***History of Java:-***

1. Java was developed by ***James Gosling*** at ***Sun Microsystems in 1996.***
2. It is commonly used to develop web applications, apps, enterprise softwares,etc
3. Java name was derived from ***Coffee Bean***.
4. It was initially called as ***‘Oak’*** later renamed as ***“Java”.***
5. It is popular, high level, object oriented programming language known for its versatility and portability across all platforms.
6. Currently JAVA is owned by ***Oracle Systems.***

***Features Of Java with Real-Life Examples:-***

1. ***OOP’s Concepts:-***
   1. Java follows OOP’s paradigm which means **everything is treated as Object** and software is designed using Classes and Objects.

***Real-Life Examples:-***  Consider a library management systems. Each book is an object, with attributes like title, author, . The library system manages these objects and performs operations like lending & returning a book.

1. ***Platform Independence :***-
   1. Java code is compiled into bytecode that runs on any platform with JVM, making JAVA programs portable across different OS like Windows, Linux, macOS.

***Real-Life Examples:-*** If we develop a mobile app in Java for android, that app can run on any android phone ,regardless of manufacturer OS or hardware configurations.

1. ***Automatic Garbage Management (Garbage Collection):-***
   1. Java does automatically handles memory management by removing unused objects through garbage collection. This helps to prevent memory leaks.

***Real-Life Examples:-*** In Banking application when users complete transactions the system automatically cleans up memory used by completed transaction objects, ensuring the system continues to perform well as more transactions are processed.

1. ***Multithreading:-*** 
   1. Java supports multithreading , which allows multiple threads to run concurrently, enabling CPU better utilization and smoother performance.

***Real-Life Examples:-*** In Online gaming application, multiple players actions can be handled by different threads, allowing the game to process all players movements simultaneously without lagging.

1. ***Robust and Secure :-***
   1. Java offers secure and strong memory management and exception handling, as it provides some features as data abstraction as well.

***Real-Life Examples:-*** In online banking application java’s security features ensure that users personal and transaction data remains safe by preventing unauthorized access to code’s sensitive information.

1. ***High performance:-***
   1. While java is interpreted via(JVM), it also supports Just-In-Time(JIT) compilation, which improves performance by compiling bytecode to native machine code during runtime.

***Real-Life Examples:-***

1. Stock Trading System.
2. IDE’s like Eclipse , VSCode,etc
3. ***Exception Handling:-*** 
   1. Java has a robust exception handling mechanism that ensures a program can handle errors gracefully without crashing.

***Real-Life Examples***:- In ticket booking system, if user tries to book seat that is already taken, then JAVA’s exception handling will catch that error and notify the user without terminating entire application.

1. ***Rich Standard Library:-*** 
   1. Java provides a rich set of libraries (API’s) that include the utilities for file handling, networking, database, connectivity, GUI and much more.

***Real-Life examples:-*** In e-commerce applications we can use libraries to handle user authentication, database transactions and secured payments without need to write manual code for the applications.

**Java program code has 5 main blocks:-**

1. Main Method
2. Methods
3. Default Constructors
4. Static blocks
5. Non-Static Blocks

***Variables in Java:-***

1. Variables are used to store data values that can be referenced and manipulated throughout your program.
2. Each variable has a type (which determines what kind of data it can hold) and a name (used to reference it in the code).

***There are three main types of variables:-***

* 1. ***Local Variable***:- It is written inside class and declared inside methods, constructors, or blocks, and have method or block scope. Scope of local Variable is limited to that specific method, constructor, block.
  2. ***Global Variable / Instance Variable***:- Declared inside a class but outside methods, constructors, blocks they are specific to each object (instance) of the class.
  3. ***Class Variables (Static Variables):*** Declared with the static keyword, shared by all objects of the class. These variables are common for entire class.

