JAVA

Java is a **programming language** and a **platform**.

Platform: Any hardware or software environment in which a program runs, is known as a platform.
Since Java has a runtime environment (JRE) and API(Application Programming interface), it is called a platform.

History

James Gosling, Mike Sheridan, and **Patrick Naughton** initiated the Java language project in June 1991 with their small team of sun engineers called **Green Team**.
Initially it was designed for small, embedded systems in electronic appliances like set-top boxes.
Firstly, it was called **"Greentalk"** by James Gosling, and the file extension was .gt

After that, it was called **Oak** and was developed as a part of the Green project.

Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.

In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

Java is an island in Indonesia where the first coffee was produced (called Java coffee).
Fun part: That's the reason for the cup symbol of java language

Java stands for revolutionary, dynamic, lively, cool, unique, and easy to spell, and fun to say.

Applications of Java

- Desktop Applications such as Student Management System, media player, antivirus, etc.
- Web Applications . Example website
- Enterprise Applications such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games, etc.

Types of Java Applications

- Standalone Applications (AWT & SWING)
- Web Application (Servlet, JSP, Hibernate)
- Enterprise (EJB)
- Mobile Application (Android)

Features

- Simple
  - Java has removed explicit pointers , operator overloading etc
  - No need to remove unreferenced objects because there is Automatic Garbage Collection in java
- Object-Oriented
  - Object-Oriented programming is a methodology which simplifies development and maintenance
  - Concepts include Classes and object, Inheritance, Abstraction, Polymorphism and many more
- Portable
  - We can Java code everywhere irrespective of the platform
- Platform-Independent

- Java is platform independent because Java has JVM, JRE & JDK which are dependent on the platform

- Secured
  - No explicit pointer
  - Java Programs run inside a virtual machine sandbox (JVM)

- Robust
  - It uses strong memory management.
  - There is a lack of pointers which avoids security problems.
  - Automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a code.
  - Exception handling and the type checking mechanism in Java.

- Architectural Neutral
  - The size of primitive types is fixed.
  - In C , int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. But it occupies 4 bytes of memory for both 32 and 64-bit architectures in java

- Interpreted
  - Byte Code is interpreted line by line at the time of running

- High Performance
- Multi Threaded
- Distributed
- Dynamic
  - Supports dynamic loading of classes

Simple Java Code and significance

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

1. **class** keyword is used to declare a class in Java.
2. **public** keyword is an access modifier that represents visibility. It means it is visible to all and can be accessed from anywhere.
3. **static** is a keyword. If we declare any method as static, it is known as the static method. The advantage of the static method is that there is no need to create an object to call the static method. The main() method is executed by the JVM, so it doesn't require creating an object to call the main() method. So, it provides memory utilization.
4. **void** is the return type of the method. It means it doesn't return any value to JVM
5. **main** represents the starting point of the program or main is method_name also
6. **String[] args** or **String args[]** is used for command line argument.
7. **System.out.println()** is used to print statement. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class.

In how many ways we can write a Java program?

1) By changing the sequence of the modifiers, method prototype is not changed in Java.

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

2) The subscript notation in the Java array can be used after type, before the variable or after the variable.

- **public static void** main(String[] args)
- **public static void** main(String []args)
- **public static void** main(String args[])

3) You can provide var-args support to the main() method by passing 3 ellipses (dots)

```
public static void main(String... args)
```

