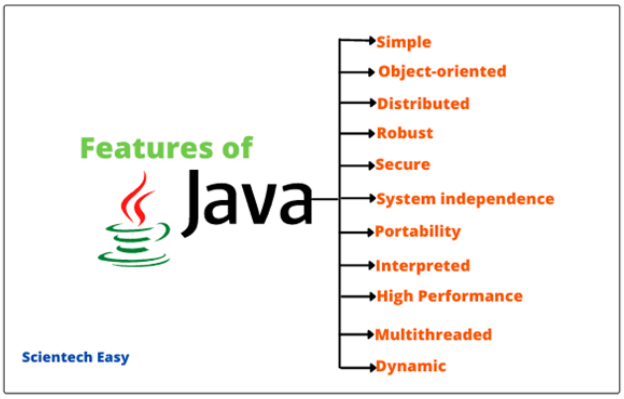
UNIT – I SYLLABUS

JAVA Evolution: History – Features, Overview of Java Language: Introduction - Simple Java program - Structure - Java tokens - Statements - Java virtual Machine. Constants - Variables - Data types - Operators and expressions.

UNIT –I

1.Describe the features of java. BT1

Java is a high-level, object-oriented programming language. This language is very easy to learn and widely used. It is known for its platform independence, reliability, and security. It follows one principle, that is "Write Once, Run Anywhere" principle. It supports various features like portability, robustness, simplicity, multithreading, and high performance, which makes it a popular choice for beginners as well as for developers.



Simple:

- Java is very easy to learn. It inherits many features from C, C++ and removes complex features like pointers, operator overloading, multiple inheritance, explicit memory allocation etc. It provides automatic garbage collection. With a rich set of libraries with thousands of useful functions, Java makes developers life easy.

Object-Oriented:

- Everything in Java is treated as an object, making it easier to model real-world entities and their interactions.
- Java is fundamentally based on the OOP paradigm. This means it organizes code around objects, which encapsulate data (attributes) and methods (functions) that operate on that data.
- Key OOP concepts like encapsulation, inheritance, and polymorphism are central to Java's design, promoting code reusability, modularity, and maintainability.

Distributed:

- Java has built-in support for distributed computing, allowing applications to be developed and deployed across multiple machines on a network.

- Features like Remote Method Invocation (RMI) and support for web services facilitate communication and data exchange between distributed components.

Secured and Robust:

- Java is a reliable programming language because it can catch mistakes early while writing the code and also keeps checking for errors when the program is running. It also has a feature called exception handling that helps deal with unexpected problems smoothly**.**

Platform independent:

- Java is platform-independent because of Java Virtual Machine (JVM).
- When we write Java code, it is first compiled by the compiler and then converted into bytecode (which is platform-independent).
- This byte code can run on any platform which has JVM installed.

Portable:

- When we write a Java program, the code first get converted into bytecode and this bytecode does not depend on any operating system or any specific computer. We can simply execute this bytecode on any platform with the help of JVM. Since JVMs are available on most devices and that's why we can run the same Java program on different platform.

Interpreted:

- Java code is not directly executed by the computer. It is first compiled into bytecode. This byte code is then understand by the JVM. This enables Java to run on any platform without rewriting code.

High Performance:

- Java is faster than old interpreted languages. Java program is first converted into bytecode which is faster than interpreted code. It is slower than fully compiled languages like C or C++ because of interpretation and JIT compilation process. Java performance is improve with the help of Just-In-Time (JIT) compilation, which makes it faster than many interpreted languages but not as fast as fully compiled languages.

Multithreading:

- Multithreading in Java allows multiple threads to run at the same time.
- It improves CPU utilization and enhancing performance in applications that require concurrent task execution.
- Multithreading is especially important for interactive and high-performance applications, such as games and real-time systems.

Dynamic:

- Java is a dynamic language, meaning it can adapt to changing environments during runtime.
- Features like dynamic class loading allow new classes to be loaded into a running program as needed, providing flexibility and adaptability.

2.What is an Operator? List the various operators supported by java explain.

Java operators are special symbols that perform operations on variables or values**.** These operators are essential in programming as they allow you to manipulate data efficiently. They can be classified into different categories based on their functionality.

Types of Operators in Java

1.Arithmetic Operators

2.Assignment Operator

3.Relational Operators

4.Logical Operators

5.Ternary /Conditional Operator

6.Bitwise Operators

1. Arithmetic Operators:

These operators involve the mathematical operators that can be used to perform various simple or advanced arithmetic operations on the primitive data types referred to as the operands. These operators consist of various unary and binary operators that can be applied on a single or two operands.

