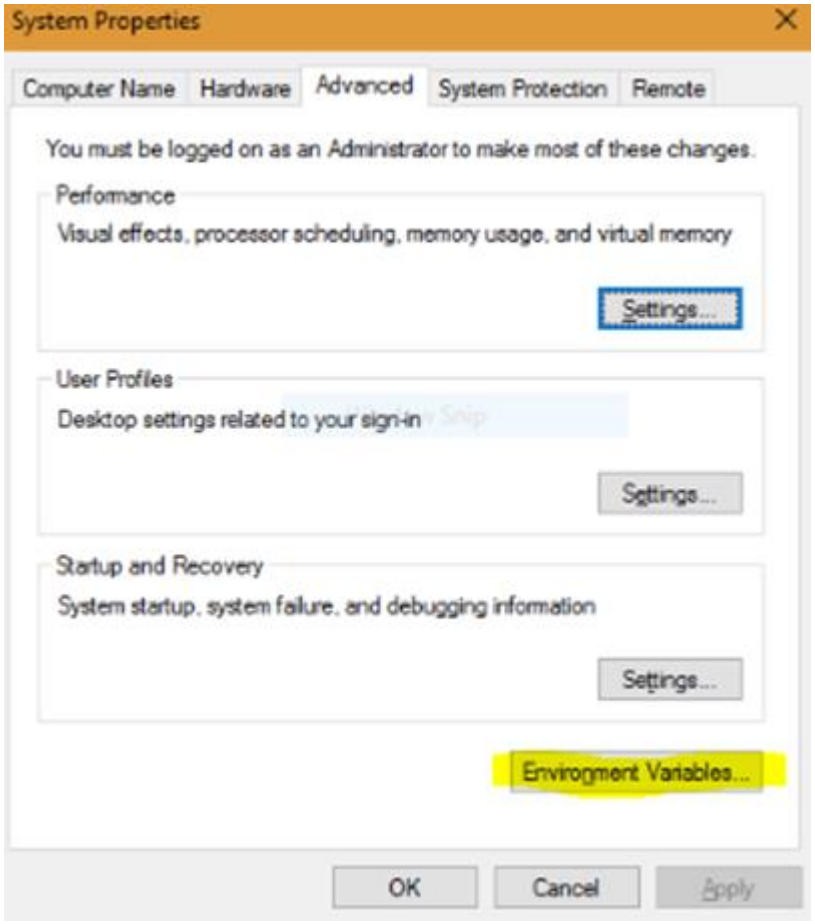
ClassName::**new**

Q.12. How to Set the Java Path Environment Variable ?

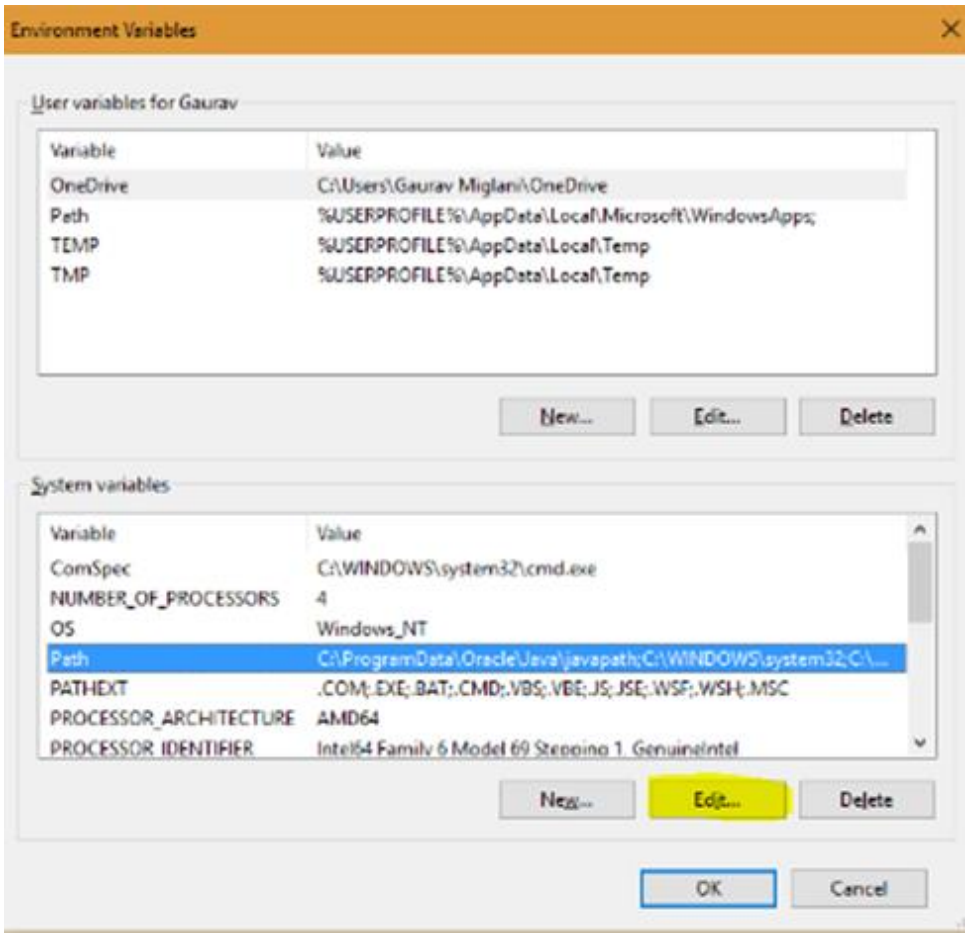
- **Step 1** : Download Java JDK from https://www.oracle.com/java/technologies/downloads/#java8-windows

- **Step 2** : After download , run the .exe file and follow the instructions to install Java on your machine.Once you install Java on your machine,you have to set up the environment variable .