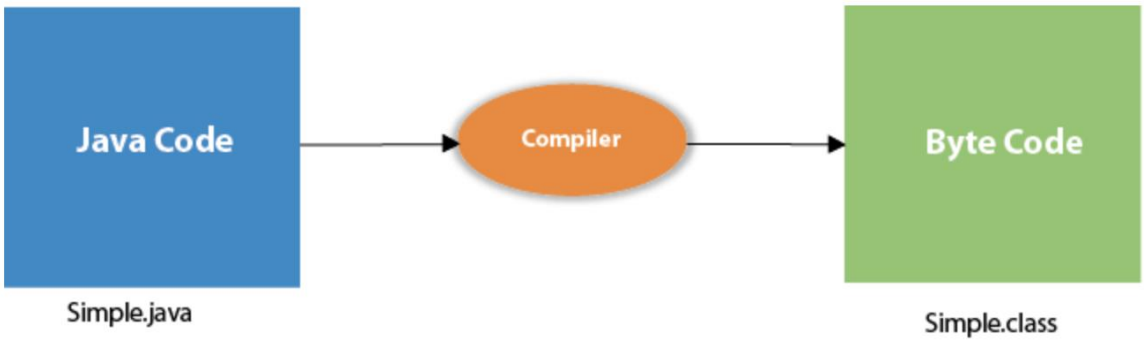
Valid Syntax

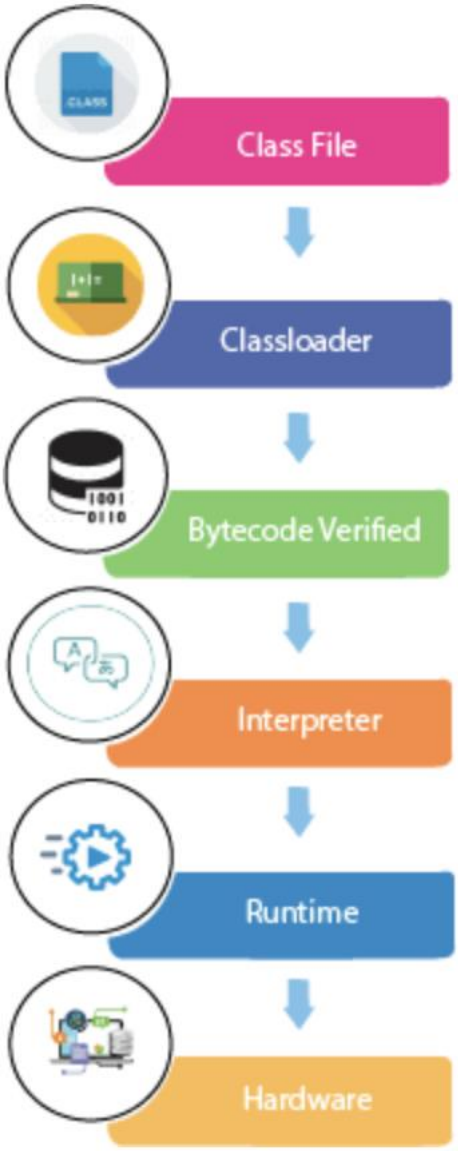
- **public static void** main(String[] args)
- **public static void** main(String []args)
- **public static void** main(String args[])
- **public static void** main(String... args)
- **static public void** main(String[] args)
- **public static final void** main(String[] args)
- **final public static void** main(String[] args)
- **final strictfp public static void** main(String[] args)

Invalid Syntax

1. **public void** main(String[] args) //static keyword is missing
2. **static void** main(String[] args) //public access Specifier is missing

3. **public void static** main(String[] args) //Misplaced static and void
4. **abstract public static void** main(String[] args) //abstract can not be used





JVM (Java Virtual Machine)

JVM (Java Virtual Machine) is an abstract machine.

It is a specification that provides runtime environment in which java bytecode can be executed.

JVM is platform dependent.

What is JVM ??

It is the combination of these three things:

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created. Command : java <Filename>

