GCA S4 14

# Programming in Java

## SEMESTER - IV

# BACHELOR OF COMPUTER APPLICATION

## BLOCK - 1

KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

## Subject Experts

**Prof. Anjana Kakati Mahanta,** Professor, Gauhati University
**Prof. (Retd.) Pranhari Talukdar,** Professor, Gauhati University
**Dr. Jyotiprokash Goswami,** Associate Professor, Assam Engineering College

**Course Co-ordinator (s) : Dr. Tapashi Kashyap Das,** Assistant Professor, KKHSOU
**Ms. Sruti Sruba Bharali**, Assistant Professor, KHSOU

## SLM Preparation Team

| UNITS | CONTRIBUTORS |
|---|---|
| 1, 5 | **Mr. Arabinda Saikia**, KKHSOU |
| 2, 3, 4, 6 | **Dr. Tapashi Kashyap Das,** KKHSOU |

## Editorial Team

Content : **Dr. Minakshi Gogoi,** GIMT
Language : **Prof. Robin Goswami,** Retd. Professor, Cotton College
Structure, Format & Graphics : **Ms. Sruti Sruba Bharali,** KKHSOU

**January, 2019**

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State Open University is made available under a Creative Commons Attribution-Non Commercial-Share Alike 4.0 License (international): http://creativecommons.org/licenses/by-nc-sa/4.0/

*Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University.*

**Head Office:** Patgaon, Rani Gate, Guwahati - 781017; Web: www.kkhsou.in/web_new
**City Ofice:** Housefed Complex, Dispur, Guwahati - 781006

The University acknowledges with thanks the financial sipport provided by the **Distance Education Bureau, UGC, New Delhi**, for the preparation of this study material.

## CONTENTS


| | | Pages |
|---|---|---|
| **UNIT 1 :** | **Introduction to JAVA** | **7-26** |
| | An overview of JAVA, Object-oriented programming features, Comparison of Java with C and C++, JAVA Applets, Java Script, Applications, Difference between Java Script and JAVA | |
| **UNIT 2 :** | **Java Programming Basics** | **27-44** |
| | Java Token & Keywords, Constants, Data types; The JAVA class libraries, Declaring a variable, The scope and lifetime of variable, Various Operators | |
| **UNIT 3 :** | **Decision and Control Statements** | **45-56** |
| | Decision Making and Control Statements: if statement, If-else, else-if, switch statement; the for, while, do-while statements | |
| **UNIT 4 :** | **Class in Java** | **57-86** |
| | Class fundamentals: Defining class, Accessing class members, Declaring objects, Passing Arguments to Methods, Returning Multiple Values from methods, Modifiers | |
| **UNIT 5 :** | **Arrays, Strings and Vectors** | **87-111** |
| | Declaring Arrays, Creating Arrays, Initializing Arrays, System Array copy(), Multi-Dimensional Arrays, Strings: string arrays, string methods, string buffer class, Vectors**,** Constructors, copy constructor, Recursive class | |
| **UNIT 6 :** | **Inheritance and Polymorphism** | **112-124** |
| | Inheritance basics, Multilevel Inheritance, Method overriding, Polymorphism, Final and abstract keyword, Static Members | |

## COURSE INTRODUCTION

Since computers cannot understand human languages, programming languages are designed for this purpose. Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java was originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++. Having prerequisite knowledge of C and C++ programming languages helps to get a much quicker understanding of Java.

Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. This course introduces Programming using Java language. The objective of this course is to introduce the learners to the Java programming language and enable them to apply these concepts for solving problems. Programming is a skill best developed by rigorous practice. So, time needs to be devoted for better understanding of this course.

The course is divided into two blocks :

**Block 1** deals with the fundamental concept of Java programming language like data types, operators, decision making statements like if, if-else, switch, break, continue etc. Class and Arrays are also covered in this block. Concepts like inheritance and polymorphism are also covered in this block.

**Block 2** concentrates on some of the most important concepts of Java programming language like packages and interfaces. Applets, AWT and swings are also discussed in this block. Exception handling and I/O are also discussed in this block.