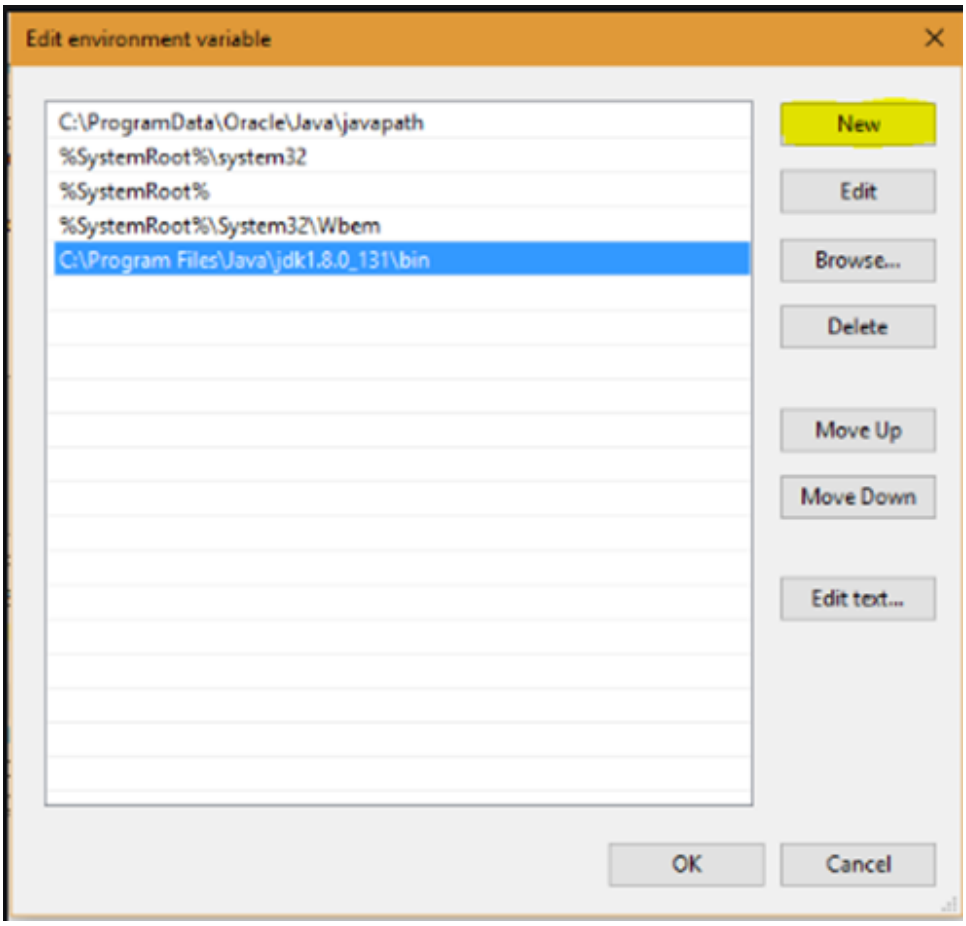
- **Step 3** : Go to **Control Panel -> System and Security -> System.** Under the Advanced System Setting option click on **Environment Variables** as highlighted below.



- **Step 4** : Now, you have to alter the "Path" variable under System variables so that it also contains the path to the Java environment. Select the "Path" variable and click on the Edit button as highlighted below.



- **Step 5** : You will see a list of different paths, click on the New button, and then add the path where java is installed. By default, java is installed in "C:\Program Files\Java\jdk\bin" folder OR "C:\Program Files(x86)\Java\jdk\bin". In case, you have installed java at any other location, then add that path.



- **Step 6 :** Click on OK, Save the settings, and you are done !! Now to check whether the installation is done correctly, open the command prompt and type ***javac -version.*** You will see that java is running on your machine.

Q13. What are the Roles of Java Compiler And Interpreter OR

Difference between Java Compiler and Interpreter. OR

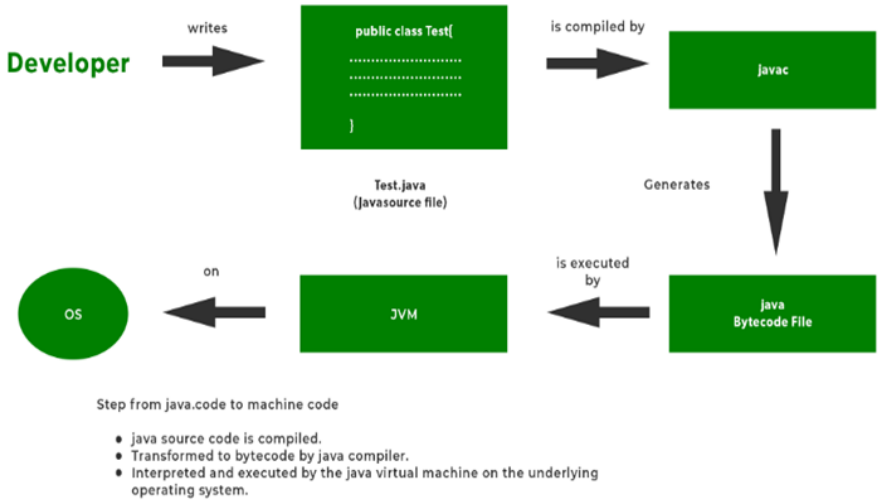
How is Interpreter Different Than a Compiler in Java?

- Compiler and Interpreter are two different ways to translate a program from programming or scripting language to machine language .
- Java is both Interpreted as well as Compiled language.

Java Compiler

- A compiler in Java translates the entire source code into a machine-code file or any intermediate code, and that file is then executed.
- It is platform-independent.
- A bytecode is basically an intermediate code generated by the compiler after the compilation of its source code.
- The compiler scans the entire program first and then translates it into machine code .
- Compiler is used by languages such as C, C++, etc.

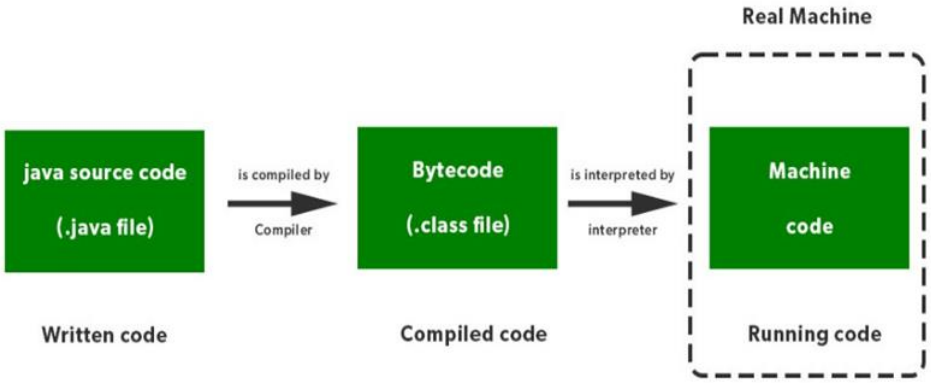
JAVA COMPILER



- **Roles of Java Compiler** : It scans the complete source code in one go and then highlights the error . Requires more memory to produce the bytecode.

Java Interpreter

- An Interpreter is a computer program that converts the high-level program statement into Assembly-level language. It converts the code into machine code when the program is run.
- The Interpreter scans the program line by line and translates it into machine code .



Roles of Java Interpreter :To convert the bytecode into the native code of the machine. This process is done line by line.

