**ASSIGNMENTS**

1. **List of keywords in java**

Class for if else

Const final finally switch

Double goto extends continue

Break int float short

Long string implements public

Private protected default abstract

This throw Throws throwable

Try Catch do while.

1. **Important classes in java.lang package.**

**Boolean**: The Boolean class wraps a value of the primitive type boolean in an object.

**Byte:** The Byte class wraps a value of primitive type byte in an object.

**Character** – Set 1, Set 2: The Character class wraps a value of the primitive type char in an object.

**Character.Subset**: Instances of this class represent particular subsets of the Unicode character set.

**Character.UnicodeBlock:**

**Class** – Set 1, Set 2 : Instances of the class Class represent classes and interfaces in a running Java application.

**ClassLoader**: A class loader is an object that is responsible for loading classes.

**ClassValue**: Lazily associate a computed value with (potentially) every type.

**Compiler**: The Compiler class is provided to support Java-to-native-code compilers and related services.

**Double**: The Double class wraps a value of the primitive type double in an object.

**Enum:** This is the common base class of all Java language enumeration types.

**Float**: The Float class wraps a value of primitive type float in an object.

**Integer**: The Integer class wraps a value of the primitive type int in an object.

**Long:** The Long class wraps a value of the primitive type long in an object.

**Math** – Set 1, Set 2: The class Math contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

**Number**: The abstract class Number is the superclass of classes BigDecimal, BigInteger, Byte, Double, Float, Integer, Long, and Short.

**Object**: Class Object is the root of the class hierarchy.

**Package**: Package objects contain version information about the implementation and specification of a Java package.

**Process**: The ProcessBuilder.start() and Runtime.exec methods create a native process and return an instance of a subclass of Process that can be used to control the process and obtain information about it.

**Runtime**: Every Java application has a single instance of class Runtime that allows the application **String** to interface with the environment in which the application is running.

**RuntimePermission**: This class is for runtime permissions.

**SecurityManager**: The security manager is a class that allows applications to implement a security policy.

**Short**: The Short class wraps a value of primitive type short in an object.

**StackTraceElement:** An element in a stack trace, as returned by Throwable.getStackTrace().

**StrictMath**- Set1, Set2: The class StrictMath contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

- Set1, Set2: The String class represents character strings.

**StringBuffer**: A thread-safe, mutable sequence of characters.

**StringBuilder:** A mutable sequence of characters.

**System**: The System class contains several useful class fields and methods.

**Thread**: A thread is a thread of execution in a program.

**ThreadLocal**: This class provides thread-local variables.

**Throwable**: The Throwable class is the superclass of all errors and exceptions in the Java language.

1. **Features of java:**

**Simple:**

Java language is easy to learn. Simple to understand.

**Object Oriented:**

It has all OOP features such as abstraction, encapsulation, inheritance and polymorphism.

**Robust:**

With automatic garbage collection and simple memory management model (no pointers like C/C++), plus language features like generics, try-with-resource. Java guides programmer toward reliable programming habits for creating highly reliable applications.

**Secure:**

No explicit pointer

Java Programs run inside a virtual machine sandbox

Class loader**:** Classloader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.

Bytecode Verifier**:** It checks the code fragments for illegal code that can violate access right to objects.

Security Manager**:** It determines what resources a class can access such as reading and writing to the local disk.

**High Performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.

It is still a little bit slower than a compiled language (e.g., C++).

**Multithreaded:** A thread is like a separate program, executing concurrently.

We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

**Platform Independence:** Java is platform independent because it is different from other languages like [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

### Dynamic:

Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

Java supports dynamic compilation and automatic memory management (garbage collection).