Operators	**Result**
+	Addition of two numbers
-	Subtraction of two numbers
*	Multiplication of two numbers
/	Division of two numbers
%	Divides two numbers and returns the remainder

2. Assignment Operators

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 given on the right-hand side of the operator is assigned to the variable on the left. Therefore, the right-hand side value must be declared before using it.

The general format of the assignment operator is:

Variable=value;

3. Relational Operator:

Relational Operators are used to check for relations like equality, greater than, and less than. They return Boolean results after the comparison and are extensively used in looping statements as well as conditional if-else statements.

Relational operators compare values and return Boolean results:

== (Equal to).

!= (Not equal to).

< (Less than).

<=(Less than or equal to).

>(Greater than).

>=(Greater than or equal to).

4. Logical Operators:

Logical Operator are used to perform "logical AND" and "logical OR" operations, similar to AND gate and OR gate in digital electronics. They are used to combine two or more conditions/constraints or to complement the evaluation of the original condition under particular consideration.

- AND Operator (&&): If(a && b) [if true execute else don't]
- OR Operator (||): If(a || b) [if one of them is true to execute; else don't]
- NOT Operator (!): !(a<b) [returns false if a is smaller than b]

Logical AND Operator (&&): -

This operator returns true when both the conditions under consideration are satisfied or are true. If even one of the two yields false, the operator results false.

Logical OR Operator (||): -

This operator returns true when one of the two conditions under consideration is satisfied or is true. If even one of the two yields true, the operator results true.

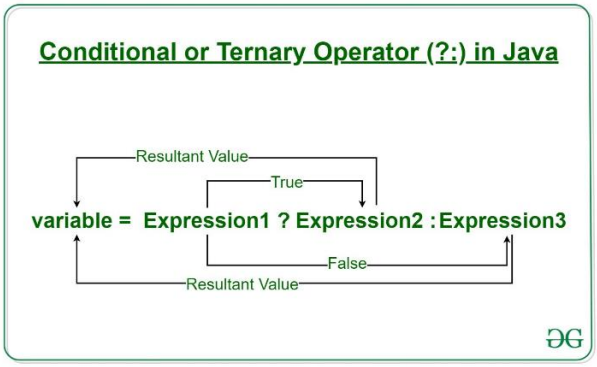
Logical NOT Operator (!): -

Unlike the previous two, this is a unary operator and if the condition is false, the operation returns true and when the condition is true, the operation returns false.

5. Ternary Operator:

Java ternary operator is the only conditional operator that takes three operands. It’s a one-liner replacement for the if-then-else statement and is used a lot in Java programming. We can use the ternary operator in place of if-else conditions or even switch conditions using nested ternary operators.

Ternary Operator improves the code readability, but for complex logic, a standard if-else statement might be better.



6.Bitwise Operators:

Bitwise Operators are used to perform the manipulation of individual bits of a number and with any of the integer types. They are used when performing update and query operations of the Binary indexed trees.

- & (Bitwise AND): returns bit-by-bit AND of input values.
- | (Bitwise OR): returns bit-by-bit OR of input values.
- ^ (Bitwise XOR): returns bit-by-bit XOR of input values.

Bitwise AND (&): -

This operator is a binary operator, denoted by '&.' It returns bit by bit AND of input values, i.e., if both bits are 1, it gives 1, else it shows 0.

Bitwise OR (|): -

This operator is a binary operator, denoted by '|'. It returns bit by bit OR of input values, i.e., if either of the bits is 1, it gives 1, else it shows 0.

Bitwise XOR (^): -

This operator is a binary operator, denoted by '^.' It returns bit by bit XOR of input values, i.e., if corresponding bits are different, it gives 1, else it shows 0.

3. Explain History of Java and write a simple java program?

Java is an Object-Oriented programming language developed by James Gosling in the early 1990s. The team initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc.

Firstly, the java was called “Green talk” by James Gosling and his team and file extension was .gt and later became to known as “OAK”.

OAK was designed for programming interactive TV systems and embedded devices. Due to trademark issues, it was later renamed Java.

In 1995, Sun Microsystems officially launched Java, introducing its "Write Once, Run Anywhere" (WORA) capability through the Java Virtual Machine (JVM).