- An interpreter is used by languages such as Java, and Python.

Q14. Write a Program For The Following .

a.WAP to Add 2 Numbers.

```
import java.util.Scanner;  // Import the Scanner class
        public class MyClass {
                public static void main(String[] args) {
                 int x, y, sum;
                Scanner myObj = new Scanner(System.in); // Create a Scanner object
                 System.out.println("Enter  a number:");
                x = myObj.nextInt(); // Read user input

                 System.out.println("Enter another number:");
```

```
                y = myObj.nextInt(); // Read user input

    sum = x + y;
    System.out.println("Sum is: " + sum);
  }
}
```

OUTPUT :

```
 Enter  a number:4
Enter another number:5
Sum is: 9
```

b. WAP To Find Factorial Of a Number.

```
class FactorialExample{
     public static void main(String args[]){
         int i,fact=1;
         int number=5;//It is the number to calculate factorial
         for(i=1;i<=number;i++){
                 fact=fact*i;
  }
          System.out.println("Factorial of "+number+" is: "+fact);
 }
}
```

OUTPUT :

```
Factorial of 4 is: 24
Factorial of 5 is: 120
```

c.WAP which asks the user to enter his/her name and greets them with "Hello<name>,have a good day" text.

```
import java.util.Scanner;

        public class ps3 {
         public static void main(String[] args) {

       Scanner sc  = new Scanner(System.in);
//       scanner class object.

       System.out.println("PLEASE, ENTER YOUR NAME");
       String name  = sc.next();

       System.out.println(" hello "  +  name  +  " have a good day ");
   }
}
```

OUTPUT :

```
PLEASE, ENTER YOUR NAME
Jiya
hello Jiya have a good day
```

Q.15. Applications Of Java.

- Java has become the most robust programming language because of its amazing features. Some of its features are platform independence, high performance, Object orientation
- Applications of Java in the real world are as follows :
- **Mobile Applications :** Java is a cross-platform framework that is used to build applications that run across smartphones and other small-screen devices.

As per a survey it is observed that Java is the second most widely used language for mobile application development.

Mobile applications that are created using Java include some popular ones like Netflix, Twitter, Spotify, and many more.

- **Artificial Intelligence :** Java is one of the best languages for AI projects. Its infrastructure is well embedded with intelligent software to enhance AI programming. It has amazing features like better interaction with users, ease of debugging, easy-to-code features, standard widget tools, and a lot more.

- **Web Applications :** Java is just perfect for developing web applications because of its ability to interact with a large number of systems. It allows us to create dynamic web applications that interact with interfaces. The presence of **JSP(Java Server Pages),** web servers, spring, and Hibernate provides feasibility in the web development process.

- **Gaming Applications :** Java has proved to be the most desirable option for game development because of the presence of a wide variety of open-source frameworks. Popular games like Mission Impossible III, Minecraft, and Asphalt 6 are developed in Java**.** Android games use Java as a primary language because Java supports the **Dalvik Virtual Machine (DVM)** which is specially designed to run on the Android platform.

Q16. Explain Java main Method .

- The first method you encounter is **public static void main(String[] args).**
- The starting point of any Java Program is the main() method. Technically, the main method is the starting point where the Java program starts its execution.
- JVM always look for this method signature to start running an application.

public

- public is an access modifier. The scope of public access modifier is everywhere. It has no restrictions.
- Data members, methods and classes that declared public can be accessed from anywhere.
- If you make a main() method public then it is allowed to be executed by any program globally.
- If you make a main() method non-public then it is not allowed to be executed by any program.

static

- JVM can not create an object of class at run time to access the main method.
- By declaring the main() method as static, JVM can invoke it without instantiating the class.
- If we don't declare the main method as static then JVM cannot call it to execute the program.

void

- Void means the Method will not return any value.
- In Java, every method provides the return type whereas Java main method doesn't return any value.

- Java program starts with main method and terminates once the main method is finished executing.
- If we make main method to return a value, JVM cannot do anything with the returned value. So there is no need to return any value**.**

main

- main is the name of Java main method.
- It is the first thing that runs when you compile and run the Java program.
- The main method is searched by JVM as the starting point where the Java program starts its execution.

String[] args

- String[] args is a parameter that accepts inputs.
- The name of the String array is **'args'** but it can be of anything based on users choice.
- The type of args is always String[] because Java main method accepts only a single argument of type String array.
- We can replace the array name with anything then also it works fine. i.e. **String[] raj** or **String[] stm**

Q17. "In Java a string and a character array are different." Justify

- In previous programs, we have already used String.
- Similar to other programming languages, in Java also, string is set of characters.

- But unlike other languages that implements string as a character array, in Java string is implemented as a library class and we use its objects to store set of characters.
- As now we know that, String is a library class and we create its objects to store set of characters in it Java provides a set off of functions and constructors that makes the string handling convenient.
- Here in Java, we will mainly study about two string companion classes; **String** and **StringBuffer.**
- Both these classes are available under java.lang package and therefore available directly in all the Java programs.
- Remember that, both these classes new desired as fast, it means that either of these classes y he derived.
- This allows certain optimization that increase to take place on comm string operations.

String Constructors

Q18. List and explain the constructors class String

- The String class contains many constructors associated to byte[], char[] stringbuffer, StringBuilder and another String. A few of them are discussed here:

  String ()

  String(char array[])

  String(char array[], int start_ index, int num_ char)

  String(String obj)

  String(byte asciiChars[])

  String (byte asciiChara[], int startIndex, int num _Chars)

- **First constructor** is the default constructor that creates an empty string. You can use this constructor as:

  String st = new String();

- This will create a sting object without any characters in it. Generally we don't create a string object without any character. But instead we use either of remaining three to create a string object to initialize by another string or character array. Second constructor creates a string object that is initialized by a character array.
- **Second constructor** creates a string object that is initialized by a character array. Use of this constructor is shown here:

  char ch[] = {'J'’,'a’,’m’,’e’,’s’};

  String st = new String(ch);

  This example will create a string object st which is initialized as "**James**".

- **Third constructor** creates a string object in which we specify index subrange from a character array.
- It requires start _ index and num _chars to specify string range. Refer following example:

  char ch[]= {'a', 'b’,’e','d’,’e’,'g', 'h');

String st =new String(ch,2,4);

  This initializes **st** with the characters **cdef**.

- We can also create a string containing the same characters as another string, by using **Fourth constructor**. Refer following example:

  String str= new String(st);

  The last two constructors work with byte[] array.