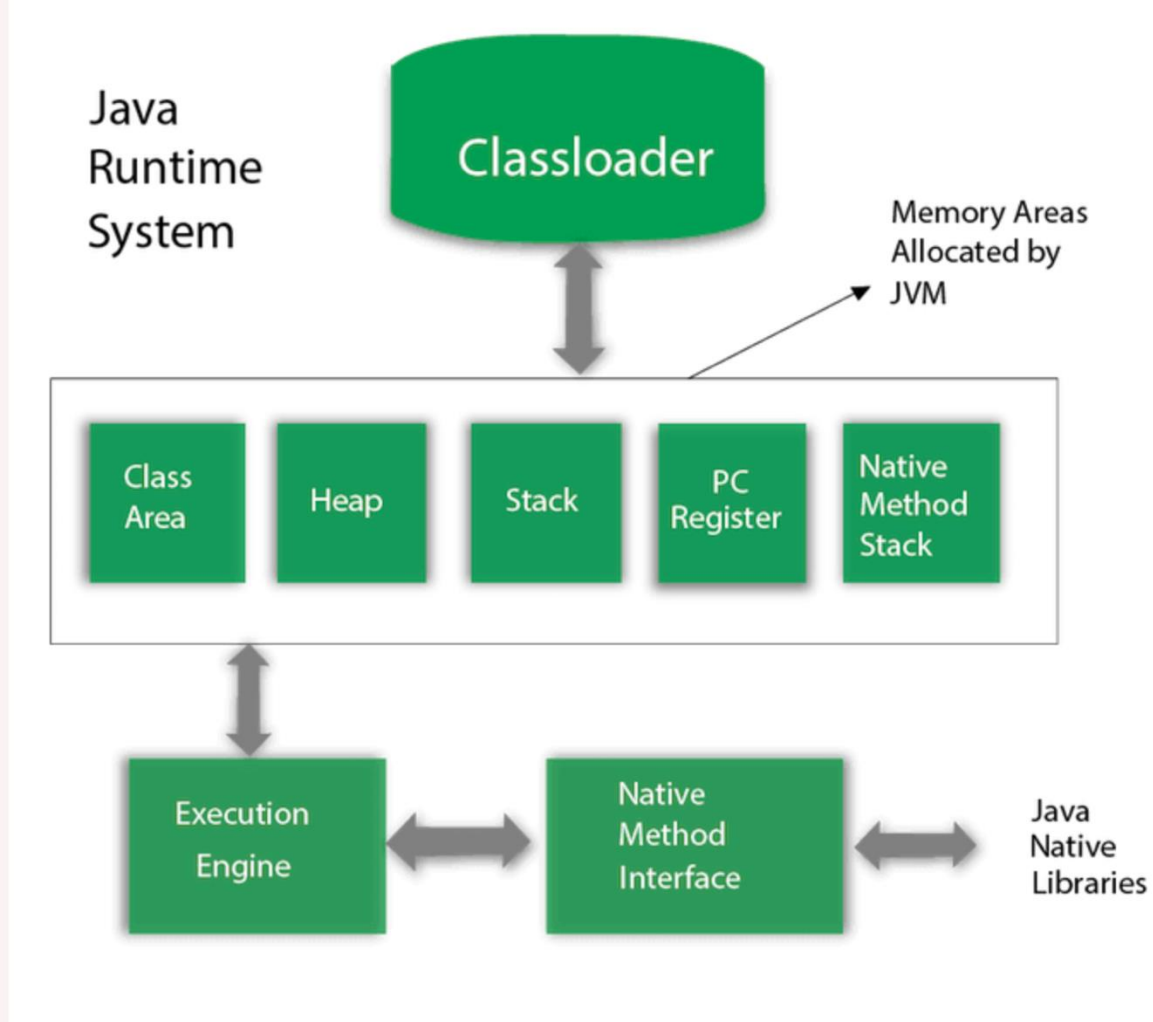
Function of JVM

- Loads code //Class loader
- Verifies code //Bytecode verifier
- Executes code
- Provides runtime environment

JVM Comprises of following things

- Memory area
- Class file format

- Register set
- Garbage-collected heap
- Fatal error reporting etc.



1) Classloader

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader.

There are three built-in classloaders in Java.

1. **Bootstrap ClassLoader**: This is the first classloader which is the super class of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition

java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes etc.

2. **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loads the jar files located inside *$JAVA_HOME/jre/lib/ext* directory.
3. **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the class files from classpath. By default, classpath is set to current directory. It is also known as Application classloader.

