Chapter 1
Introduction to Java

1. Java

Java is a powerful high-level, class-based, object-oriented programming language. It is a general-purpose programming language intended to let application developers *write once, run anywhere* (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture.

Java is used to develop desktop and mobile applications, big data processing, embedded systems, web applications, and so on. According to Oracle, the company that owns Java, Java runs on more than 3 billion devices worldwide, which makes Java one of the most popular programming languages.

2. History of Java

In 1991, a research group working as part of Sun Microsystems's "Green" project headed by James Gosling was developing software to control consumer electronic devices. The goal was to develop a programming language that could be used to develop software for consumer electronic devices like TVs, set-top boxes, and the like. As a result the team announced a new language named “Oak”. Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.

In 1992, the team demonstrated the application of their new language to control a list of home applications using a hand-held device with a tiny touch-sensitive screen.

In 1994, the members of the Green project developed a World Wide Web (WWW) browser completely in Java that could run Java applets. The browser was originally called WebRunner, and later renamed to HotJava.

In 1995, Oak was renamed “Java” because it was already a trademark by Oak Technologies. Java is an island of Indonesia where the first coffee was produced (called java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having coffee near his office.

JDK 1.0 was released in January 23, 1996. After the first release of Java, there have been many additional features added to the language and many versions have been released till now. Now Java is being used in Windows applications, Web applications, enterprise applications, mobile applications, cards, etc.

In January, 2010, Oracle Corporation acquired Sun Microsystems. Oracle Corporation is the current owner of official implementation of Java.

3. Features of Java

Java programming language has some excellent features which play an important role in the popularity of this language. The features of Java are also known as java *buzzwords*.

3.1. Simple

Java syntax is similar to C and C++. However, most of the poorly used and confusing parts form C++ are omitted or managed. There is no need for header files, pointer arithmetic (or even a pointer syntax), structures, unions, operator overloading, virtual base classes, and so on. There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

3.2. Object-Oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and operation. Object-oriented programming languages use concepts such as object, class, inheritance, polymorphism, abstraction, encapsulation etc.

3.3. Distributed

Java is designed as a distributed language for creating applications that runs on networks. Java has features like Remote Method Invocation (RMI) and Enterprise Java Beans (EJB) for creating distributed applications. Java programming language can be used to develop applications that run concurrently on multiple machines and interact with each other on the network.

3.4. Robust

Robust simply means strong. Java provides many safeguards to ensure reliable code. It has strict compile time and run time code checking. It handles all memory management problems and also incorporates strong garbage collection facility.

3.5. Secure

Java is best known for its security. From the beginning, Java was designed to make certain kinds of attacks impossible. Java is secure because of following reasons.

- **It does not have explicit pointer**.
- **Java Programs run inside a virtual machine sandbox** – Untrusted code is executed in a sandbox environment where it could not impact the host system. Users were assured that nothing bad could happen because Java code, no matter where it came from, could never escape from the sandbox.
- **Class loader –** Class loader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine (JVM) dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.
- **Bytecode Verifier:** It checks the code fragments for illegal code that can violate access right to objects.
- **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.

Some other securities can also be provided by an application developer explicitly through SSL (Secure Socket Layer), JAAS (Java Authentication and Authorization Service), and Cryptography etc.

3.6. Architecture Neutral

The Java compiler generates an architecture-neutral object file format (called bytecode). The compiled code is executable on many processors, given the presence of the Java runtime system. The Java compiler does this by generating bytecode instructions which have nothing to do with a particular computer architecture. Rather, they are designed to be both easy to interpret on any machine and easy to translate into native machine code.

3.7. Portable

The phrase write once, run anywhere (WORA) is the major concept for the portability that Java adheres by compiling Java code into bytecode. Java facilitates you to carry the Java bytecode to any platform. Because of this, any Java code that you write and compile into bytecode will run on any operating system for which there exists a compatible Virtual Machine.

3.8. Interpreted

The Java interpreter can execute Java bytecodes directly on any machine to which the interpreter has been ported. Java interpreter generates machine code that can be directly executed by the machine that is running the Java program.

3.9. High Performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C, C++).

In the early years of Java, many users disagreed with the statement that the performance was “more than adequate.” Today, however, the just-in-time compilers have become so good that they are competitive with traditional compilers and, in some cases, even outperform them because they have more information available. For example, a just-in-time compiler can monitor which code is executed frequently and optimize just that code for speed.

3.10. Multithreaded

Java allows multithreading so that Java programs have the capability to run multiple things at the same time.

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

3.11. Dynamic

Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++. Java supports dynamic compilation and automatic memory management (garbage collection).

4. How Java Impacted the Internet

Java has a profound effect on the Internet. Java simplifies web programming. It also innovated a new type of networked program called the applet. Java also addressed some of the issues associated with the Internet: portability and security.

4.1. Java Web Programming

Java is a commonly used language for web development, especially on the server-side. **Java web applications** are distributed applications that run on the internet. Web development with Java allows us to create dynamic web pages where users can interact with the interface. Java has servlet and JSP (Java Server Page) technologies for web programming.

4.2. Java Applets

At the time of Java's creation, one of its most exciting features was the applet. An *applet* is a special kind of Java program that is executed by Java enabled web browsers.