Each unit of these blocks includes some along-side boxes to help you know some of the difficult, unseen terms. Some "EXERCISES" have been included to help you apply your own thoughts. You may find some boxes marked with: "LET US KNOW". These boxes will provide you with some additional interesting and relevant information. Again, you will get "CHECK YOUR PROGRESS" questions. These have been designed to self-check your progress of study. It will be helpful for you if you solve the problems put in these boxes immediately after you go through the sections of the units and then match your answers with "ANSWERS TO CHECK YOUR PROGRESS" given at the end of each unit.

## BLOCK INTRODUCTION

This is the first block of the course '***Programming in Java***'. After completing this block, learners will be able to write basic Java programs using various decision control statements and also using concepts like inheritances and polymorphism.

This block comprises the following six units :

**Unit - 1** gives us an overview of Java. A comparison of Java with C programming language is also provided in this unit. The unit also gives an introduction to Java applets and Java script.

**Unit - 2** introduces us to Java programming basics. Concepts like data types, class libraries, variables and operators are discussed in this unit.

**Unit - 3** describes different decision and control statements. Different types of decision statements like if, if-else, switch, for, while and do-while are discussed in this unit.

**Unit - 4** introduces us to class fundamentals like class, accessing class members etc. Concepts like declaring objects, passing arguments to methods etc are described in this unit.

**Unit - 5** deals with arrays, strings and vectors. Different topics like creating and initializing arrays, string arrays are discussed in this unit. Vectors along with constructors and copy constructors are also covered in this unit.

**Unit - 6** gives us an introduction to inheritance and polymorphism. Different types of inheritances along with concept of polymorphism are described in this unit.

# UNIT 1 : INTRODUCTION TO JAVA

## UNIT STRUCTURE


1.1 Learning Objectives

1.2 Introduction

1.3 Overview of Java

1.4 Basic Features of Java

1.5 C/C++ and Java Language Family

1.6 Platform Independence of Java

1.7 Java Environment

1.8 Installing the Java SDK

1.9 Creating and Running Java Programs

1.10 Let Us Sum Up

1.11 Further Readings

1.12 Answers to Check Your Progress

1.13 Model Questions


## 1.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- describe the basic features of Java programming language
- describe Java environment
- learn to install Java SDK
- create Java Programs and compile & run them

## 1.2 INTRODUCTION

The greatest challenges and most exciting opportunities for software developers today lie in tackling the power of networks. Most of the applications created today, will almost certainly be run on machines connected to the global networks i.e., Internet.

We know that by the mid 1990s, the World Wide Web had

transformed the online world. Through a system of hypertext, users of the Web were able to select and view information from all over the world. However, while this system of hypertext gave users a high degree of selectivity over the information they chose to view, their level of interactivity with that information was low. Moreover, the Web lacked true interactivity—real-time, dynamic, and visual interaction between the user and application.

Sun Microsystems, a company best known for its high-end Unix workstations, developed a programming language named Java to create software that can run on many different kinds of devices. Java language was designed to be small, simple, and portable across platforms and operating systems, both at the source and at the binary level.

Java brings this missing interactivity to the Web. With a Java-enabled Web browser, it is possible to encounter animations and interactive applications. Java programmers can make customized media formats and information protocols that can be displayed in any Java-enabled browser. Java's features enrich the communication, information, and interaction on the Web by enabling users to distribute executable content—rather than just HTML pages and multimedia files—to users. This ability to distribute executable content is the power of Java.

In this unit, we will introduce you to the Java programming language. We will discuss the basic features of Java and how to install the Java Development Kit. We will also discuss how to write a Java program and the procedure for compiling and running a Java program.

## 1.3 OVERVIEW OF JAVA

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991. It took 18 months to develop the first working version. Work on Java originally began with the goal of creating a platform-independent language and operating system for consumer electronics. The original intent was to use C++, but as work progressed in this direction, the Java developers realized that they would be better served by creating their own language rather than extending C++. This language was initially called "Oak" but was renamed

"Java" in 1994 as the Web emerged. Then after Java was used as the basis for a Web browser, called WebRunner. WebRunner was successfully demonstrated, and the *Java*/*HotJava* project took off.