Evolution and Web Integration:

1994: The team recognized Java's potential for the burgeoning World Wide Web.

Platform Independence: They realized the need for a language that could run on any operating system, leading to Java's development.

"Write Once, Run Anywhere": This principle became a core tenet of Java, enabled by the JVM.

Official Launch: Java was officially launched by Sun Microsystems in 1995.

Key Characteristics:

- **Object-Oriented Programming:**

Java is an object-oriented language, promoting code reusability and organization.

- **Platform Independence:**

The JVM allows Java code to run on any platform with a compatible JVM.

- **Security Features:**

Java was designed with security in mind, particularly for network environments.

- **Rich Ecosystem:**

Java has a vast ecosystem of libraries, frameworks, and tools for various applications.

Java Program Structure:

1.A Java program may contain many classes of which only one class defines a main method.

2.Classes contain data members and methods that operate on the data members of the class. Methods may contain data type declarations and executable statements.

3.To write a Java program, we first define classes and then put them together.

Documentation Section

Package Statement

Import Statement

Interface Section

Class Definition

Main class

{

Main method ()

{

}}

Documentation Section:

1.This section consists of commenting lines. The comments can be used anywhere in the program.

2.Generally, the comment section includes the documentation of the java program. Comments can help us to understand the code to non-programmers.

3.In general, comments include the following data.

a. Name of the program

b. Purpose of the program

c. Author of the program

d. Copy right information

4.Java supports different kinds of comments.

a. Single line comments (//): ex: // This is my first java program.

b. Multi line comment: If comment runs over a number of lines we use multi line comment (/* …. */) Ex: /* This program was developed at Aditya Degree College

By Aditya students */

c. Java document comment (/** … */)

Package statements:

1.The first line allowed in java program is package section. It contains package name.

2.Package name informs the compiler that all the classes are belonging to this package.

3.General Syntax: package package_name;

4.Package statement is optional when our classes must not have to be the part of the package.

Import Statement:

1.This statement is used to add a specific class or classes of a package to the current java program. Ex: import java.lang.System;

2.The above statement imports the class “System” from java.lang package to our program. Ex: import java.lang.*;

3.The above statement imports all the classes form the package java.lang to our program. The import statement instructs the interpreter to load the classes from the specified package.

Interface section:

1.The interface section is exactly similar to the class but it includes a group of method declarations.

2.It is used only when we are implementing “Multiple Inheritance”.

Class Definition Section:

1.A java program may contain any number of class definitions.

2.Classes are primary and essential components of a java program.

3.A class is used to create user-defined data-types.

4.Every class contains a set of properties and methods that operate on the properties.

Main Class Definition:

1.Any java program contains a main method, and it is the starting point of execution of the program.

2.Main() method creates the objects of various classes and established communication between them.

3.Once controller reaches the boundary end of main program, the program terminates and the controller pass back to operating system.

Simple Java Program:

```
Class Sample{

public static void main(String[]args)

{

System.out.println("Hello World");

}

}
```

4. Describe Variable and explain different types of variables?

Variables:

1.A variable is an entity whose value may vary during program execution.

2.A variable is an identifier that can be used to store values. Variables store different values but one at a time.

3.A data-type is always associated with a variable. The data-type of the variable decides what types of value it can store.

Variable Declaration

We must declare a variable before it is used in program. The declaration of variable contains data-type which indicates what type of value it is going to be contained and what is the range of values.

Syntax: - datatype variablename; //variable declaration

Initialization of a Variable:

When a variable is declared, it commonly contains some default values. We can assign some initial values to the variable while declaration, it is called as initialization.

Ex: int a=10; variable initialization

Scope of Variables:

Depending on the place of declaration of a variable, the variable holds some properties like life time.

There are three types of variables.

1.Local variables:

a. The variables that are declared and used inside the method are called as local variables. These can be accessed from starting of the method to ending of the method.

b. The value of local variable is not visible to outside of the method. Local variables can also declare inside a block.

c. When the controller comes out of the block or method, the value of local variables exits.

2.Instance variables:

a. These variables are declared inside a class. Instance variables are created when the objects are instantiated and then they are associated with objects.

b. They take different values for each object. The memory is not common for all the objects, memory to each object will be individually allocated.

c. Instance variables should not be preceded with the keyword “static”.