- **Fifth constructor** creates a string object and initializes it with ASCII character set as byte array. Refer following example.

  byte ascii[]= (65,66,67,68,69,70);

- String st = new String(ascii);This will initialize String st with **ABCDEF**.
- **The sixth constructor** allows us to use ASCII character set with sub-range.

Q19. List and explain the commonly used method of class String.

- The String class defines numbers of library methods which helps us to perform string manipulation tasks.
- Here is a list of commonly used methods along with a small description on them.

Int length()

- Returns length of entered string

String toLowerCase()

- Converts and returns a string into lowercase

String toUppercase()

- Converts and returns a string into uppercase

String replace(char cl, char c2)

- Replaces cl with c2 in string and return

boolean equals(String obj)

- Compares string with specified string object and returns true if strings are equal otherwise returns false

boolean equalsIgonoreCase(String obj)

- Compares string with specified string object by ignoring case of characters and returns true if strings are equal otherwise returns false

char charAt(int index)

- Returns character present at specified index.

char[] toCharArray()

- Convert all the characters in a String object into a character array boolean startsWith(String str)

boolean ends With(String str)

- Determines whether given string begins with specified string or not

int indexOf(char)

- To search for the first occurrence of specified character.

int lastIndexOf(char)

- To search for the first occurrence of specified character.

void getChars(int src_start_index, int src_end_index, char destination_array[], int desti_array_start_index)

- Assigns a substring to destination character array

byte[] getBytes(String charset_name)

- This method is actually an alternative to getChars() that stores the characters in an array of bytes and it uses the default character-to-byte conversions provided by the platform.

int compareTo(String str)

- Compares invoking string object with parameter string and returns difference of ASCII values. Returns Zero if strings are totally equal otherwise any non-zero value.

String concat(String str)

- Concatenates invoking string with parameter string and returns new concatenated string.

String trim()

- Returns a new string after removing whitespaces that appears in leading or trailing end of the invoking string object

Q.20) What are the Statements in Java

- A statement causes the Computer to carry out some definite action.
- In Java, each statement is a complete unit of execution.
- There are three Different classes of statements in Java as Expression statements, compound statements, and control statements.
- They are described below:

1)Expression Statements:

A Java expression consists of variables, operators, literals, and method calls.

We can convert an expression into a statement by terminating the expression with a ;. These are known as expression statements.

a=100 //expression

a=100; //statement

2)Compound Statements:

In several places Java requires you to write one statement. But you might want to put several statements there instead of just one.

You do that by packaging several statements up into a single statement, called a compound statement

. Think of it as like putting a few things into a box and closing the box, so that it looks like just one thing, the box.

A compound statement begins with a left curly brace and ends with a right curly brace. In between, write any sequence of statements, one after another.

You can put any number of statements inside the braces, including only one, or even none. Compound statement { } is a do-nothing statement.

3)Control Statements:

A control statement works as a determiner for deciding the next task of the other statements whether to execute or not.

An 'If' statement decides whether to execute a statement or which statement has to execute first between the two. In Java, the control statements are divided into three categories which are selection statements, iteration statements, and jump statements.

A program can execute from top to bottom but if we use a control statement. We can set an order for executing a program based on values and logic.

- Decision Making in Java
  - Simple if Statement

- if...else Statement
- Nested if statement
- if...else if...else statement
- Switch statement

- Looping Statements in Java
  - While
  - Do...while
  - For
  - For-Each Loop
- Branching Statements in Java
  - Break
  - Continue
  - Return

Q.21) What are White Spaces?

Blank Spaces should be:

1)Between keywords and following parenthesis

Eg: white(true){...}

2)After commas in argument lists

Eg:aMethod(arg1, arg2).

3)Between binary operators and their arguments except from '.' operator. Between expressions in a for statement . After the type in a case expression ,e.g: a=(Vector) b;

4)The java.lang.Character.is.WhiteSpace() is an inbuilt method in a java that determines if the specified character(Unicode code point) is white space according to java.

Q.22) What is identifiers? Write the rules of identifiers.

Ans:-

Identifiers in Java are symbolic names used for identification. They can be a class name, variable name, method name, package name, constant name, and more. However, In Java, There are some reserved words that can not be used as an identifier.

For every identifier there are some conventions that should be used before declaring them. Let's understand it with a simple Java program:

```
1. public class HelloJava {
2.     public static void main(String[] args) {
3.         System.out.println("Hello JavaTpoint");
4.     }
5. }
```

From the above example, we have the following Java identifiers:

1. HelloJava (class name)
2. main (main method)
3. String (Predefined Class name)
4. args (String variables)
5. System (Predefined class)
6. out (variable name)
7. println (method)

Rules for Identifiers in Java

There are some rules and conventions for declaring the identifiers in Java. If the identifiers are not properly declared, we may get a compile-time error. Following are some rules and conventions for declaring identifiers:

1. A valid identifier must have characters [A-Z] or [a-z] or numbers [0-9], and underscore(_) or a dollar sign ($). for example, @javatpoint is not a valid identifier because it contains a special character which is @.
2. There should not be any space in an identifier. For example, java tpoint is an invalid identifier.

3. An identifier should not contain a number at the starting. For example, 123javatpoint is an invalid identifier.
4. An identifier should be of length 4-15 letters only. However, there is no limit on its length. But, it is good to follow the standard conventions.
5. We can't use the Java reserved keywords as an identifier such as int, float, double, char, etc. For example, int double is an invalid identifier in Java.
6. An identifier should not be any query language keywords such as SELECT, FROM, COUNT, DELETE, etc.

Q23. Distinguish between keyword and identifier.

Ans:-

SR. NO.	KEYWORD	IDENTIFIER
1	Keywords are predefined word that gets reserved for working program that have special meaning and cannot get used anywhere else.	Identifiers are the values used to define different programming items such as variables, integers, structures, unions and others and mostly have an alphabetic character.
2	Specify the type/kind of entity.	Identify the name of a particular entity.
3	It always starts with a lowercase letter.	First character can be a uppercase, lowercase letter or underscore.

4	A keyword should be in lower case.	An identifier can be in upper case or lower case.
5	A keyword contains only alphabetical characters.	An identifier can consist of alphabetical characters, digits and underscores.
6	They help to identify a specific property that exists within a computer language.	They help to locate the name of the entity that gets defined along with a keyword.
7	No special symbol, punctuation is used.	No punctuation or special symbol except ‘underscore’ is used.
8	Examples of keywords are: int, char, if, while, do, class etc.	Examples of identifiers are: Test, count1, high_speed, etc.

Q.24) Write the list of keywords Or reserved identifier Or reserved words for literal values Or unused.