HotJava, Java, and the Java documentation and source code were made available over the Web, as an alpha version, in early 1995. Initially Java was hosted on SPARC Solaris, and then on Windows NT. In the summer of 1995, Java was ported to Windows 95 and Linux. In the fall of 1995 the Java Beta 1 version was released through Sun's Web site, and Java support was introduced in the Netscape 2.0 browser.

The Java Beta 1 release led scores of vendors to license Java technology, and Java porting efforts were initiated for all major operating systems.

In December 1995 the Java Beta 2 version was released, and JavaScript was announced by Sun and Netscape. Java's success became inevitable when, in early December, both Microsoft and IBM announced their intention to license Java technology.

On January 23, 1996, Java 1.0 was officially released and made available for download over the Internet. JavaScript was also released. Netscape 2.0 now provides support for both Java and JavaScript.



***HotJava :*** HotJava is a Web browser that is written in Java. HotJava is a Java-enabled browser. This means that HotJava can execute Java applets contained on Web pages. In order to accomplish this, HotJava calls the Java runtime system. The Netscape 2.0 browser, like HotJava, is also Java enabled. It contains a copy of the Java runtime system embedded within it.

## 1.4 BASIC FEATURES OF JAVA

The Java team has summed up the basic features of Java with the following list of buzzwords :

- **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.

- **Object-oriented :** Java is a true object-oriented language. Many of Java's object-oriented concepts are inherited from C++, the language on which it is based, but it borrows many concepts from other object-oriented languages as well. Like most object-oriented programming languages, Java includes a set of class libraries that provide basic data types, system input and output capabilities, and other utility functions. These basic classes are part of the Java development kit, which also has classes to support networking, common Internet protocols, and user interface toolkit functions.
- **Robust :** Java is a robust language. The multiplatform 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.
- **Secure :** Prior to Java, most users did not download executable programs frequently from Internet, and those who did scanned them for viruses prior to execution. Even so, most users are still worried about the possibility of infecting their systems with virus. In addition to viruses, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords, by searching the contents of your computer's local file system. Java answers both of these concerns by providing a "firewall" between a networked application and your computer. When you use a Java-compatible Web browser, you can safely download Java applets without fear of viral infection or malicious intent. Java achieves this protection by confining a Java program to the Java execution environment and not allowing it access to other parts of the computer.

- **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 elegant yet sophisticated solution for multiprocess synchronization that enables you to construct smoothly running interactive systems.
- **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 the Java Virtual Machine in an attempt to alter this situation. Their goal was "write once; run anywhere, any time, forever." To a great extent, this goal was accomplished.
- **Portable :** In addition to being architecture-neutral, Java code is portable. It is an important design goal of Java.
- **Distributed :** Java is designed for the distributed environment in the Internet, because it handles TCP/IP protocols. In fact, accessing a resource using an 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.

## 1.5 C/C++ AND JAVA LANGUAGE FAMILY

C was developed to meet general system programming needs. It was quickly adapted for general use and found widespread acceptance. C is a high-level procedural language that has many low-level features. These features help to make it versatile and efficient. However, many of these features give it a reputation for being cryptic and hard to maintain. C++ extends the C language to provide object-oriented features. The language is backward compatible with C, and code from the two languages can be used with each other with little difficulty. C++ has found quick acceptance and is supported by a number of pre-built specialized classes.

Java can be considered the third generation of the C/C++ family. It is not backward compatible with C/C++ but was designed to be very similar to these languages. The creators of Java intentionally left out some of the features of C/C++ that have been problematic for programmers. Java is strongly object-oriented. In fact, one cannot create Java code that is not object-oriented. Java's portability is a key advantage and is the reason why Java is often used for Web development.

## 1.6 PLATFORM INDEPENDENCE OF JAVA

We are already familiar how to compile a C or C++ program. With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. It means - ***Java combines both these approaches thus making Java a two-stage system***. The Java development environment has two parts : *a Java compiler* and *a Java interpreter*.

In the Java programming language, all source code is first written in plain text files ending with the **.java** extension. Those source files are then compiled into **.class** files by the ***javac*** compiler. With the compiler, first you translate a program into an inter-mediate code called ***Java bytecodes.***

Bytecodes are not machine instructions and therefore, in the second stage, ***Java interpreter*** generates machine code that can be directly executed by the machine that is running the Java program. The interpreter parses and runs each Java bytecode instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed.