3.Static Variables:

a. If a variable is declared inside the class and it is shared by all the objects of that class then that variables are referred to as “static” variables.

b. These are called as class variables.

c. The keyword “static” is used to declare static variables. Ex: static int a;

d. Common memory is allocated for static variable for all the objects and this object share the value of the static variable.

e. The static variables are called with class name not with object.

DATA TYPE:

1.Data type is used to specifies the type of storing values in a variable. Java supports different kinds of data types.

2.For storing value in a variable, each variable in java must have a data type.

3.Data type specifies the size and the range of values that can be stored in a variable.

4.The following are the list of data types available in java.

Memory Efficiency: Choosing the right type (byte vs int) saves memory.

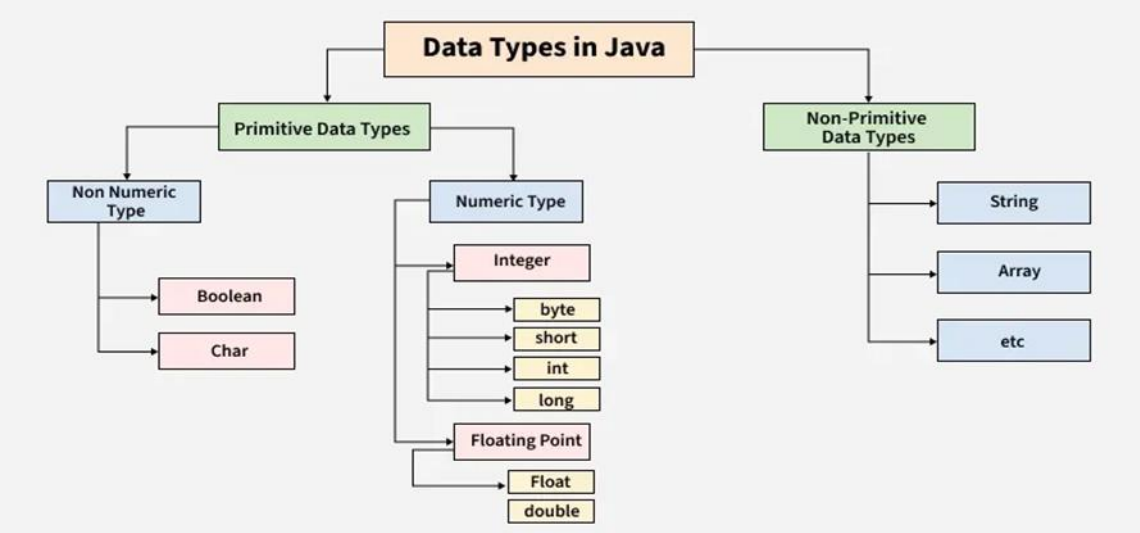
Performance: Proper types reduce runtime errors.

Code Clarity: Explicit typing makes code more readable.

Java has two categories

1.Primitive data type

2.Non-Primitive data type



1. Primitive Data Type: Primitive data types are predefined by the language and named by a keyword. There are eight primitive data types supported by Java. Below is the list of the primitive data types

- byte
- short
- int
- long
- float
- double
- Boolean
- Char

byte Data Type

The byte data type is an 8-bit signed two's complement integer with a minimum value of -128 and a maximum value of 127.

The default value of a byte variable is 0, which is used to save space in large arrays, which is mainly beneficial in integers since a byte is four times smaller than an integer.

EX: byte a=100;

Byte b=-50;

Short Data type:

The short data type is a 16-bit signed two's complement integer, which provides a range of values from -32,768 to 32,767.

The default value for a short variable is 0.

EX: short s=10000;

short s=-20000;

int Data Type:

The int data type is a 32-bit signed two's complement integer, allowing for a wide range of values from -2,147,483,648 to 2,147,483,647 .

The default value for an int variable is 0.

EX: int a = 100000;

int b = -200000;

long Data Type

The long data type is a 64-bit signed two's complement integer, capable of representing a vast range of values from -9,223,372,036,854,775,808 (-263) to 9,223,372,036,854,775,807 (inclusive) (263 -1). This data type is used when a wider range than int is needed, where its default value is 0L.

EX: long a = 100000L;

long b = -200000L;

float Data type:

The **float** data type is a single-precision 32-bit IEEE 754 floating-point representation. It is particularly useful for saving memory in large arrays of floating-point numbers. Its default value is 0.0f.

EX: float f1 = 234.5f;

double Data Type

The **double** data type is a double-precision 64-bit IEEE 754 floating-point representation, which is generally used as the default data type for decimal values.where its default value is 0.0d.

