Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Introduction to JAVA/5a9e9e8b4f99451ac0377ce2

TUTORIAL

Introduction to JAVA

Topics

1.2 Relationship with C and C++

1.3 Features of JAVA

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems in 1991. This language was initially called “Oak” but was renamed “Java” in 1995. Somewhat surprisingly, the original impetus for Java was not the Internet! Instead, the primary motivation was the need for a platform-independent language that could be used to create software to be embedded in various consumer electronic devices, such as toasters, microwave ovens, and remote controls. As you can probably guess, many different types of CPUs are used as controllers. The trouble was that most computer languages are designed to be compiled for a specific target. For example, consider C++.

Although it is possible to compile a C++ program for just about any type of CPU, to do so requires a full C++ compiler targeted for that CPU. The problem, however, is that compilers are expensive and time-consuming to create. In an attempt to find a better solution, Gosling and others worked on a portable, cross-platform language that could produce code that would run on a variety of CPUs under differing environments. This effort ultimately led to the creation of Java.

By 1993 it became obvious to members of the Java design team that the problems of portability frequently encountered when creating

code for embedded controllers are also found when attempting to create code for the Internet. This realization caused the focus of Java to switch from consumer electronics to Internet programming. So, while it was the desire for an architecture-neutral programming language that provided the initial spark, it was the Internet that ultimately led to Java's large-scale success.

Relationship with C and C++

Java is directly related to both C and C++. Java inherits its syntax from C. Its object model is adapted from C++. Java's relationship with C and C++ is important for several reasons. First, many programmers are familiar with the C/C++ syntax. This makes it easy for a C/C++ programmer to learn Java and, conversely, for a Java programmer to learn C/C++.

Second, Java's designers did not "reinvent the wheel." Instead, they further refined an already highly successful programming paradigm. The modern age of programming began with C. It moved to C++, and now to Java. By inheriting and building upon that rich heritage, Java provides a powerful, logically consistent programming environment that takes the best of the past and adds new features required by the online environment. Perhaps most important, because of their similarities, C, C++, and Java define a common, conceptual framework for the professional programmer. Programmers do not face major rifts when switching from one language to another.

One of the central design philosophies of both C and C++ is that the programmer is in charge! Java also inherits this philosophy. Except for those constraints imposed by the Internet environment, Java gives you, the programmer, full control. If you program well, your programs reflect it. If you program poorly, your programs reflect that, too. Put differently, Java is not a language with training wheels. It is a language for professional programmers. Java has one other attribute in common with C and C++: it was designed, tested, and refined by real, working programmers. It is a language grounded in the needs

and experiences of the people who devised it. There is no better way to produce a top-flight professional programming language.

Features of JAVA

No overview of Java is complete without a look at the Java buzzwords. Although the fundamental forces that necessitated the invention of Java are portability and security, other factors played an important role in molding the final form of the language. The key considerations were summed up by the Java design team in the following list of buzzwords.

Simple: Java has a concise, cohesive set of features that makes it easy to learn and use. Java was designed to be easy for the professional programmer to learn and use effectively.

Secure: Java provides a secure means of creating Internet applications. As you are likely aware, every time that you download a “normal” program, you are risking a viral infection. When you use a Java-compatible Web browser, you can safely download Java applets without fear of viral infection or malicious intent.

Portable: Java programs can execute in any environment for which there is a Java run-time system. Many types of computers and operating systems are in use throughout the world i.e. Windows, MacOS, Ubuntu etc. and many are connected to the Internet. For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed.

Object-oriented: Java embodies the modern, object-oriented programming philosophy. Like C++, Java can support an object-oriented approach to writing software.

Robust: Java encourages error-free programming by being strictly typed and performing run-time checks. The ability to create robust programs was given a high priority in the design of Java.

Multithreaded: Java provides integrated support for multithreaded programming. Java was designed to meet the real-world

requirement of creating interactive, networked programs.

High performance: The Java bytecode is highly optimized for speed of execution. The Java bytecode was carefully designed so that it would be easy to translate directly into native machine code for very high performance by using a just-in-time compiler.

Distributed: Java was designed with the distributed environment of the Internet in mind. Java is designed for the distributed environment of the Internet, because it handles TCP/IP protocols. In fact, accessing a resource using a URL is not much different from accessing a file. This allowed objects on two different computers to execute procedures remotely. Java revived these interfaces in a package called Remote Method Invocation (RMI).

Dynamic: Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. This makes it possible to dynamically link code in a safe and expedient manner.