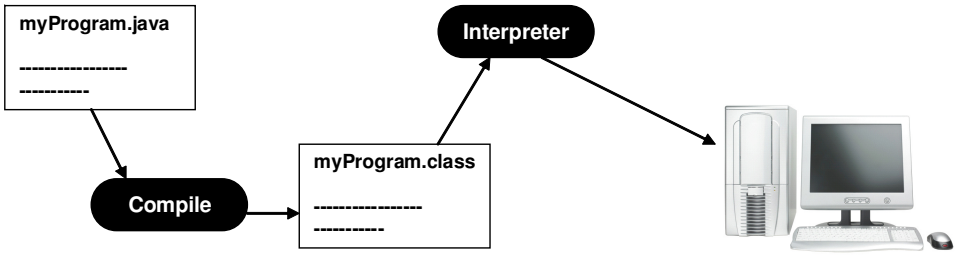



**Figure 1.1 : A Java program first compiled and then interpreted**

Java bytecodes help make "write once, run anywhere" possible. You can compile your program into bytecodes on any platform that has a Java compiler. The bytecodes can then be run on any implementation of the ***Java Virtual Machine (JVM)***. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Unix workstation, or on an iMac, as shown in fig 1.2.

Normally, when you compile a program written in C or C++ or in most other languages, the compiler translates your program into machine codes or processor instructions. Those instructions are specific to the processor your computer is running—so, for example, if you compile your code on a Pentium system, the resulting program will run only on other Pentium systems. If you want to use the same program on another system you have to go back to your original source, get a compiler for that system, and recompile your code.



***Java Virtual Machine (JVM) :*** Java compiler produces an intermediate code known as ***bytecode*** for a machine that does not exist. This machine is called the ***JVM*** and it exists only inside the computer memory. The bytecodes are also known as *virtual machine code* which are not the actual machine code. The actual machine codes are generated by the Java interpreter only.