Example

double d1 = 123.4;

boolean Data Type

The boolean data type represents a single bit of information and can hold one of two possible values: true or false. This data type is used for simple flags that track true/false conditions where its default value is ***false***.

Example

boolean one = true;

char Data Type

The **char** data type is a single 16-bit Unicode character, which represents a wide range of characters from different languages and symbols. With a range '**\u0000**' (or **0**) to '**\uffff**' (or **65,535** inclusive). This data type is primarily used to store individual characters.

Example:

char letterA = 'A';

The default value for char is “null”.

2. Non-Primitive Data Types (Object Types): The non-primitive data types are not predefined. The reference data types are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed. For example, Employee, Puppy, etc.

String: The string is a class in Java, and it represents the sequences of characters**.**

Arrays: Arrays are created with the help of primitive data types and store multiple values of the same type.

Classes: The classes are the user-defined data types and consist of variables and methods.

Interfaces: The interfaces are abstract types that are used to specify a set of methods.

The default value of any reference variable is null.

Q. Explain how to implement a java Program?

Implementing a Java program involves three steps.

1.Creating a Java Program

2.Compiling a Java Program

3.Running a Java Program

1**.Creating a Java Program:**

Creating a java program means writing a java program in any one of the text editors like notepad. Before writing a java program, first we need to check whether the java software installed or not.

If installed, then you can start writing the java program. Ex: import java.lang.*;

class FirstProgram

{

Public static void main(String args[])

{

System.out.println("Welcome to Java World");

}

}

Save the above program with the name FirstProgram.java.

2.Compiling a Java Program:

By using java compiler we can convert the source program into byte code. This byte code is generated in any machine. This byte code is system dependent.

General Syntax: C:\>javac FileName.java

C:\>javac FirstProgram.java

It will generate dot class file, i.e., FirstProgram.class.

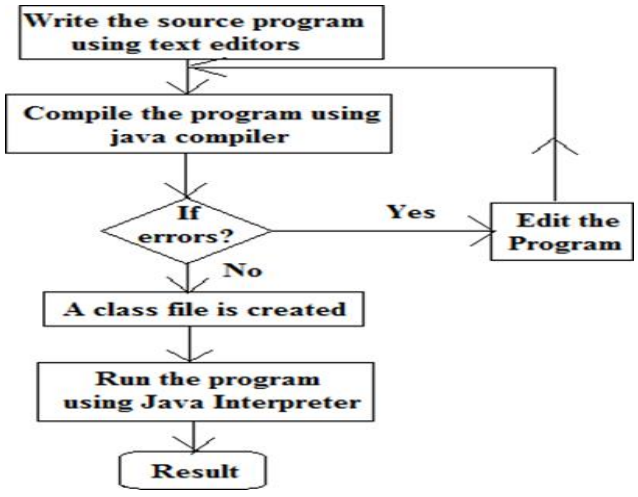
3.Running a Java Program:

The byte code is not directly executed by any machine. The java interpreter can take the byte code and generate machine executable instructions.

General Syntax: C:\>java ClassName

C:\>java FirstProgram

The following flow chart represents a sequence of steps involved in implementing a Java program



What is a Token? List various types of tokens supported by java?

Java Tokens:

The smallest individual unit in a language is called tokens. The compiler recognizes the tokens for building up expressions and statements. There are five major categories of Java tokens available in Java Programming.

Keywords:

- These are also called as reserved words of the language.
- Keywords have a specific meaning.
- We cannot use the keywords as identifiers.
- All keywords are written in lower case letters only.

In Java, we have more than 50 keywords.

abstract	Assert	boolean	Break	byte	case	catch
Char	Class	const	continue	default	do	double
Else	Enum	extends	Final	finally	float	for
Goto	If	implements	Import	instanceof	int	interface
Long	Native	new	Package	private	protected	public
Return	Short	static	Strictfp	super	switch	synchronized
This	Throw	throws	transient	try	void	volatile
while						

Identifiers:

- These are the programmer designed tokens.
- Those are used for giving names to variables, names to classes, names to packages, names to interfaces and to programs.
- There are certain rules and conventions, we must follow while using identifiers.

Rules for Identifiers:

- They have alphabets, digits and an underscore(_).
- It must begin with alphabets and followed by digits.
- It must not be a keyword.
- Other than the underscore no special symbols are permitted.
- Spaces should not be allowed in variable names.
- Upper case letters defer from lower case letters.