2) Class(Method) Area

Class(Method) Area stores runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

It contains:

1. **A virtual processor**
2. **Interpreter**
    - Read bytecode stream then execute the instructions.
3. **Just-In-Time(JIT) compiler**
    - It is used to improve the performance.
    - JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.
    - Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

8) Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc.

Java uses JNI framework to send output to the Console or interact with OS libraries.

JRE (Java Runtime Environment)

- The Java Runtime Environment is a set of software tools which are used for developing Java applications.
- It is used to provide the runtime environment.
- It is the implementation of JVM. It physically exists.
- It contains a set of libraries + other files that JVM uses at runtime.

JDK (Java Development Kit)

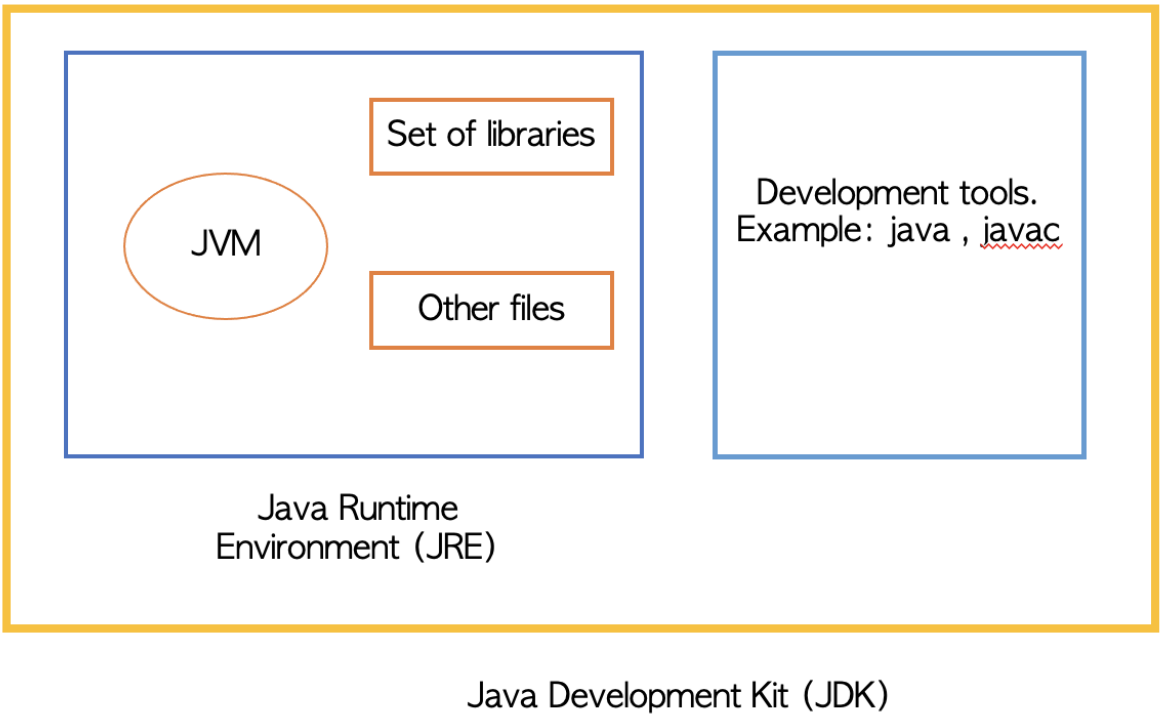
The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets.

It physically exists. It contains JRE + development tools.

