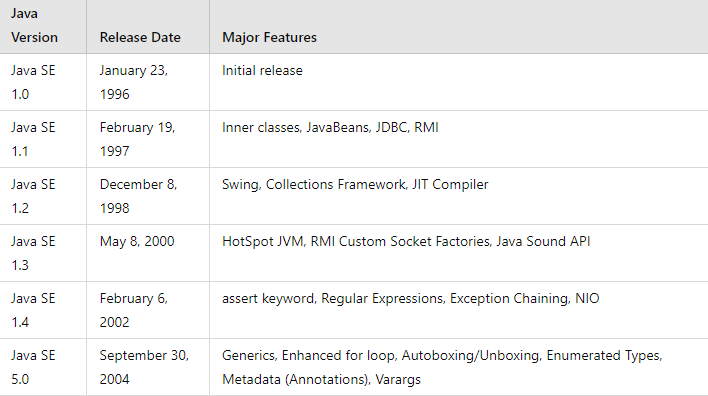
Java

* Java is a class-based, object-oriented programming language developed by James Gosling at Sun Microsystems in the year 1995.
* Java is not a fully object-oriented programming language as it supports primitive datatypes like int, float, etc., which are not objects.
* Java is based on the Write Once, and Run Anywhere (WORA) principle, meaning that the compiled Java code can run on all machines that support Java without the need for recompilation.
* Java is owned by Oracle and is used for:

1. Mobile applications (Especially Android apps)
2. Web applications
3. Games
4. Database Connections
5. And much, much more!

**History of Java**

* Java is a programming language created in 1991 by **James Gosling, Mike Sheridon and Patrick Naughton**, a team of Sun engineers known as the **Green Team**.
* First public implementation of Java was released in1996 as **Java 1.0**



A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

The first version of Java is Java 1.0 which was released in 1996 and the latest version is Java 22 which is released in 2024.

**Why Java is named as Java**

* James Gosling and his team initiated a project to develop a language for digital devices such as set-top boxes, television, etc., and called this project **Greentalk** and its file extension was .**gt** and later become to known as **OAK**.
* The name OAK was used by Gosling after an OAK tree that remained outside his office and OAK was also a national tree of so many nations like USA, France, Germany, etc. But later they had to rename it as it was already a trademark of **OAK Technologies**.
* Gosling and his team did a brainstorm session after which they came up several names out of which **JAVA** was decided after much discussion.
* **Java** is the name of island in Indonesia where the **first coffee** (named as Java) was produced, and this name was chosen by Gosling while having coffee near his office.

**Key Terminology**

Before learning Java, one must be familiar with the following terms of Java:

* **JVM (Java Virtual Machine)**
* **Bytecode**
* **JDK (Java Development Kit)**
* **JRE (Java Runtime Environment) or Java RTE**
* **Garbage Collector**
* **Classpath**

***JVM (Java Virtual Machine):***

* JVM acts as a **run-time** engine to run Java applications. JVM is the one that calls the **main** method present in Java code. JVM is a part of **JRE**.
* The compilation phase of a Java program is done by **JAVAC** compiler which is a primary Java compiler included in the Java Development Kit (JDK). It takes the program as input and generates bytecode as output.
* In the running phase of a program, JVM executes the bytecode generated by compiler.
* The main purpose of JVM is to execute the bytecode produced by the JAVAC compiler. Every Operating System has a different JVM but the output they produce after the execution of byte is same across all the systems. This is why Java is also known as a **platform-independent language.**

***Bytecode:***

* The JAVAC compiler of JDK compiles source code to bytecode so that it can be executed by JVM.
* This bytecode is saved as .**class** file by the compiler. To view the bytecode, a disassembler like **javap** is required.

***JDK (Java Development Kit):***

* It is a **complete kit** that includes everything including **compiler, JRE, Java debugger, Java docs, etc.**
* For a program to execute in Java, we need to **install JDK** on our computer to **create, run and compile the Java program**.

***JRE (Java Runtime Environment):***

* JRE is a part of JDK which allows a Java program to run soon after the installation.
* JRE works as a translator and a facilitator between a Java program and an operating system. It is made up of multiple elements which are:
  + **JVM**
  + **Java class libraries**
  + **Java class loaders**

***Garbage Collector:***

* **Garbage collection** in Java is a process by which Java programs perform **automatic memory management**.
* When Java programs run on the JVM objects are created on the heap, which is a portion of memory dedicated to the program. Eventually some objects will no longer be needed. The **garbage collector** finds these **unused objects** and **deletes** them to **free up memory**.
* **Java garbage collection** is an **automatic process** of looking at heap memory, identifying which objects are in use and which are not, and **deleting** the **unused** **objects**.
* An in-use object, or a referenced object, means that some part of your program still maintains a pointer to that object. An unused or unreferenced object is no longer referenced by any part of your program.

***Classpath:***

The classpath is the **file path** where the **Java runtime** and **Java compiler** look for **.class files** to **load**. By default, JDK provides many libraries. If you want to include external libraries, they should be added to the classpath.

**Features of Java**

1. Java is **platform-independent**, which means that code written in Java can run on any platform that has a Java Virtual Machine (JVM) installed.
2. Java is known for its “**write once, run anywhere**” philosophy, which makes it a popular choice for cross-platform development.
3. Java provides **automatic memory management** through garbage collection, which makes it easier to write and maintain code.
4. Java is a **strongly typed language**, which means that every variable and expression has a specific type that must be declared before use.
5. Java supports **multithreading**, which makes it possible to write programs that can perform multiple tasks simultaneously.

**Execution of a Java program**



This diagram illustrates the process of executing a Java program. Here's a summary and explanation of each component:

* **Source Code (A.java):** The Java source code written by the programmer.
* **Java Compiler (Javac):** Compiles the source code into bytecode.
* **Bytecode (.class):** The compiled intermediate code that is platform independent.
* **Class Loader:** Loads the .class files into the Java Virtual Machine (JVM).
* **Bytecode Verifier:** Checks the bytecode for security and correctness.
* **JIT Compiler:** Just-In-Time compiler converts bytecode into native machine code during runtime.
* **Native Machine Code (001101):** The platform-specific machine code executed by the CPU.
* **Java Virtual Machine (JVM):** The environment that loads, verifies, and executes Java bytecode.

**Useful Links**

[**How to Download and Install Java for 64-bit machine?**](https://www.geeksforgeeks.org/how-to-download-and-install-java-for-64-bit-machine/?ref=lbp)

[**Setting up the environment in Java**](https://www.geeksforgeeks.org/setting-environment-java/?ref=lbp)

[**How to Download and Install Eclipse on Windows?**](https://www.geeksforgeeks.org/how-to-download-and-install-eclipse-on-windows/?ref=lbp)