UNIT 3 JAVA LANGUAGE BASICS

Structure **Page Nos.**

3.0 INTRODUCTION

Java is an island in Indonesia, or you can say Java is type of coffee, but Java is most popular as a programming language. In **1991** Java Programming Language stepped into the world and got dominant status. In this unit you will learn the advantages and strength of Java, and what actually makes Java powerful and popular.

Broadly a Java program can be divided into two types, **Application** and **Applets.** The differences between the two are explained in this unit. The concept of Java virtual Machine makes this language platform independent. With this concept of Java virtual machine we will move towards the programming of Java language. You can learn the basic programming concept of Java and how you can compile and execute the Java Application and Applet programs. To improve your programming concepts, you can move towards the declaration and initiations of different data types and variables. Similar to C, Java also contains operators. Those are explained at the end of this unit.

3.1 OBJECTIVES

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

- describe the strength of Java language;
- explain Java virtual machine concept;
- differentiate between Java applet and Java application;
- write simple programs in Java;
- compile and execute Java applets and application programs;
- describe the data types in Java, and
- declare and initialize of variables in the programs.

3.2 INTRODUCTION TO JAVA

Java was developed by a team of computer professionals under the guidance of James Gosling at Sun Microsystems in 1991. They wanted to give a suitable name. Initially they called it **"oak"** after seeing an oak tree from their window, but after some weeks they were discussing another name and were sipping Java coffee, so one of them suggested the name **"Java".** Other than this there is no interesting story/reason about its name **Java.** The only reason I can say is that its designers wanted to give a

beautiful name to their beautiful language, just as all parents want to give a sweet name to their sweet child.

Java is a **simple** (similar to C/C++), **scalable** (easy to integrate), **object oriented** (able to program real life complexities)**, general purpose** programming language with powerful features, which can be used to develop a variety of applications from simple web animations to high-end business applications that program hand-held devices, microwave appliances, cross platform server applications, etc.

Java is a **strongly typed language**. This specification clearly distinguishes between the compile time errors that must be detected at compile time and those that occur at run time.

Generally a language is either compiled or interpreted, but Java is both compiled as well as interpreted. First a Java program is complied and comes in the form of **"Java byte code"** (which is an intermediate code). Then this Java byte code is run on interpreter to execute a program. This byte code is the actual power of Java to make it popular and dominating over other programming languages. We will discuss this topic in more detail in a later section of this unit.

3.2.1 Basic Features

In the last decade Java has become very popular. There are many reasons why Java is so popular and some of these reasons are explained here: **carefully read all the features of Java and try to realize its strength.**

Platform Independent

Java is Platform independent. The meaning of platform here may be confusing for you but actually this word is poorly defined. In the computer industry it typically means some combination of hardware and system software but here you can understand it as your operating system.

Java is compiled to an intermediate form called **Java byte-code** or simply byte code. A Java program never really executes immediately after compilation on the host machine. Rather, this special program called the Java interpreter or Java Virtual Machine reads the byte code, translates it into the corresponding host machine instructions and then executes the machine instruction. A Java program can run on any computer system for which a JVM (**Java Virtual Machine**) and some library routines have been installed. The second important part which makes Java portable is the elimination of hardware architecture dependent constructs. For example, Integers are always four bytes long and floating-point variables follow the IEEE 754.You don't need to worry that the interpretation of your integer is going to change if you move from one hardware to another hardware like Pentium to a PowerPC. You can develop the Java program on any computer system and the execution of that program is possible on any other computer system loaded with JVM. For example, you can write and compile the Java program on Windows 98 and execute the compiled program on JVM of the Macintosh operating system. The same concept is explained in Figure 1 given below.

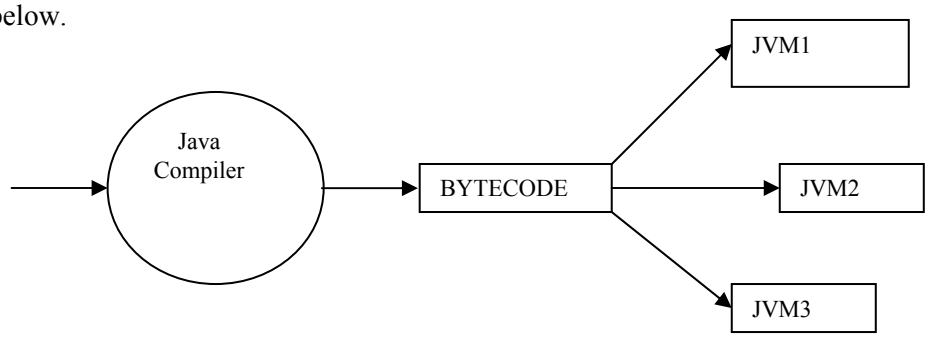


Figure 1: Compilation and execution of Java program

Object Oriented

As you know that in objects represent object-oriented languages data. Objects have two sections. The first is **Data** (instance variables) and the second is **methods**. Data represents what an object is. A method represents what an object does. The Data and methods are closely related to the real world structure and behavior of objects. Object oriented programming has a number of advantages like. Simpler to read program, efficient reuse of programming segments, robust and error-free code.

