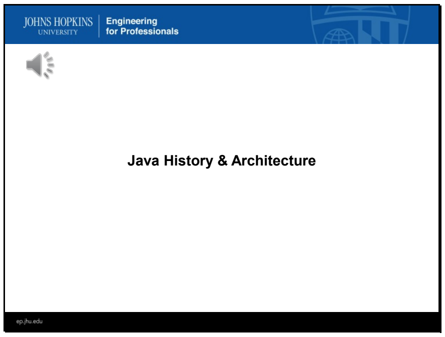
1



Java History & Architecture

ep.jhu.edu

In this lesson, you will learn about the history of the Java programming language, the important aspects of the Java language architecture, and some of the important advantages that Java has over other high-level languages.

2

History of the Java Language

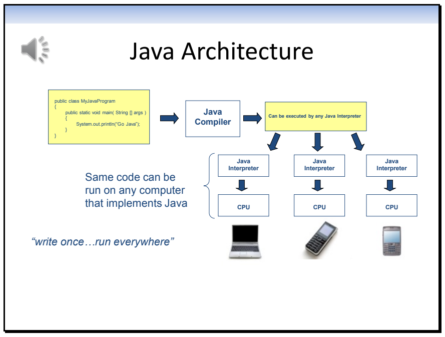
Java was developed by Sun Microsystems, a company that is headquartered in the San Francisco Bay area of California, and which was acquired by Oracle Corporation in 2010. Java was fist released by Sun in 1996. There have been a number of subsequent releases since 1996.

The primary motivation behind the development of the Java language was to develop a reliable language that could be used to program embedded applications for the consumer electronics industry---applications such as intelligent toasters, stereo equipment, and other types of electronics devices. An embedded application is an application in which computer programs may be loaded and run automatically when you turn on a device or actually be embedded in the silicon chips contained in the device. Sun's early work on embedded applications utilized the C++ programming language, but problems with reliability of that language motivated Sun to develop a new language that was similar to C++ in its syntax structure but that didn't have some of the more dangerous features of C++.

Today, most people have heard of Java in conjunction with the Internet and the World Wide Web, and Java has become the programming language of choice for many types of Internet programming applications.

Here's a little bit of Java trivia that you may find interesting. Java was originally called Oak, named after an oak tree that was outside the office window of Java's primary creator, James Gosling. When they did a trademark search, Sun found out that there was already a language called Oak, so they had to come up with another name. When a group of sun employees visited a local coffee house, the name Java came up, and it stuck.

3



Java Architecture

```
public class MyJavaProgram
{
        public static void main( String [] args )
        {
                System.out.println("Go Java");
        }
}
```

Java Compiler

Can be executed by any Java Interpreter

Java Interpreter

CPU

Java Interpreter

CPU

Java Interpreter

CPU

Same code can be run on any computer that implements Java

"write once…run everywhere"

In an earlier lesson, you learned that the central processing unit, or CPU, of a computer executes simple program instructions written in machine language, and that each type of computer has its own version of machine language. You also learned that high-level languages like C and C++ use a compiler to translate the high-level programming instructions into low-level machine language, and some other high-level languages used an interpreter to translate programs into machine language.

Java works a bit differently than other high-level languages. It uses a combination of compilation and interpretation. Instead of generating executable program components in machine code format, a Java compiler generates a type of machine language called ***byte-code***. The byte-code is a very special type of machine language, but it is not a machine language for any particular type of computer. In fact, it is a machine language for a computer that really doesn't exist. This computer that really doesn't exist is called a Java Virtual Machine.

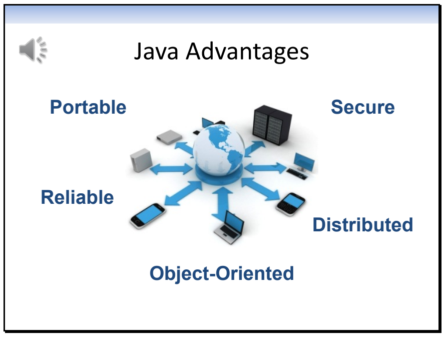
The advantage to this approach is that byte-code is ***architecturally neutral***. This means that Java programs can be written once, compiled once, and run on any computer system that implements a Java interpreter. For example, the bytecode for a specific program can be run on say a laptop that uses an Intel CPU and the Windows operating system, a smart cell phone that uses a special type of CPU and a proprietary operating system, and a Personal Digital Assistant device that uses yet another type of CPU and operating system. So, one set of code can be run on many computers.

You may be wondering "so, what's the big deal with that?" or "why not just take the original Java program that the programmer writes and just compile it into the machine language of each type of computer, like you do for programs written in other languages?"

There are several reasons why this is a big deal, but I'll just mention one now, and save the others for later. The Java compiler, like all high-level language compilers, is a very complex program.

To write a new compiler for each new type of computer that is developed takes a lot of work. And, until the new compiler is developed, not many applications will exist for the new computer. The Java byte-code interpreter, on the other hand, is a relatively simple program. It doesn't take much work to write a new interpreter for any new type of computer, so it's easy for Java programs to run on many existing types of computers as well as those that haven't even been thought of yet. And, since it doesn’t take long to write a Java interpreter for a new device, applications can be made available much more quickly.   Java programmers call this "write once, run everywhere," and it makes Java programs portable.

4



Java Advantages

Portable

Secure

Reliable

Distributed

Object-Oriented

There are a number of advantages that Java has compared to other high-level languages.

One advantage is **portability**…the ability to run a single Java program on many different computers…which we already discussed, and which is due to the special way that Java is architected.

Another advantage is Java’s **reliability**. Java was developed for writing highly reliable computer programs. For example, it wouldn't be too cool if all of a sudden your cell phone shut itself off whenever you had an incoming call. Because it was designed with high reliability in mind, Java contains features that make it harder for programmers to write programs that contain subtle, hard to catch, errors.

Java is **distributed**. Many computer programs today are implemented on a distributed basis. This means that the various components making up the application may reside on different computers scattered across a network, like a cell-phone network or the Internet. Java has built-in features that enable distributed applications to be developed more easily and quickly than many other programming languages.

Java is **secure**. Because Java was designed with distributed and network applications in mind, it has many built-in security features that are not found in other programming languages. The Java interpreter, for example, provides an additional layer of security when you download Java programs embedded in web pages over the Internet. When you download such programs, called applets, the Java interpreter incorporates security measures that minimize the possibility that those programs will do damage to your computer…like mess up your file system.

Java is **object-oriented**. Object-oriented programming is a relatively new approach to writing computer programs that has many benefits. The fundamental building blocks in object-oriented programs are things called objects. Java strongly supports the object-oriented programming approach and also provides an extensive set of built-in objects that can be used by programmers to make programs easier to build. Object-oriented programming concepts will be discussed in later lessons.