Ans:- Java Reserved Keywords

- Java reserved keywords are predefined words, which are reserved for any functionality or meaning.
- We can not use these keywords as our identifier names, such as class name or method name. These keywords are used by the syntax of Java for some functionality.
- In the Java programming Language , a keyword is any one of 67 reserved words that have a predefined meaning in the language.
- If we use a reserved word as our variable name, it will throw an error.

In Java, every reserved word has a unique meaning and functionality.

Consider the below syntax:

1. double marks;

in the above statement, double is a reserved word while marks is a valid identifier.

Below is the list of reserved keywords in Java:

abstract	continue	for	protected	transient
assert	default	goto	public	try
boolean	do	if	static	throws
break	double	implements	strictfp	package
byte	else	import	super	private
case	enum	interface	short	switch
catch	extends	instanceof	return	void

char	final	int	synchronized	volatile
class	finally	long	throw	date
const	float	native	this	while

Q.25) **what is comments and Explain the types of comments?**

- In a program, comments are like indents one makes, they are used so that it is easier for someone who isn't familiar with the language to be able to understand the code.
- It will also make the job easier for you, as a coder, to find errors in the code since you will be easily able to find the location of the bug.
- Comments are ignored by the compiler while compiling a code, which makes the job more complex in the long run when they have to go through so much code to find one line.

In Java there are three types of comments:

1. Single-line comment.
2. Multi-line comment.
3. Documentation comment.

A. Single-line comments

A beginner-level programmer uses mostly single-line comments for describing the code functionality. It's the easiest typed comments.

Syntax:

//Comments here(Text in this line only is considered as comment)

B. Multi-line Comments:

To describe a full method in a code or a complex snippet single line comments can be tedious to write since we have to give '//' at every line. So to overcome this multi-line comments can be used.

Syntax:

/*Comment starts

Comment ends*/

C. Documentation Comments:

This type of comment is used generally when writing code for a project/software package, since it helps to generate a documentation page for reference, which can be used for getting information about methods present, its parameters, etc.

/**Comment start

*

*tags are used in order to specify a parameter

*or method or heading

*HTML tags can also be used

*such as <h1>

*

comment ends/

Q.26) What Are Curly Braces in Java?

Ans:-

- A Java program is like an outline. With an outline, you can organize thoughts and ideas, help people see forests instead of trees, and generally show that you're a member of the Tidy Persons Club. The program in the listing starts with a big header line that says, "Here comes a class named

Displayer." After that first big header, a subheader announces, "Here comes a method named main."

- Now, if a Java program is like an outline, why doesn't a program look like an outline? What takes the place of the Roman numerals, capital letters, and other things? The answer is twofold:

1. In a Java program, curly braces enclose meaningful units of code.
2. You, the programmer, can (and should) indent lines so that other programmers can see the outline form of your code at a glance.

3. In a Java program, everything is subordinate to the top line — the line with class in it. To indicate that everything else in the code is subordinate to this class line, you use curly braces. Everything else in the code goes inside these curly braces.

   In an outline, some stuff is subordinate to a capital letter A item. In a Java program, some lines are subordinate to the method header. To indicate that something is subordinate to a method header, you use curly braces.

If you put curly braces in the wrong places or omit curly braces where the braces should be, your program probably won't work at all. If your program works, it'll probably work incorrectly.

If you don't indent lines of code in an informative manner, your program will still work correctly, but neither you nor any other programmer will be able to figure out what you were thinking when you wrote the code.

Q.27) Why are curly braces in programming languages important?

Ans:-

Curly braces play a big role in code structure within popular programming languages such as Java, C++ and more. Here's how to properly line up code with curly brackets.

At their core, all programming languages share similarities. In their most basic form, they all break down into the same common sets of functions:

1. data variable declarations
2. conditional logic
3. iterative functions

- Computer programs can use data, evaluate if-then conditions on the data and use extremely fast iterative loops to perform these functions.
- In software programs, these functions get organized into methods. Object-oriented languages further organize these functions into classes or objects.
- Regardless of whether a given programming language is object-oriented or procedural, these fundamental concepts still apply.

  However, one major difference hinges on how developers use curly braces in programming languages.

Curly-brace code blocks

Different programming languages have various ways to delineate the start and end points of a programming structure, such as a loop, method or conditional statement. For example, Java and C++ are often referred to as curly brace languages because curly braces are used to define the start and end of a code block.

```
public void flagTest()
{
  boolean flag = true;
  int i = 10;
  if (flag == true)
  {
```

```
   for( int i=0; i<10; i++ )
{
  System.out.print("flag is true");
}
 }
 flag = false;
}
```

Q.28) What is code of block?

Ans:-

1. Java allows two or more statements to be grouped into blocks of code, also called code blocks.
2. This is done by enclosing the statements between opening and closing curly braces.
3. Once a block of code has been created, it becomes a logical unit that can be used any place that a single statement can.
4. The key point here is that whenever you need to logically link two or more statements, you do so by creating a block.
5. Let's look at another example. The following program uses a block of code as the target of a for loop

```
/*
Demonstrate a block of code.
Call this file "BlockTest.java"
*/
class BlockTest {
```

```
public static void main(String args[]) {
 int x;
 int y = 3;
 for(x = 0; x<y;x++) // the target of this loop is a block
{
        System.out.print(x);
 }//for
}//main
}//class
```

Output: 0 1 2

1. In this case, the target of the for loop is a block of code and not just a single statement.
2. Thus, each time the loop iterates, the statement inside the block will be executed.
3. This fact is, of course, evidenced by the output generated by the program.

Q.29) What is variable and explain the type of variables

Ans:-

1. Variable in Java is a data container that saves the data values during Java program execution.
2. Every variable is assigned a data type that designates the type and quantity of value it can hold. A variable is a memory location name for the data.
3. A variable is a name given to a memory location. It is the basic unit of storage in a program.
1. The value stored in a variable can be changed during program execution.
2. A variable is only a name given to a memory location. All the operations done on the variable affect that memory location.
3. In Java, all variables must be declared before use.

How to declare variables?

We can declare variables in Java as pictorially depicted below as a visual aid.

From the image, it can be easily perceived that while declaring a variable, we need to take care of two things that are:

1. datatype: Type of data that can be stored in this variable.

2. data_name: Name given to the variable.

In this way, a name can only be given to a memory location. It can be assigned values in two ways:

1. Variable Initialization
2. Assigning value by taking input

How to initialize variables?