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

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

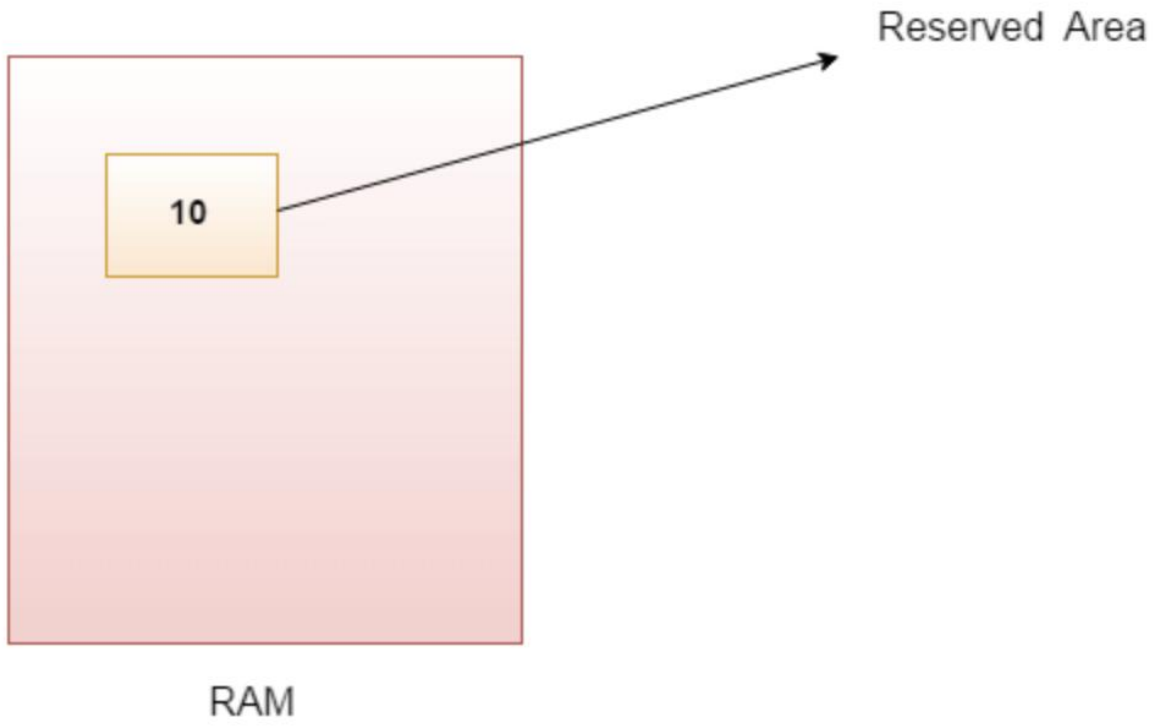


Variables

A variable is a container which holds the value. A variable is assigned with a data type.

Variable is a name of memory location.

A variable is the name of a reserved area allocated in memory. In other words, it is a name of the memory location. It is a combination of "vary + able" which means its value can be changed.



Types of Variables

There are three types of variables

- local variable
- instance variable
- static variable

Local Variable

A variable declared inside the body of the method is called local variable.

You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

Instance Variable

A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static.

It is called an instance variable because its value is instance-specific and is not shared among instances.

Static variable

A variable that is declared as static is called a static variable. It cannot be local.

You can create a single copy of the static variable and share it among all the instances of the class.

Memory allocation for static variables happens only once when the class is loaded in the memory.

Datatype

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive datatype :** The non-primitive data types includes Arrays, String etc.

Datatype	Default Value	Default Size	Range	Example
bool	false	1Bit	-	Boolean value = false
byte	0	1Byte	-128 to 127(Inclusive)	byte a=10,b=-20
short	0	2Byte	-32,768 to 32,767 (inclusive)	**short** s = 10000 **short** r = -5000
int	0	4Byte	-2^31 to 2^31 -1 (inclusive)	**int** a = 100000 **int** b = -200000
long	0L	8Byte	-2^-63 to 2^63-1	long a=100000L
float	0.0f	4Byte	unlimited	float f1=234.5f
double	0.0d	8Byte	unlimited	double d1=12.3
char	'\u0000'	1Byte	16bit Unicode character	char a ='a'

