Unit 1

Introduction to Java

Structure

- 1.1 Introduction Objectives
- 1.2 History of Java
- 1.3 Features of Java
- 1.4 Java Magic: Byte Code
- 1.5 Summary
- 1.6 Terminal Questions
- 1.7 Answers

1.1 Introduction

Java is a simple language that can be learned easily, even if you have just started programming. A Java programmer need not know the internal of Java. The syntax of Java is similar to C++. Unlike C++, in which the programmer handles memory manipulation, Java handles the required memory manipulations, and thus prevents errors that arise due to improper memory usage. This unit orients you towards understanding of basic Java features. To further enrich your knowledge about the history and basic features of Java read "Java 2 – Complete Reference".

Objectives

After studying this unit, you should be able to:

- describe the history of Java
- explain the features of Java
- explain the Java Magic Byte Code

1.2 History of Java

Although the Java programming language is usually associated with the World Wide Web, its origin predates the web. Java began life as the programming language **Oak**.

Oak was developed by the members of the Green Project, which included **Patrick Naughton**, **Mike Sheridan** and **James Gosling**, a group formed in 1991 to create products for the smart electronics market. The team decided

that the existing programming languages were not well suited for use in consumer electronics. The chief programmer of Sun Microsystems, James Gosling, was given the task of creating the software for controlling consumer electronic devices. The team wanted a fundamentally new way of computing, based on the power of networks, and wanted the same software to run on different kinds of computer, consumer gadgets and other devices. Patenting issues gave a new name to Oak – Java.

During that period, **Mosaic**, the first graphical browser, was released. Nonprogrammers started accessing the World Wide Web and the Web grew dramatically. People with different types of machines and operating systems started accessing the applications available on the web. Members of the Oak team realized that Java would provide the required cross-platform independence that is, independence from the hardware, the network, and the operating system. Very soon, Java became an integral part of the web.

Java works just everywhere, from the smallest devices to supercomputer. Java technology components (programs) do not depend on the kind of computer, telephone, television, or operating system they run on. They work on any kind of compatible device that supports the Java platform.

Self Assessment Questions:

- 1. The earlier name of Java was _____
- 2. The members of Green Project were _____, ____ and _____.
- 3. _____ is the first graphical browser.

1.3 Features of Java

Java defines data as objects with methods that support the objects. Java is purely object-oriented and provides abstraction, encapsulation, inheritance and polymorphism. Even the most basic program has a class. Any code that you write in Java is inside a class.

Java is tuned for Web. Java programs can access data across the Web as easily as they access data from a local system. You can build distributed applications in Java that use resources from any other networked computer.

Java is both interpreted and compiled. The code is complied to a *bytecode* that is binary and platform independent. When the program has to be

Page No. 2

executed, the code is fetched into the memory and interpreted on the user's machine. As an interpreted language, Java has simple syntax.

When you compile a piece of code, all errors are listed together. You can execute only when all the errors are rectified. An interpreter, on the other hand, verifies the code and executes it line by line. Only when the execution reaches the statement with error, the error is reported. This makes it easy for a programmer to debug the code. The drawback is that this takes more time than compilation.

Compilation is the process of converting the code that you type, into a language that the computer understands - machine language. When you compile a program using a compiler, the compiler checks for syntactic errors in code and list all the errors on the screen. You have to rectify the errors and recompile the program to get the machine language code. The Java compiler compiles the code to a bytecode that is understood by the Java environment.

Bytecode is the result of compiling a Java program. You can execute this code on any platform. In other words, due to the bytecode compilation process and interpretation by a browser, Java programs can be executed on a variety of hardware and operating systems. The only requirement is that the system should have a Java-enabled Internet browser. The Java interpreter can execute Java code directly on any machine on which a Java interpreter has been installed.

Thanks to bytecode, a Java program can run on any machine that has a Java interpreter. The bytecode supports connection to multiple databases. Java code is *portable*. Therefore, others can use the programs that you write in Java, even if they have different machines with different operating systems.

Java forces you to handle unexpected errors. This ensures that Java programs are robust (reliable), bug free and do not crash.

Due to strong type-checking done by Java on the user's machine, any changes to the program are tagged as error and the program will not execute. Java is, therefore, **secure**.

Java is *faster* than other interpreter-based language like BASIC since it is compiled and interpreted.