It can be perceived with the help of 3 components that are as follows:

1. datatype: Type of data that can be stored in this variable.
2. variable_name: Name given to the variable.

value: It is the initial value stored in the variable.
Illustration:

```
int time = 10, speed = 20;

// Declaring and initializing integer variable
```

Types of Variables in Java

Now let us discuss different types of variables which are listed as follows:

1. Local Variables
2. Instance Variables
3. Static Variables

1. Local Variables

A variable defined within a block or method or constructor is called a local variable.

1. These variables are created when the block is entered, or the function is called and destroyed after exiting from the block or when the call returns from the function.
2. The scope of these variables exists only within the block in which the variables are declared, i.e., we can access these variables only within that block.
3. Initialization of the local variable is mandatory before using it in the defined scope.

2. Instance Variables

Instance variables are non-static variables and are declared in a class outside of any method, constructor, or block.

1. As instance variables are declared in a class, these variables are created when an object of the class is created and destroyed when the object is destroyed.
2. Unlike local variables, we may use access specifiers for instance variables. If we do not specify any access specifier, then the default access specifier will be used.
3. Initialization of an instance variable is not mandatory. Its default value is 0.
4. Instance variables can be accessed only by creating objects.

3. Static Variables

Static variables are also known as class variables.

1. These variables are declared similarly as instance variables. The difference is that static variables are declared using the static keyword within a class outside of any method, constructor or block.
2. Unlike instance variables, we can only have one copy of a static variable per class, irrespective of how many objects we create.

3. Static variables are created at the start of program execution and destroyed automatically when execution ends.
4. Initialization of a static variable is not mandatory. Its default value is 0.
5. If we access a static variable like an instance variable (through an object), the compiler will show a warning message, which won't halt the program. The compiler will replace the object name with the class name automatically.
6. If we access a static variable without the class name, the compiler will automatically append the class name.

Q.30 Distinguish between Local variable, Intance variable and Static variable

Ans:-

Local Variable	Instance Variable	Static Variable
Defined within a method or a code block	Defined outside a method at the class level	Defined outside a method at the class level
Is only accessible in the method/code block where it is declared	Is accessible throughout the class	Is accessible throughout the class

Remains in memory as long as the method executes	Remains in memory as long as the object is in memory	Remains in memory as long as program executes
Does not require any special keyword	Does not require any special keyword but any access specifier (private, protected or public) can be specified. Typically, private or protected is used	Requires the static keyword to be specified. In addition, any access specifier (private, protected or public) can be specified. Typically, public is used
Requires to be initialized before it is used	Is given default value based on its data type, so does not require to be initialized before it is used	Is given default value based on its data type, so does not require to be initialized before it is used.

Q.31) What is variable name?

Ans:-

Every programming language has its own set of rules and conventions for the kinds of names that you're allowed to use, and the Java programming language is no different. The rules and conventions for naming your variables can be summarized as follows:

1. Variable names are case-sensitive.
2. A variable's name can be any legal identifier — an unlimited-length sequence of Unicode letters and digits, beginning with a letter, the dollar sign "$", or the underscore character "_".
3. The convention, however, is to always begin your variable names with a letter, not "$" or "_". Additionally, the dollar sign character, by convention,

is never used at all. You may find some situations where auto-generated names will contain the dollar sign, but your variable names should always avoid using it.

4. A similar convention exists for the underscore character; while it's technically legal to begin your variable's name with "_", this practice is discouraged. White space is not permitted.
5. A keyword must not be used

Factorial of 4 is: 24

Q.32) What are Primitive Data Types

- In Java, the primitive data types are the predefined data types of Java.
- They specify the size and type of any standard values. Java has 8 primitive data types namely byte, short, int, long, float, double, char and boolean.
- When a primitive data type is stored, it is the stack that the values will be assigned.
- When a variable is copied then another copy of the variable is created and changes made to the copied variable will not reflect changes in the original variable.

integer: This group includes byte, short, int, long

byte : It is 1 byte(8-bits) integer data type. Value range from -128 to 127. Default value zero. Example: byte b=10;

short : It is 2 bytes(16-bits) integer data type. Value range from -32768 to 32767. Default value zero. Example:

short s=11;

Int : It is 4 bytes(32-bits) integer data type. Value range from -2147483648 to 2147483647. Default value zero.

Example: int i=10;

Long : It is 8 bytes(64-bits) integer data type. Value range from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Default value zero. Example: long l=100012;

Characters:

This group represent char, which represent symbols in a character set, like letters and numbers.

Char: It is 2 bytes(16-bits) unsigned unicode character. Range 0 to 65,535.

Example: char c='a'; Code:

Boolean: Boolean type is used when we want to test a particular condition during the execution of the program. There are only two values that a Boolean type can take: true or false.

Remember, both these words have been declared as keyword. Boolean type is denoted by the keyword Boolean and uses only 1 bit of storage.

Object Data Type: These are also referred to as Non-primitive or Reference Data Type. They are so-called because they refer to any particular object. Unlike the primitive data types, the non-primitive ones are created by the users in Java. Examples include arrays, strings, classes, interfaces etc. When the reference variables will be stored, the variable will be stored in the stack and the original object will be stored in the heap. In Object data type although two copies will be created they both will point to the same variable in the heap, hence changes made to any variable will reflect the change in both the variables. Here is a Java program to demonstrate arrays(an object data type) in Java.

Q.33) Write a Short note on Object Reference Types.

In Java there are four types of references differentiated on the way by which they are garbage collected.

Strong References

Weak References

Soft References

Phantom References

Prerequisite: Garbage Collection

Strong References: This is the default type/class of Reference Object. Any object which has an active strong reference are not eligible for garbage collection. The object is garbage collected only when the variable which was strongly referenced points to null. MyClass obj = new MyClass ();

Here 'obj' object is strong reference to newly created instance of MyClass, currently obj is active object so can't be garbage collected. Obj = null; //'obj' object is no longer referencing to the instance. So the 'MyClass type object is now available for garbage collection.

Weak References: Weak Reference Objects are not the default type/class of Reference Object and they should be explicitly specified while using them.

This type of reference is used in WeakHashMap to reference the entry objects .

If JVM detects an object with only weak references (i.e. no strong or soft references linked to any object object), this object will be marked for garbage collection.

To create such references java.lang.ref.WeakReference class is used.

These references are used in real time applications while establishing a DBConnection which might be cleaned up by Garbage Collector when the application using the database gets closed.

Two different levels of weakness can be enlisted: Soft and Phantom

Soft References: In Soft reference, even if the object is free for garbage collection then also its not garbage collected, until JVM is in need of memory badly.The objects gets cleared from the memory when JVM runs out of memory.To create such references java.lang.ref.SoftReference class is used.