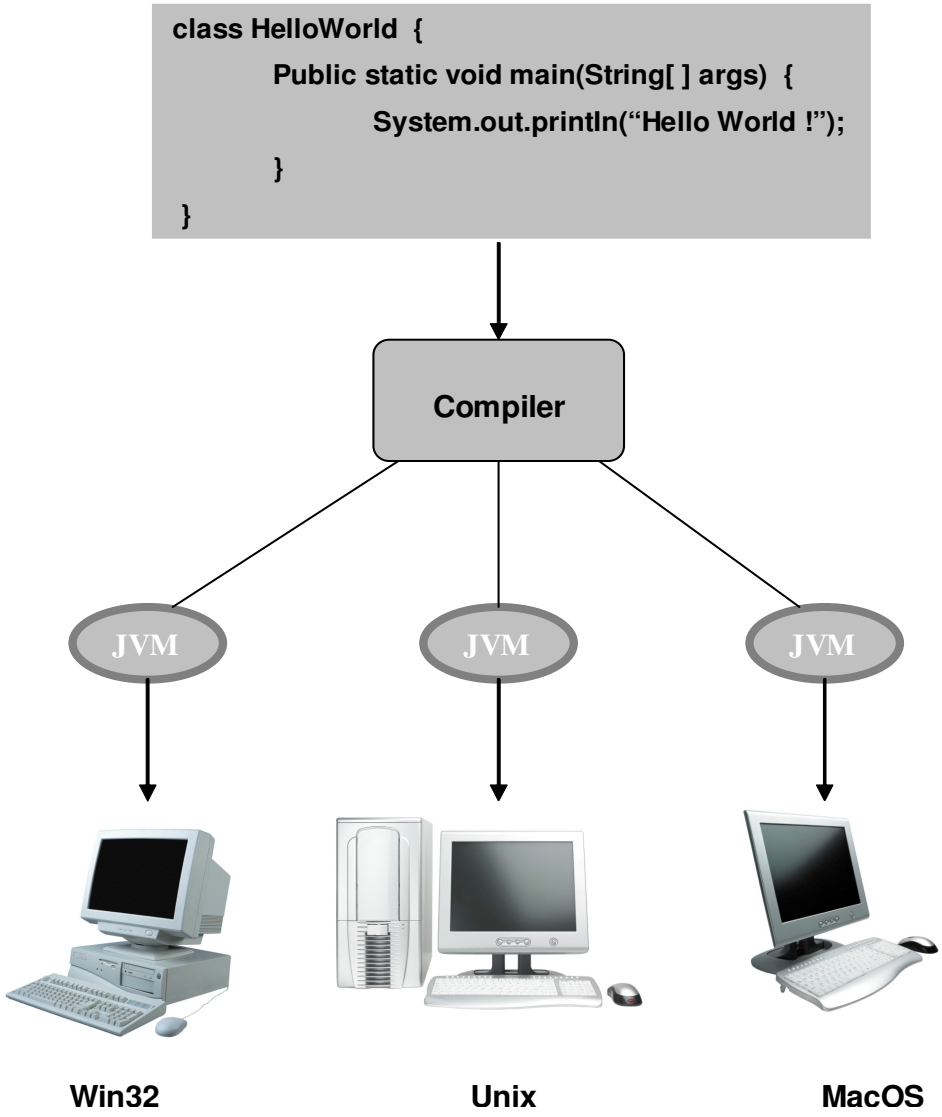



**Figure 1.2 : Program written once and can run on almost any plateform**

## 1.7 JAVA ENVIRONMENT

Java environment includes a large number of development tools and hundreds of classes and methods. The development tools are part of the system known as *Java development Kit* (JDK) and the classes and methods are part of the *Java Standard Library* (JSL), also known as the *Application Programming Inteface* (API).

**Java development Kit :** The Java development Kit comes with a collectlion of tools that are used for development and running Java programs.

Some of them are :

| | |
|---|---|
| *java* | The loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. |
| *javac* | The compiler, which converts source code into Java bytecode |
| *jar* | The archiver, which packages related class libraries into a single JAR file. |
| *javadoc* | The documentation generator, which automatically generates documentation from source code comments |
| *jdb* | The Java debugger |
| *jps* | The process status tool, which displays process information for current Java processes |
| *javap* | The class file disassembler |
| *appletviewer* | This tool can be used to run and debug Java applets without a web browser. |
| *javah* | The C header and stub generator, used to write native methods. |

An **application programming interface (API)** is an interface implemented by a software program to enable interaction with other software, similar to the way a user interface facilitates interaction between humans and computers. Java APIs include hundreds of classes and methods grouped into several functional packages. Most commonly used packages are :

- Language support package
- Utility package
- Input/Output Package
- Networking Package
- AWT(Abstract Window Tool Kit) Package
- Applet Package

## 1.8 INSTALLING THE JAVA SDK

Java is a programming language that allows programs to be written that can then be run on more than one type of operating system. A program written in Java can run on Windows, UNIX, Linux etc. as long as there is a Java runtime environment installed.

For the first time, you just want to run Java programs so, download the Java Runtime Environment, or JRE. Suppose, you want to develop applications for Java, download the Java Development Kit, or JDK. The JDK includes the JRE, so you do not have to download both separately.

You can download the JDK from the Sun Microsystems, free of charge by using this URL http://java.sun.com/javase/downloads/index.jsp . The first download page should look like the top page shown in fig 1.3.