Operators

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

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

- Unary Operator
  - expr++ , ++expr, expr--, --expr, ~, ! , +expr, -expr
- Arithmetic Operator
  - Additive +,-
  - Multiplicative *,/,%
- Shift Operator
  - << , >>, >>>
- Relational Operator
  - Comparison Operator < , > , <= , >= , instanceof,
  - Equality Operator == , !=
- Bitwise Operator
  - BITWISE AND (&)

        - BITWISE OR (|)
        - BITWISE XOR (^)
    - Logical Operator
        - Logical AND (&&)
        - Logical OR (||)
    - Ternary Operator (?:)
    - Assignment Operator (=, += , -= , *= , /= , %=)

Keyword

- Java keywords are also known as reserved words.
- Keywords are particular words that act as a key to a code.
- These are predefined words by Java so they cannot be used as a variable or object name or class name.

A list of Java keywords or reserved words are given below:

- **abstract:** Java abstract keyword is used to declare an abstract class.
- **boolean:** Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.

- **break:** break keyword is used to break the loop or switch statement. It breaks the current flow of the program at specified conditions.
- **byte:** byte keyword is used to declare a variable that can hold 8-bit data values.
- **case:** case keyword is used with the switch statements to mark blocks of text.
- **catch:** Used in exception handling
- **char:** char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters

- **class:** class keyword is used to declare a class.
- **continue:** continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
- **default:** default keyword is used to specify the default block of code in a switch statement.
- **do:** do keyword is used in the control statement to declare a loop. It can iterate a part of the program several times.
- **double:** double keyword is used to declare a variable that can hold 64-bit floating-point number.
- **else:** else keyword is used to indicate the alternative branches in an if statement.
- **enum:** enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.
- **extends:** extends keyword is used to indicate that a class is derived from another class or interface.
- **final:** final keyword is used to indicate that a variable holds a constant value. It is used with a variable. It is used to restrict the user from updating the value of the variable.
- **finally:** used in exception handling
- **float:** float keyword is used to declare a variable that can hold a 32-bit floating-point number.
- **for:** for keyword is used to start a for loop.
- **if:** if keyword tests the condition. It executes the if block if the condition is true.
- **implements:** implements keyword is used to implement an interface.
- **import:** import keyword makes classes and interfaces available and accessible to the current source code.
- **instanceof:** instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.
- **int:** int keyword is used to declare a variable that can hold a 32-bit signed integer.

- **interface:** interface keyword is used to declare an interface. It can have only abstract methods.
- **long:** long keyword is used to declare a variable that can hold a 64-bit integer.
- **native:** native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface).
- **new:** new keyword is used to create new objects.
- **null:** null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.
- **package:** package keyword is used to declare a Java package that includes the classes.
- **private:** private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.
- **protected:** protected keyword is an access modifier. It can be accessible within the package and outside the package but through inheritance only. It can't be applied with the class.
- **public:** public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.
- **return:** return keyword is used to return from a method when its execution is complete.
- **short:** short keyword is used to declare a variable that can hold a 16-bit integer.
- **static:** static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is mainly used for memory management.
- **strictfp:** strictfp is used to restrict the floating-point calculations to ensure portability.
- **super:** super keyword is a reference variable that is used to refer to parent class objects. It can be used to invoke the immediate parent class method.

- **switch:** Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.
- **synchronized:** synchronized keyword is used to specify the critical sections or methods in multithreaded code.
- **this:** this keyword can be used to refer the current object in a method or constructor.
- **throw:** Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exceptions. It is followed by an instance.
- **throws:** Java throws keyword is used to declare an exception. Checked exceptions can be propagated with throws.
- **transient:** transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.
- **try:** try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.
- **void:** void keyword is used to specify that a method does not have a return value.
- **volatile:** volatile keyword is used to indicate that a variable may change asynchronously.
- **while:** while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iteration is not fixed, it is recommended to use the while loop.

Access Modifiers

There are two types of modifiers

- Access modifiers
- Non-access modifiers

The access modifiers specifies the accessibility or scope of a field, method, constructor, or class.

We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc. Here, we are going to learn the access modifiers only.