String

- A Java string is a sequence of characters that exists as an object of the class java.lang. Java strings are created and manipulated through the string class. Once created, a string is immutable – its value cannot be changed.
- A string is sequence of characters. A class is a user-defined template for creating an object. A string class is a user-defined template for creating and manipulating string objects, which are sequences of characters.

Methods of class string enable the following capabilities:

Examining individual characters in the string;

Comparing strings;

Searching strings; and

Copying strings with characters converted from uppercase to lowercase or vice versa.

Creating a Java string

Below is one simple syntax for creating a Java string.

String greeting = "Hello world!"; In this example, "Hello world!" is a string literal, which is a series of characters encased in quotation marks. The compiler will create an object with that value, and the string value is written into the source code of a computer program.

Whenever a new string variable or object is created, it is stored in computer memory.

Memory is split into two high-level blocks, the stack and the heap. Java stores object values in heap memory; references to the value are stored in the stack.

Another way to create strings is to use the new keyword, as in the following example.

String s1 = new String("Hello world!");

That code will create an s1 string object and a reference variable.

As noted, Java strings are immutable. Once created, a string using string class cannot be changed. Any attempted changes will create another string instance.

Users can perform basic operations, such as comparing strings. The .equals() method is used to compare strings, as in the following example.

String str = “one”;

String st = “one”;

System.out.println (str.equals(st));

The above code will compare the two values of each string to see if they are the same

Brief about Operators available in Java Language

Operator in Java is a symbol that is used to perform operations. For example: +, -, *, / etc.

There are many types of operators in Java which are given below:

1. Unary Operator,
2. Arithmetic Operator,
3. Shift Operator,
4. Relational Operator,
5. Bitwise Operator,
6. Logical Operator,
7. Ternary Operator and
8. Assignment Operator

Properties of Operators :

Operators, functions ,constants and variables are combined together to form expressions .

Precedence of Operators:

Operator Type	**Category**	**Precedence**
Unary	Postfix Prefix	expr++ expr-- ++expr. expr +exprexpr ~!
Arithmetic	multiplicative additive	*/% + -

Shift	shift	<< >> >>>
Relational	comparison equality	< > <=>= instanceof == !=
Bitwise	Bitwise AND bitwise exclusive OR Bitwise inclusive OR	& ^ |
Logical	Logical AND Logical OR	&& ||
Ternary	Ternary	? :
Assignment	Assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Category Of Operators:
Unary The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.: Incrementing/decrementing a value by one

Binary: A binary operator is an operator ,which operates on one operands(y+v).

Ternary : A ternary operator is an operator,which operates on three operands.

Q.34) What are the Types of Operators available in Java:

1.Arithmetic Operator:-

- Operators constitute the basic building block to any programming language, they are classified based on the functionality they provide.
- Java too provides many types of operators which can be used according to the need to perform various calculations and functions.
- One of them is Arithmatic Operator,these operators involve the mathematical operators that can be used to perform various simple or advanced arithmetic operations on the primitive data types.

- These operators consist of various unary and binary operators that can be applied on a single or two operands.
- Now lets Look at each one of the arithmetic operators in Java:-

1. **Addition(+):** This operator is a binary operator and is used to add two operands.

   Syntax:

   num1 + num2

   Example:

   num1 = 10, num2 = 20

sum = num1 + num2 = 30

2. **Subtraction(-):** This operator is a binary operator and is used to subtract two operands.

Syntax:

num1 - num2

Example:

num1 = 20, num2 = 10sub = num1 - num2 = 10

3. **Multiplication(*):** This operator is a binary operator and is used to multiply two operands.

Syntax:

num1 * num2

Example:

num1 = 20, num2 = 10mult = num1 * num2 = 200

4. **Division(/):** This is a binary operator that is used to divide the first operand(dividend) by the second operand(divisor) and give the quotient as a result.

Syntax:

num1 / num2

Example:

num1 = 20, num2 = 10div = num1 / num2 = 2

5. **Modulus(%):** This is a binary operator that is used to return the remainder when the first operand(dividend) is divided by the second operand(divisor).

Syntax:

num1 % num2

Example:

num1 = 5, num2 = 2mod = num1 % num2 = 1. Assignment Operator:-• Operators constitute the basic building block to any programming language. • These operators are used to assign values to a variable. • The left side operand of the assignment operator is a variable, and the right side operand of the assignment operator is a value.• The value on the right side must be of the same data type of the operand on the left side. Otherwise, the compiler will raise an error. • Therefore, the right-hand side value must be declared before using it or should be a constant.

- The Assignment Operator is generally of two types. They are:
- **1. Simple Assignment Operator:** The Simple Assignment Operator is used with the "=" sign where the left side consists of the operand and the right side consists of a value. The value of the right side must be of the same data type that has been defined on the left side.
- **2. Compound Assignment Operator:** The Compound Operator is used where +,-,*, and / is used along with the = operator.
- Let's look at each of the assignment operators and how they operate:

1. (=) operator:

This is the most straightforward assignment operator, which is used to assign the value on the right to the variable on the left. This is the basic definition of an assignment operator and how it functions.

Syntax:

num1 = num2;

Example:

a = 10;

ch = 'y';

2. (+=) operator:

This operator is a compound of ‘+’ and ‘=’ operators. It operates by adding the current value of the variable on the left to the value on the right and then assigning the result to the operand on the left.

Syntax:

num1 += num2;

Example:

a += 10

3. (-=) operator:

This operator is a compound of ‘-‘ and ‘=’ operators. It operates by subtracting the variable’s value on the right from the current value of the variable on the left and then assigning the result to the operand on the left.

Syntax:

num1 -= num2;

Example:

a -= 10 This means,a = a - 10

4. (*=) operator:

This operator is a compound of ‘*’ and ‘=’ operators. It operates by multiplying the current value of the variable on the left to the value on the right and then assigning the result to the operand on the left.

Syntax:

num1 *= num2;

Example:

a *= 10This means,a = a * 10

5. (/=) operator:

This operator is a compound of ‘/’ and ‘=’ operators. It operates by dividing the current value of the variable on the left by the value on the right and then assigning the quotient to the operand on the left.

Syntax:

num1 /= num2;

Example:

a /= 10This means,a = a / 10

6. (%=) operator:

This operator is a compound of '%' and '=' operators. It operates by dividing the current value of the variable on the left by the value on the right and then assigning the remainder to the operand on the left.