Conventions for Identifiers:

- Class names begin with upper case alphabet.
- If method name contains two words, we merge them by capitalizing the first letter of second word.

  Ex: getData( )

Literals:

- Literals are the sequence of characters that represent constant value i.e., to be stored in a variable.
- Java supports five categories of Literals.

  1.Integer Literals (25, +10, -23,…)

  2.Float Literals (3.4, -3.4,….)

  3.Character Literals („A‟, „a‟,…..)

  4.String Literals (“Moon”, JAVA”,….)

  5.Boolean Literals (true,false)

Operators:

- An operator is a symbol that is used to manipulate Data.
- An operator takes one (or) more inputs (operands) and produces result by operating them.
- If the operator takes only one operand that is called unary operator (+a,-a,…).
- If the operator takes two operands then it is called as binary operator(a+b, c=a/b,…).

Separator:

- These are the symbols used to indicate where the group of code is divided and arranged.
- The following are the distinct separators.

  ; → Used to separate statements or terminates the statement.

  ,→Used to separate continues identifiers in variable declaration.

  . →Used to separate package names from classes and also used to separate the variable names with objects.

  [ ]→Used declare an array and specifying the size.

  { }→Used to indicate starting of a class, interface, method and also used to initialize the values for an array.

  ( )→Used in method definition and calling of methods.

Explain about java Statements.

1.**Empty Statement**:- These do nothing and are used during program development as a place holder.

2.**Labeled Statement:-** Any statement may begin with a label. Such labels must not be keywords.

3.**Expression Statements:-** Most of the java statements are expression statements. Java has 7 types of expression statements.

4.**Selection Statements**:- These are used to select one of several control flows. There are three types of selection statements in Java.

5.**Iterative Statement**:- These specify how and when looping will take place. There are three types of iterative statements in Java.

6.**Jump Statements**:- Jump statements pass control to the beginning or end of the current block, or to a labeled statement.

7.**Synchronization Statement**:- These are used for handling issues with multithreading.

8.**Guarding Statements**:- Guarding statements are used for safe handling of code that may cause exceptions.

Q.What are Constants? Explain about different types of constants?

In Java, a constant refers to a value or a variable whose value cannot be changed once it has been initialized.

Constants are declared using the final keyword. This keyword signifies that the variable's value can only be assigned once.

A common convention for naming constants is to use all uppercase letters with underscores separating words (e.g., MAX_VALUE).

Constants are used for values that are fixed and should not change, such as mathematical constants (e.g., PI), configuration settings, or fixed limits.

Syntax :

final data_type CONSTANT_NAME = value;

final: The final keyword indicates that the value of the constant cannot be changed once it is initialized.

data_type: The data type of the constant such as int, double, boolean, or String.

CONSTANT_NAME: The name of the constant which should be written in all capital letters with underscores separating words.

value: The initial value of the constant must be of the same data type as the constant.

Java supports various types of constants. They are as follows:

Integer constants

Real constants

Character constants

String constants

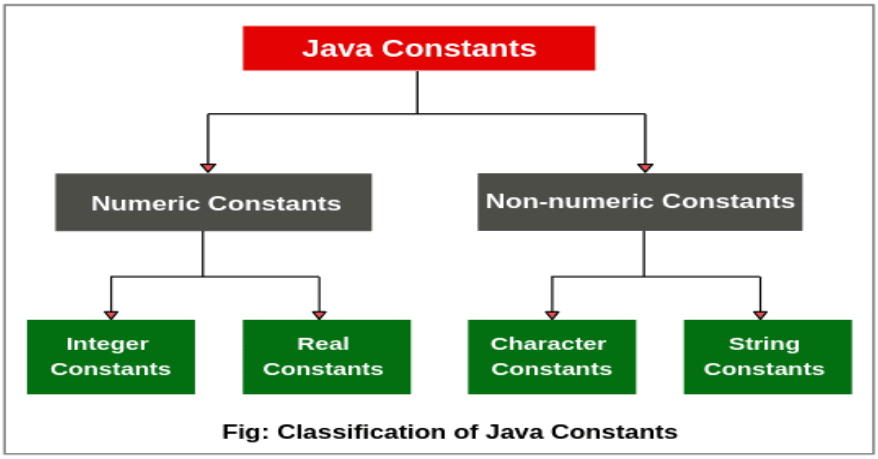


Fig: Classification of Java Constants

Integer Constant:

An integer constant is a sequence of digits without a decimal point. There are three types of integer constants.