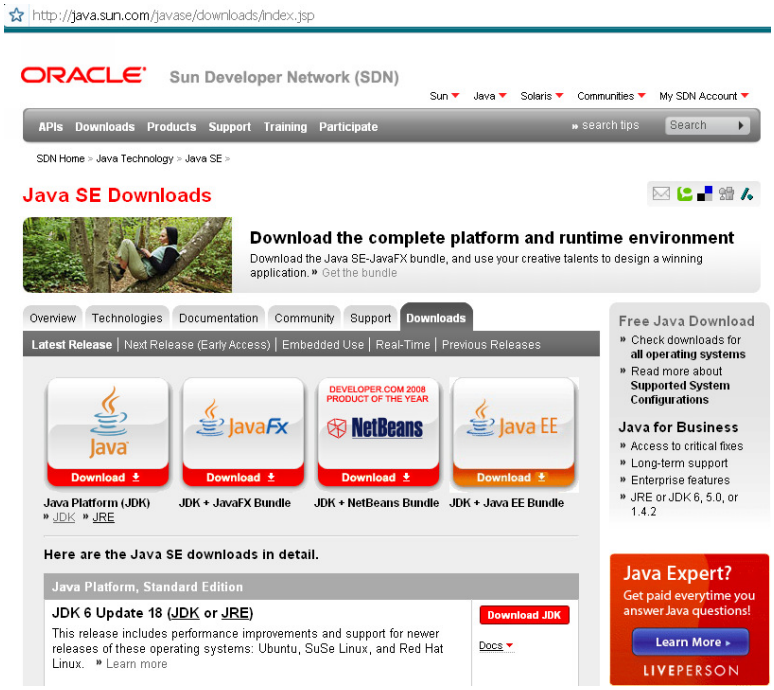



**Figure 1.3 : Download page for JDK**

Click on the ***Download JDK*** tab shown, and this will save the **jdk-6u18-windows-i586.exe** file on your computer.

After the download is complete, for installing the software, double click the .exe file, and this will automatically install the software by giving some instruction.

The JDK has the diectory structure as shown in the fig 1.4.

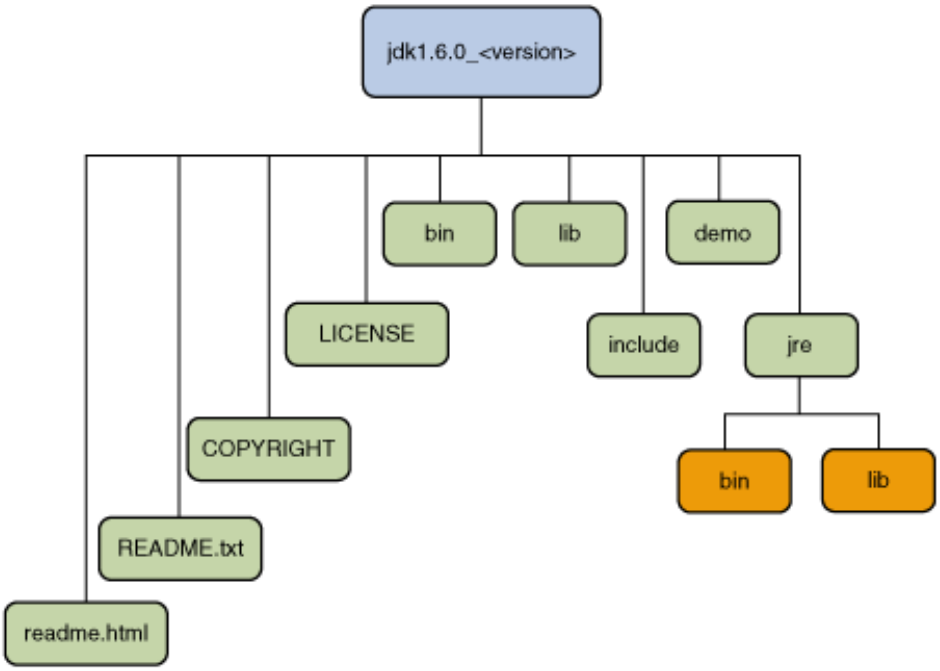



**Figure 1.4 JDK directory structure**

To compile a Java program on the command line, you will use the command ***javac*** and to run a compiled program on the command line you will use the ***java*** command. These two commands are executed by running the **javac.exe** and **java.exe** programs that are located in the **bin** folder of the folder **jdk1.6.0_18** in C drive. For your operating system (e.g. Windows or Linux) to be able to run these programs, you have to tell it where they are i.e. you can set the PATH variable if you want to be able to conveniently run the JDK executables (javac.exe, java.exe, javadoc.exe, etc.) from any directory.

If you don't set the PATH variable, you need to specify the full path to the executable every time you run it, such as :

*C:> "\Program Files\Java\jdk1.6.0_<version>\bin\javac" MyClass.java*

**Procedure for setting the PATH variable :**

a) Click Start > Control Panel > System on Windows XP or Start > Settings > Control Panel > System on Windows 2000.

b) Click Advanced > Environment Variables.

c) Add the location of bin folder of JDK installation for PATH in **User Variables** and **System Variables**.