Syntax:

num1 %= num2;

Example:

a %= 3 This means,a = a % 3

Relational Operator:-

- Operators constitute the basic building block to any programming language.
    - **Java Relational Operators** are a bunch of binary operators used to check for relations between two operands, including equality, greater than, less than, etc.
    - They return a boolean result after the comparison and are extensively used in looping statements as well as conditional if-else statements and so on.
    - The general format of representing relational operator is:

Syntax:

variable1 *relation_operator* variable2

Operator 1: 'Equal to' operator (==)

This operator is used to check whether the two given operands are equal or not. The operator returns true if the operand at the left-hand side is equal to the right-hand side, else false.

Syntax:

var1 == var2

Operator 2: ‘Not equal to’ Operator(!=)

This operator is used to check whether the two given operands are equal or not. It functions opposite to that of the equal-to-operator. It returns true if the operand at the left-hand side is not equal to the right-hand side, else false.

Syntax:

var1 != var2

Operator 3: ‘Greater than’ operator(>)

This checks whether the first operand is greater than the second operand or not. The operator returns true when the operand at the left-hand side is greater than the right-hand side.

Syntax:

var1 > var2

Operator 4: ‘Less than’ Operator(<)

This checks whether the first operand is less than the second operand or not. The operator returns true when the operand at the left-hand side is less than the right-hand side. It functions opposite to that of the greater-than operator.

Syntax:

var1 < var2

Operator 5: Greater than or equal to (>=)

This checks whether the first operand is greater than or equal to the second operand or not. The operator returns true when the operand at the left-hand side is greater than or equal to the right-hand side.

Syntax:

var1 >= var2

Operator 6: Less than or equal to (<=)

This checks whether the first operand is less than or equal to the second operand or not. The operator returns true when the operand at the left-hand side is less than or equal to the right-hand side.

Syntax:

var1 <= var2 Increment & Decrement Operator:-

- **1) Increment Operators:** The increment operator is used to increment the value of a variable in an expression. In the Pre-Increment, the value is first incremented and then used inside the expression. Whereas in the Post-Increment, the value is first used inside the expression and then incremented.
- **Syntax:**

// PREFIX++m // POSTFIXm++ where m is a variable.

- **2) Decrement Operators:** The decrement operator is used to decrement the value of a variable in an expression. In the Pre-Decrement, the value is first decremented and then used inside the expression. Whereas in the Post-Decrement, the value is first used inside the expression and then decremented.

Syntax:

// PREFIX--m // POSTFIXm-- where m is a variable.

Q.35) Differences between Increment And Decrement Operators

Increment Operator	Decrement Operator
• Increment Operator adds 1 to the operand.	• Decrement Operator subtracts 1 from the operand.
• Postfix increment operator means the expression is evaluated first using the original value of the variable and then the variable is incremented(increased).	• Postfix decrement operator means the expression is evaluated first using the original value of the variable and then the variable is decremented(decreased).

- Prefix increment operator means the variable is incremented first and then the expression is evaluated using the new value of the variable.
- Generally, we use this in decision-making and looping.
- Prefix decrement operator means the variable is decremented first and then the expression is evaluated using the new value of the variable.
- This is also used in decision-making and looping.

Q.36) What are Logical Operators .

1.Logical operators are used to performing logical “AND”, “OR” and “NOT” operations, i.e. the function similar to AND gate and OR gate in digital electronics.

2. They are used to combine two or more conditions/constraints or to complement the evaluation of the original condition under particular consideration.

3.One thing to keep in mind is the second condition is not evaluated if the first one is false, i.e. it has a short-circuiting effect. Used extensively to test for several conditions for making a decision.

1.**AND Operator** (**&&**) – if(a && b) [if true execute else don’t]

2.**OR Operator** (**||**) – if(a || b) [if one of them is true execute else don’t]

3.**NOT Operator** (**!**) – !(a<b) [returns false if a is smaller than b]

Example For Logical Operator in Java

Here is an example depicting all the operators where the values of variables a, b, and c are kept the same for all the situations.

a = 10, b = 20, c = 30 For AND operator:

Condition 1: c > a

Condition 2: c > b

Output: True [Both Conditions are true] For OR Operator:

Condition 1: c > a

Condition 2: c > b

Output: True [One of the Condition if true]

For NOT Operator:

Condition 1: c > a

Condition 2: c > b

Output: False [Because the result was true and NOT operator did it's opposite]

Q.37) Describe Bitwise Operators.

The bitwise operators are the operators used to perform the operations on the data at the bit-level. When we perform the bitwise operations, then it is also known as bitlevel programming. It consists of two digits, either 0 or 1. It is mainly used in numerical computations to make the calculations faster.

We have different types of bitwise operators in the C programming language. The following is the list of the bitwise operators:

Operator	Meaning of operator
&	Bitwise AND operator
|	Bitwise OR operator
^	Bitwise exclusive OR operator
~	One's complement operator (unary operator)
<<	Left shift operator
>>	Right shift operator

the following 6 operators are bitwise operators (work at bit-level)

1. The & (bitwise AND) in C or C++ takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
2. The | (bitwise OR) in C or C++ takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.
3. The ^ (bitwise XOR) in C or C++ takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.
4. The << (left shift) in C or C++ takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.
5. The >> (right shift) in C or C++ takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
6. The ~ (bitwise NOT) in C or C++ takes one number and inverts all bits of it.

Q.38) Describe Conditional Operators

The conditional (ternary) operator is the only JavaScript operator that takes three operands: a condition followed by a question mark (?), then an expression to execute if the condition is truthy followed by a colon (:), and finally the expression to execute if the condition is falsy. This operator is frequently used as an alternative to an if...else statement.

Syntax
expression ? expression : expression

The conditional operator (? :) is a ternary operator (it takes three operands). The conditional operator works as follows:

1. The first operand is implicitly converted to bool. It is evaluated and all side effects are completed before continuing.

2. If the first operand evaluates to true (1), the second operand is evaluated.
3. If the first operand evaluates to false (0), the third operand is evaluated. The result of the conditional operator is the result of whichever operand is evaluated — the second or the third. Only one of the last two operands is evaluated in a conditional expression.

   Conditional expressions have right-to-left associativity. The first operand must be of integral or pointer type. The following rules apply to the second and third operands:
4. If both operands are of the same type, the result is of that type.
5. If both operands are of arithmetic or enumeration types, the usual arithmetic conversions (covered in Standard Conversions) are performed to convert them to a common type.
6. If both operands are of pointer types or if one is a pointer type and the other is a constant expression that evaluates to 0, pointer conversions are performed to convert them to a common type.
7. If both operands are of reference types, reference conversions are performed to convert them to a common type.
8. If both operands are of type void, the common type is type void.
9. If both operands are of the same user-defined type, the common type is that type.

If the operands have different types and at least one of the operands has userdefined type then the language rules are used to determine the common type.