Multithreading is the ability of an application to perform multiple tasks at the same time. You can create multithreading programs using Java. The core of Java is also multithreaded.

The following definition of Java by Sun Microsystems lists all the features of Java.

'Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded and dynamic language.'

Self Assessment Questions:

- 4. _____ is the process of converting typed code to machine code.
- 5. Java code is _____, so that it can easily run on any systems.
- 6. Java is _____ than other interpreted language.

1.4 Java Magic: Byte Code

The key factor that allows Java to solve both the security and the portability problems just described is that the output of a Java compiler is not executable code. Rather, it is bytecode. *Bytecode* is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the *Java Virtual Machine (JVM)*. That is, in its standard form, the JVM is an *interpreter for bytecode*. This may come as a bit of surprise. As you know, C++ is compiled to executable code. In fact, most modern languages are designed to be compiled, not interpreted, mostly because of performance concerns. However, the fact that a Java program is executed by the JVM helps to solve the major problems associated with downloading programs over the Internet. Here is why.

Translating a Java program into bytecode helps it to run much easier in a wide variety of environments. The reason is straightforward: only the JVM needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it. Remember, although the details of the JVM will differ from platform to platform, all interpret the same Java bytecode. If a Java program was compiled to native code, then different versions of the same program should exist for each type of CPU connected to the Internet. This is, of course, not a feasible solution.

Thus, the interpretation of bytecode is the easiest way to create truly portable programs.

The fact that a Java program is interpreted also helps to make it secure. Because the execution of every Java program is under the control of the JVM, the JVM can contain the program and prevent it from generating side effects outside the system. As you will see, safety is also enhanced by certain restrictions that exist in the Java language. When a program is interpreted, it generally runs substantially slower than it would run if compiled to executable code. However, with Java, the difference between the two is not so great. The use of bytecode enables the Java run-time system to execute programs much faster than you might expect.

The Internet helped catapult Java to the forefront of programming, and Java, in turn, has had a profound effect on the Internet. The reason for this is quite simple. Java expands the universe of objects that can move about freely in cyberspace. In a network, two very broad categories of objects are transmitted between the server and your personal computer: passive information and dynamic, active programs. For example, when you read your e-mail, you are viewing passive data. Even when you download a program, the program's code is only a passive data until you execute it. However, a second type of object can be transmitted to your computer: a dynamic, self-executing program. Such a program is an active agent on the client computer, yet it is initiated by the server. For example, a program might be provided by the server to display properly the data that the server is sending. Though desirable and dynamic, the network programs present serious problems in the areas of security and portability. Prior to Java, cyberspace was effectively closed to half the entities that now live there. As you will see, Java addresses those concerns and, by doing so, has opened the door to an exciting new form of program: the applet.

The Java Buzzwords

No discussion of the genesis 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 also played an important role in molding the final form of the language. The key considerations were summed up by the Java team in the following list of buzzwords:

- Simple
- Secure
- Portable
- Object-oriented
- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High performance
- Distributed
- Dynamic

Two of these buzzwords have already been discussed: secure and portable. Let's examine what each of them implies.

Simple

Java was designed to be easy for the professional programmer to learn and use effectively. Assuming that you have some programming experience, you will not find Java hard to master. If you already understand the basic concepts of object-oriented programming, learning Java will be even easier. Best of all, if you are an experienced C++ programmer, moving to Java will require very little effort. Because Java inherits the C/C++ syntax and many of the object-oriented features of C++, most programmers have little trouble learning Java. Also, some of the more confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. Beyond its similarities with C/C++, Java has another attribute that makes it easy to learn: it makes an effort not to have *surprising* features. In Java, there are some clearly defined ways to accomplish a given task.

Object-Oriented

Although influenced by its predecessors, Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean, usable, pragmatic approach to objects. Borrowing liberally from many seminal object-software environments of the last few decades, Java manages to strike a balance between the purists' "everything is an object" paradigm and the pragmatists' "stay out of my way" model. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high performance non objects.