In the early days of Java, applets were a crucial part of Java programming. They illustrated the power and benefits of Java, added an exciting dimension to web pages, and enabled programmers to explore the full extent of what was possible with Java. Although it is likely that there are still applets in use today, over time they became less important. For reasons that will be explained, beginning with JDK 9, the phase-out of applets began, with applet support being removed by JDK 11.

4.3. Security

Every time when you download and execute a program, you are taking a risk, because the program might contain a harmful code like viruses, worms etc. This code can damage the client computer or gain unauthorized access to system resources.

A program that is downloaded and executed on the client computer must be prevented from doing harm. We can download and execute Java programs on the client computer and prevent it from accessing other parts of the computer so that no harm will be done to the client computer.

4.4. Portability

Portability is a major aspect of the Internet because there are many different types of computers and operating systems connected to it. If a Java program were to be run on virtually any computer connected to the Internet, there needed to be some way to enable that program to execute on different systems. Java allows the same application to be downloaded and executed by a wide variety of CPUs, operating systems, and browsers.

5. Procedure vs. Object-Oriented Programming

Procedural oriented programming is a type of programming where a structured method of creating programs is used. With procedure-oriented programming, a problem is broken up into parts and each part is then broken up into further parts. All these parts are known as procedures, routines, subroutines, or functions. They are separate but work together when needed. A main program centrally controls them all. Some procedure-oriented languages

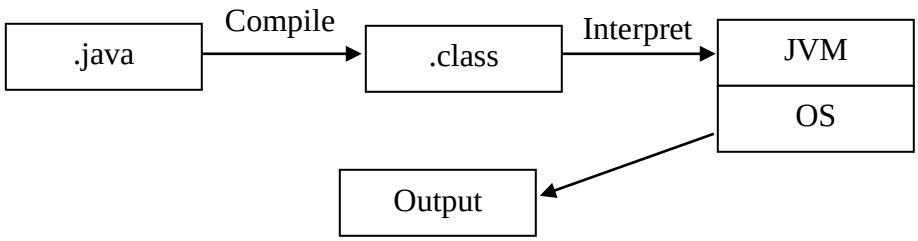
are COBOL (Common Business Oriented Language), FORTRAN (Formula Translation), and C.

Object-oriented programming is the most powerful programming paradigm. Object-oriented programming requires the designer to specify the data as well as the operations to be applied on those data as a single unit. The combined unit of data and the operations is called an object. A program made using this language is therefore made up of a set of cooperating objects instead of an instructions list. Object-oriented programming languages use concepts such as object, class, inheritance, polymorphism, abstraction, encapsulation etc. Some examples are C++, Java, and Python etc.

6. How Java Works

When Java compiler compiles a Java source code (.java file) it converts it into special architecture neutral executable program called *bytecode* that can only be run by using the special program that acts as a machine called **JVM** (**Java Virtual Machine**). JVM is not an actual machine but the run time system for Java.

The java compiler produces an intermediate code known as *bytecode* for the Java Virtual Machine (JVM). This bytecode is not hardware specific. To generate hardware specific code (known as machine code), JVM interprets the bytecode by acting as an intermediary between bytecode and machine code.



7. Setting Up Java Programming Environment

To set up Java programming environment in our computer, we first download appropriate version of Java Development Kit (jdk) form the website URL, https://www.oracle.com/java/technologies/javase-downloads.html. We then install and configure it. For example, in Windows Operating System, we set **PATH** environment variable to **jdk\bin** directory.

Now you can type your java program using any suitable text editor (for example, notepad) and save this file with filename.java as its name. You can use terminal window (for example, MS-DOS) to compile and run your Java program using **javac** and **java** commands respectively.

You can also use an integrated development environment (IDE) to write and run your Java programs easily. Excellent choices are the freely available Eclipse, IntelliJ IDEA, and NetBeans.

8. Writing Your First Java Program

Here is the simple program in java and this can be written in any available text editors like notepad, notepad++ etc.

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

Write the above code and save it as a filename **Welcome.java** (remember Capital 'W'). Open a command prompt and navigate to the folder where you saved your **Welcome.java** file. You are ready to compile here. To compile the java file use the command **javac** that comes along with jdk as shown below.

javac Welcome.java

When this is done there are two possibilities: either there is no message and the prompt appears or there are messages that show the errors. If there are errors, your program will not compile and you must correct them to run the program.
After the successful compilation of **Welcome.java**, **Welcome.class** file is created. This is the file that contains the bytecode. To run this bytecode, type the following command

java Welcome

The result should be displayed to the screen as **Hello World**!

9. Command Line Arguments

Every Java program has a main method with a **String[] args** parameter. This parameter indicates that the main method receives an array of strings, the arguments specified on the command line. For example, consider the program given below:

```
public class Welcome
{
	public static void main(String[]args)
	{
		System.out.println(args[0] + "\n" + args[1]);
	}
}
```

If the program is called as

java Welcome Hello Hi
The command line then would have two items sent to it: Hello and Hi. The variable arguments would contain an array of these two strings as:

```
args[0] = "Hello"
args[1] = "Hi"
```

The program then prints the message
Hello
Hi

Exercises

1. Write a program to display Hello World.
2. Change your program in Q.N.1 using command line input.
3. Write a program to add two numbers. Use command line arguments to provide input to your program.
4. Write a program to find area and circumference of a circle. Use command line arguments to take input.