Java is a true object-oriented language, which provides a platform to develop an effective and efficient application and program real life complexities. Java does not allow methods without class, thus an application consists of only the object which makes it true OOl. Most of the Object-oriented concepts in Java are inherited from C++ which makes it easy for traditional programmers to understand it.

Easy to Learn

Java is easy to learn for programmers because it is (syntax) similar to C and C++ and most of the complex parts of C/C++ have been excluded including **operator overloading, multiple inheritance and pointers**. Approximately half of the bugs in C and C++ programs are related to memory allocation and de-allocation. Therefore the important addition in Java is automatic memory allocation and de-allocation.

Do not think Java is very simple, it is both a simple as well as complex language depending on how you use it because Java has a wide range of applications, simple to complex.

Robust

Java provides checking for possible problems at two levels, one at the compile time and the other at the run time, so programs are highly **reliable** and eliminate situations that are error-prone compared to C/C++. The best and worst features of C and C++ are **pointers** that help in direct manipulation of memory addresses. The power of pointers is as a great tool used by expert programmers for developing system software, driver, etc. But many times pointers are the main cause of **runtime errors** because of improper use of memory. Java eliminates pointer manipulation completely from the language, and therefore eliminates a large source of runtime errors. Java programmers need not remember to de-allocate memory in programs since there is a **garbage collection mechanism** which handles de-allocation of memory. It provides powerful a robust exception handling mechanism to deal with both expected and unexpected errors at run time.

Secure

Java is intended to work in networked and distributed environments by providing security. All the references to memory are **symbolic references,** meaning that the user is not aware where in the memory program is present, it totally depends on the JVM and machine on which the program is running. Each applet is loaded on its own memory space, which avoids the information interchange between applets.

Java applets can be executed in run time environment that restricts them from introducing **viruses**, deleting and modifying files in the host computer. The Java enabled web browser checks the byte code of applets to ensure that it should not do anything wrong before it will run the applet. Furthermore, Java is a **strongly typed language**, which means that variables should be declared and variables should not change types. Type casting are strictly limited highly **sensible, therefore** you can cast an **int** to a **long** or you can cast a **byte** to a **short** but you cannot cast an **int** to a boolean or an **int** to a **String**. The major security issue in today's software world is

BUGS. Unintended bugs are responsible for more data loss than data loss because of viruses. In Java it is easier to write bug-free code then in other languages.

Multi-threaded

Before answering what is multithreading, let me explain you what '**thread**' is. Simply, a thread is a program's path of execution. In your problems, when multiple events or actions need to occur at the same time, how you will handle it? For example, a program is not capable of drawing pictures when you keep pressing keys of the keyboard. The program gives its full attention to receiving the keyboard input and doesn't draw the picture properly.

The best solution to this problem is the execution of two or more sections of a program at the same time and this technique is known as multithreading. Multithreaded applications deliver their potent power by running many threads **concurrently** within a single program. A web browser, for instance, can print a file in the background while it downloads a page in one window and formats the page as it downloads. The ability of an individual program to do more than one thing at the same time is most efficiently implemented through threads.

Java is inherently multi-threaded, for example **garbage collection** subsystem runs as a low-priority thread. A single Java program can have many different threads executing independently and continuously, for example, different Java applets on the same web page can run together with getting equal time from the processor. Because multithreaded applications share data and all threads of an application exists in the same data space therefore to maintaining reliability is sometime difficult. To making easy the use of threads Java offers features for synchronization between threads.

Dynamic

Java was designed to adapt to an evolving environment, therefore the Java compiler is smart and dynamic. If you are compiling a file that depends on other non-compiled files, then the compiler will try to find and compile them also. The compiler can handle methods that are used before they're declared. It can also determine whether a source code has been changed since the last time it was compiled. In Java classes that were unknown to a program when it was compiled can still be loaded into it at runtime. For example, a web browser can load applets of other classes without recompilation.

High Performance

As we know in Java we have to first compile the program, then execute it using Java interpreter. In general, interpreters are slow, because an interpreter executes programs instruction by instruction while Java is a fast-interpreted language. Java has also been designed so that the run-time system can optimize their performance by compiling bytecode to native machine code on the fly (execute immediately after compilation). This is called "**just in time"** (JIT) compilation.

According to 'Sun' with JIT compilation, Java code can execute nearly as fast as native compiled code and maintain its transportability and security but array bounds checking are a problem of the natively compiled Java code. Many companies are an working on native-machine-architecture compilers for Java. These will produce an executable code that does not require a separate interpreter, and that is indistinguishable in speed from C++.

Java offers two flavors of programming, Java applets and Java application. Applets are small Java programs (mostly) that can be downloaded over a computer network and run from a web page by using a Java enabled browser like Netscape / Microsoft Internet Explorer. Applets used to add dynamic features like animation, sound etc. to web pages.

3.2.2 Java Virtual Machine Concepts

When a Java program is compiled it is converted to **byte code** which is then executed by the Java interpreter by translating the byte code into machine instructions.