Robust

The multiplatformed environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. Thus, the ability to create robust programs was given a high priority in the design of Java. To gain reliability, Java restricts you in a few key areas, to force you to find your mistakes early in program development. At the same time, Java frees you from having to worry about many of the most common causes of programming errors. Because Java is a strictly typed language, it checks your code at compile time. However, it also checks your code at run time. In fact, many hard-to-track-down bugs that often turn up in hard-to-reproduce run-time situations are simply impossible to create in Java. Knowing that what you have written will behave in a predictable way under diverse conditions is a key feature of Java. To better understand how Java is robust, consider two of the main reasons for program failure: memory management mistakes and mishandled exceptional conditions (that is, run-time errors). Memory management can be a difficult, tedious task in traditional programming environments. For example, in C/C++, the programmer must manually allocate and free all dynamic memory. This sometimes leads to problems, because programmers will either forget to free memory that has been previously allocated or, worse, try to free some memory that another part of their code is still using. Java virtually eliminates these problems by managing memory allocation and de-allocation for you. (In fact, de-allocation is completely automatic, because Java provides garbage collection for unused objects.) Exceptional conditions in traditional environments often arise in situations such as division by zero or "File not found," and they must be managed with clumsy and hard-to-read constructs. Java helps in this area by providing object-oriented exception handling. In a well-written Java program, all runtime errors can and should be managed by your program.

Multithreaded

Java was designed to meet the real-world requirement of creating interactive, networked programs. To accomplish this, Java supports multithreaded programming, which allows you to write programs that do many things simultaneously. The Java run-time system comes with an

Sikkim Manipal University - DDE

elegant yet sophisticated solution for multiprocess synchronization that enables you to construct smoothly running interactive systems. Java's easyto-use approach to multithreading allows you to think about the specific behavior of your program, not the multitasking subsystem.

Architecture-Neutral

A central issue for the Java designers was that of code longevity and portability. One of the main problems facing programmers is that no guarantee exists that if you write a program today, it will run tomorrow – even on the same machine. Operating system upgrades, processor upgrades, and changes in core system resources can all combine to make a program malfunction. The Java designers made several hard decisions in the Java language and in the Java Virtual Machine as an attempt to alter this situation. Their goal was "write once; run anywhere, any time, forever." To a great extent, this goal was accomplished.

Interpreted and High Performance

As described earlier, Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java bytecode. This code can be interpreted on any system that provides a Java Virtual Machine. Most previous attempts at cross-platform solutions have been done at the expense of performance. Other interpreted systems, such as BASIC, Tcl, and PERL, suffer from almost insurmountable performance deficits. Java, however, was designed to perform well on very low-power CPUs. As explained earlier, while it is true that Java was engineered for interpretation, 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. Java run-time systems that provide this feature lose none of the benefits of the platform-independent code. "Highperformance cross-platform" is no longer an oxymoron.

Distributed

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. The original version of Java (Oak) included features for intra-address space messaging. This allowed objects on two different computers to execute procedures remotely. Java has recently revived these interfaces in a package called **Remote Method**

Invocation (RMI). This feature brings an unparalleled level of abstraction to client/server programming.

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. This makes it possible to dynamically link code in a safe and expedient manner. This is crucial to the robustness of the applet environment, in which small fragments of bytecode may be dynamically updated on a running system.

Self Assessment Questions:

- 7. _____ is the ability of an application to perform multiple tasks at a time.
- 8. Java is tuned for _____.
- 9. Java is both ______ and _____.

1.5 Summary

- Java is a programming language developed by Sun Microsystems.
- Java is :
 - Simple It is easy to learn Java.
 - Object-oriented Everything in Java is in form of classes and objects.
 - Distributed Java programs can access data across a network.
 - Compiled and Interpreted The Java code you write is compiled to bytecode and interpreted when you execute the program.
 - Robust Java programs are less prone to error.
 - Architecture Neutral and Portable The bytecode can be executed on a variety of computers running on different operating system.
 - Secure Java does not allow a programmer to manipulate the memory of the system.
 - A high performance programming language Java programs are faster when compared to programs written in other interpreter-based languages.

- Multithreaded It allows multiple parts of a program to run simultaneously.
- Dynamic Maintaining different versions of an application is very easy in Java.

1.6 Terminal Questions

- 1. Give the features of Java.
- 2. What is Bytecode?

1.7 Answers

Self Assessment Questions:

- 1. Oak
- 2. Patrick Naughton, Mike Sheridan, James Gosling
- 3. Mosaic
- 4. Compilation
- 5. Portable
- 6. Secure
- 7. Multithreading
- 8. Web
- 9. Compiled and Interpreted

Terminal Questions:

- 1. Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi-threaded and dynamic language. (Refer section 1.3)
- Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). (Refer section 1.3)