***Notes:-***

* *Java is a case-sensitive Language*
* *Class Name, interface Name, etc always written in ‘Pascal’ standard.*
* *Variables & Methods always written in ‘camelCase’ standard.*
* *Same name variables are not allowed in same method.*
* *A class can have same named variable but one should be local and other should global, in this case if the variable is called inside method it will always give preference to its own variable and if the variable is called outside method then the global variable will get called.*
* *Command to compile multiple files in folder-> javac \*.java*
* *Command to open existing java classes created and displays all methods and fields present in that class -> javap ClassName*
* *Command to open specific method of a class -> javap -c ClassName methodName*
* *In Eclipse* ***.java files stored in -> src folder*** *and* ***.CLASS files stored in -> bin folder src folder has all source code files and bin folder has all compiled binary code class files.***

|  |  |  |
| --- | --- | --- |
| ***Comparison Index*** | ***C++*** | ***Java*** |
| ***Platform-independent*** | ***C++ is platform-dependent.*** | ***Java is platform-independent.*** |
| ***Mainly used for*** | ***C++ is mainly used for system programming.*** | ***Java is mainly used for application programming. It is widely used in Windows-based, web-based, enterprise, and mobile applications.*** |
| ***Design Goal*** | ***C++ was designed for systems and applications programming. It was an extension of the***[***C programming language***](https://www.javatpoint.com/c-programming-language-tutorial)***.*** | ***Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed to be easy to use and accessible to a broader audience.*** |
| ***Multiple inheritance*** | ***C++ supports multiple inheritance.*** | ***Java doesn't support multiple inheritance through class. It can be achieved by using***[***interfaces in java***](https://www.javatpoint.com/interface-in-java)***.*** |
| ***Operator Overloading*** | ***C++ supports***[***operator overloading***](https://www.javatpoint.com/cpp-overloading)***.*** | ***Java doesn't support operator overloading.*** |
| ***Pointers*** | ***C++ supports***[***pointers***](https://www.javatpoint.com/cpp-pointers)***. You can write a pointer program in C++.*** | ***Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.*** |
| ***Compiler and Interpreter*** | ***C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.*** | ***Java uses both compiler and interpreter. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform-independent.*** |
| ***Thread Support*** | ***C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.*** | ***Java has built-in***[***thread***](https://www.javatpoint.com/multithreading-in-java)***support.*** |
| ***Documentation comment*** | ***C++ doesn't support documentation comments.*** | ***Java supports documentation comment (/\*\* ... \*/) to create documentation for java source code.*** |
| ***Inheritance Tree*** | ***C++ always creates a new inheritance tree.*** | ***Java always uses a single inheritance tree because all classes are the child of the Object class in Java. The Object class is the root of the***[***inheritance***](https://www.javatpoint.com/inheritance-in-java)***tree in java.*** |
| ***Object-oriented*** | ***C++ is an object-oriented language. However, in the C language, a single root hierarchy is not possible.*** | ***Java is also an***[***object-oriented***](https://www.javatpoint.com/java-oops-concepts)***language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object*** |

***Q1) Can we have save java file with different name?***

* ***Yes we can have save java file with ay name only if there is no public class present in that file, if public class is there then we need to save file with that name only.***

***Package & Import :-***

1. Package is nothing but a folder or directory in java which is similar to folder in our Operating systems.
2. Package is declared using keyword **package**
3. **import** keyword is used to use data of other class package in current working class.
4. **com**. Is mostly used package name where com refers to a Company.
5. **package com.ntrs, com.infosys, com.cognizant**
6. Two or more classes within same package doesn’t need import keyword.
7. If we want to use classes from another package then only we need to use import keyword.
8. If file has multiple import (more than 5) of same file then IDE automatically changes to **\***
9. Package arranges number of classes, interfaces and sub-packages of same type in particular group/ folder/directory.
10. It is a type of file folder containing same type of files.
11. If we want to share codes then we just need to share binary code package to other people and then they can run it without having source code.

***How to create package using Notepad :-***

***javac -d . fileName.java***

***java filename.className***

***eg:-***

**1.Compile the Welcome.java file:**

**Command: javac Welcome.java**

**2. This command creates a Welcome.class file. To place the class file in the appropriate package directory, use:**

**Command: javac -d . Welcome.java**

**3. This command will create a new folder called FirstPackage. To run the class, use:**

**Command: java FirstPackage.Welcome**

