Java History:

! Here's a brief overview of the history of Java, along with a sample example to illustrate its usage:

The history of Java begins in the early 1990s when a team of developers at Sun Microsystems, led by James Gosling, started working on a new programming language. They aimed to create a language that could be used for developing software that could run on various platforms, regardless of the underlying hardware or operating system. This led to the birth of Java.

In 1995, Sun Microsystems released the first public version of Java, known as Java 1.0. It quickly gained popularity due to its unique features and cross-platform compatibility. One of the key aspects of Java was its "Write Once, Run Anywhere" philosophy, which meant that Java programs could be written once and executed on any device or platform that had a Java Virtual Machine (JVM) installed.

Here's a sample example of a simple Java program that demonstrates the basic syntax and structure of the language:

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

In this example, we define a class named "HelloWorld." The `public static void main(String[] args)` method is the entry point of any Java program, where the execution starts. Within this method, we use the `System.out.println()` statement to print the string "Hello, World!" to the console.

Java has evolved significantly over the years, with numerous updates and additions to the language. It has become one of the most widely used programming languages in various domains, such as enterprise software development, mobile app development (Android), web development, and more.

Java's rich ecosystem includes a vast array of libraries, frameworks, and tools, making it a versatile language for building a wide range of applications. Its strong emphasis on platform independence, combined with its robustness, security features, and extensive community support, has contributed to its enduring popularity in the software development industry.

Let's go through each line of the code and explain its purpose:

1. `public class HelloWorld {`
   - This line begins the definition of a Java class named "HelloWorld." A class is a blueprint or template for creating objects in Java.

Public private protected

2. `public static void main(String[] args) {`
   - This line defines the `main` method, which is the entry point of the Java program. It is a special method that gets executed when you run a Java application. It accepts an array of strings (`String[] args`) as a parameter, which can be used to pass command-line arguments to the program.

Public -access modifier
Static–static values
Void -you may return or may not maynot

3. `System.out.println("Hello, World!");`
   - This line prints the string "Hello, World!" to the console. `System.out` refers to the standard output stream, and `println()` is a method that prints the provided string followed by a newline. In this case, it will display "Hello, World!" in the console output.

4. `}`
   - This closing curly brace marks the end of the `main` method.

5. `}`
   - This closing curly brace marks the end of the `HelloWorld` class definition.

When you run this Java program, the `main` method will be executed, and it will print "Hello, World!" to the console.

So, when you put all the lines together, this Java program defines a class called "HelloWorld" with a `main` method that prints "Hello, World!" to the console.