- Decimal integer
- Octal integer
- Hexadecimal integer

Decimal Integer:

Decimal integer is a sequence of digits from 0 to 9. These constants are either positive or negative. Plus sign is optional.

Octal Integer:

An octal integer constant is a sequence of any combination of digits from 0 to 7. The octal integer always starts with 0. The digits 8 and 9 are not valid in octal notation.

Hexadecimal Integer:

Hexadecimal integer constants consist of digits 0 to 9 and alphabets “a” to “f” or “A” to “F”. These constants are preceded by 0x. Letter “A” to “F” represents the numbers 10 to 15.

Example program:

```
public class IntegerConst {

public static void main(String[] args)
```

```
{
   int a, b, c;
   a = 20; // decimal notation.
   b = 020; // octal notation.
   c = 0x20F; // hexadecimal notation.
   System.out.println("Decimal notation 20: " +a);
   System.out.println("Octal notation 020: " +b);
   System.out.println("Hexadecimal notation 0x20F: " +c);
  }
}
```

Real constants:

Real constants consist of a sequence of digits with fractional parts or decimal points. These constants are also called floating-point constants in Java.

A real constant number can also be expressed in exponential or scientific notation. For example, the value 235.569 can also be written as 2.35569e2 in exponential notation. Here, e2 means multiply by 10^2.

Single Character Constants

A single character constant (or simply character constant) is a single character enclosed within a pair of single quote.

Ex:'5', 'x', ';', ' ',

String Constants

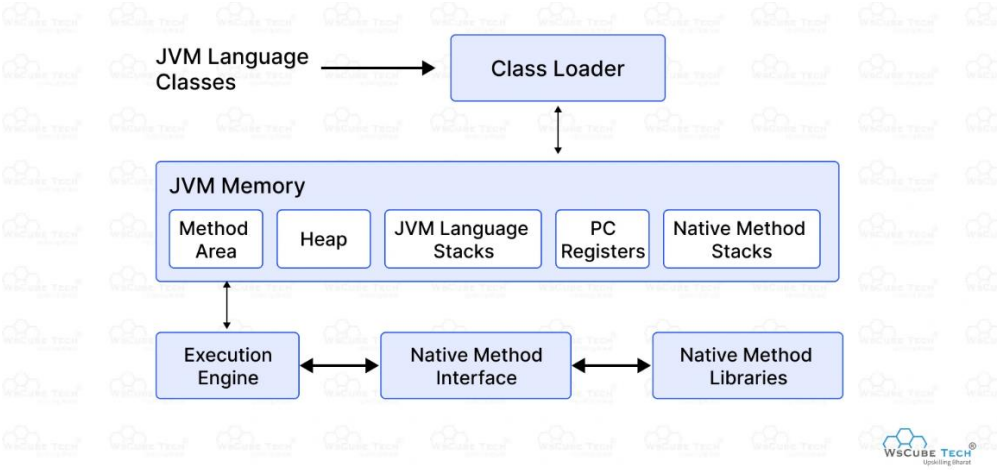
A string constant is a sequence of characters within a pair of double-quotes. The characters can be alphabets, special characters, digits, and blank spaces.

Ex: "Hello Java", "20"

Q. Explain about JVM (Java Virtual Machine).

JVM stands for Java Virtual Machine. It is a part of the Java platform that allows Java programs to run on any device or operating system without rewriting the code. When you write a Java program and compile it, the code is converted into a special format called bytecode.

The JVM reads this bytecode and translates it into machine code that your system understands, making Java platform-independent. Its main purpose is to execute Java bytecode safely and efficiently. It also handles memory management, error checking, and garbage collection.



Components of Java Virtual Machine:

1**. Class Loader**

The Class Loader loads .class files (Java bytecode) into memory when your program runs.

It's like a librarian who fetches only the books (classes) you need at that moment, saving space and time.

2. Bytecode Verifier

This checks that the bytecode is safe and doesn't do anything harmful, like accessing private data or breaking memory rules.

It's like a security guard checking your ID before letting you into a building.

3**. Execution Engine**

The Execution Engine runs the bytecode. It can either interpret the code line by line or use a Just-In-Time (JIT) Compiler to convert it into machine code for faster execution.

Garbage Collector

It automatically finds and removes unused objects from the heap to free up memory.

It is like a cleanup crew that gets rid of junk you no longer use.