Java interpreter is part of Java runtime environment. Byte code is an intermediate code independent of any machine and any operating system. Program in Java run time environment, which is used to interpret byte code, is called Java Virtual Machine (JVM). The Java compiler reads Java language source files, translates the source into Java byte codes, and places the byte codes into class files.

Any machine for which Java interpreter is available can execute this byte code. That's why Java is called **Machine independent and Architecture neutral**. *Figure 2* shows that Java compiler is accepting a Java program and producing its byte code. This byte code can be executed on any operating system (Window-98, Macintosh, Linux etc.) running on any machine with suitable Java interpreter of that machine.

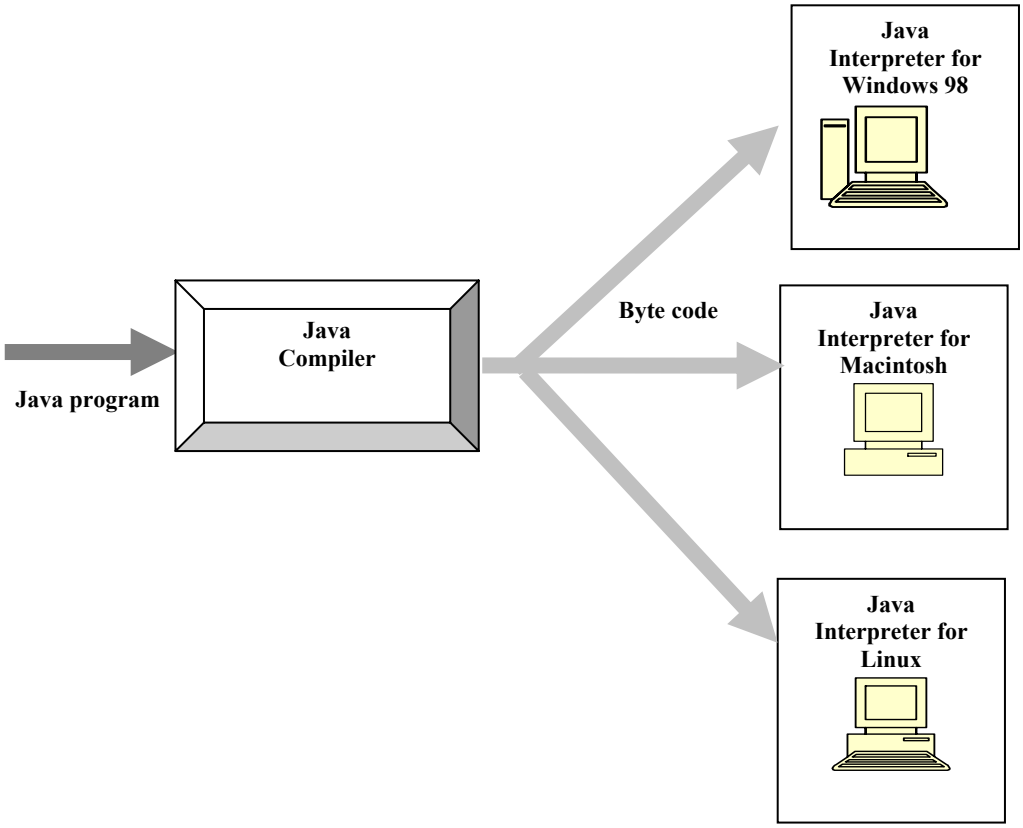


Figure 2: Java is Machine Independent and Architecture Neutral

The JVM plays the main role to making Java portable. It provides a **layer of abstraction** between the compiled Java program and the hardware platform and operating system. The JVM is central to Java's portability because compiled Java programs run on the JVM, independent of whatever hardware is used.

☞ Check Your Progress 1

1) What is the meaning of "virtual" in Java virtual machine?

……

……

2) How can you say that Java is a secure programming language?

……

……

3) What was the first name given to Java by its team?

……

……

In the next section let us see how you can develop a Java program on your own machine. After developing a program try to run it on different machines so that you can understand the meaning of machine Independence in practice.

3.2.3 A Simple Java Program

Java offers two flavors of programming, Java applets and Java application. It is important to understand the basics of both flavors. Let us taste both cups of Java programming one by one.

Java application Program

Let us write your first program with the file name "Application". This program, after successful compilation and run, prints the words "This is My First Java Program" on your display. This program is very basic but the goal of this program is not to teach you how to print words on your display but to tell you how to type, save, compile and execute Java programs. In this section you can learn many issues that can go wrong even if your source code is correct.

To create your own program you should follow the three steps given below. In the next section you will get an explanation of each step in detail:

1. First of all using any text editor, type Java source file of your program as given below.
2. Now compile the source file-using compiler named **Java c** that takes your source file and translates its statements into a bytecode file.
3. Run the program using interpreter named **Java** that takes your bytecode file and translates them into machine instructions that your computer can understand.

// Code start here…. don't include these line numbers in your program

```
1. /* This is my First Java Application Save the file as Application.Java: same as class
name */
2.  class Application
3.        {
4.            public static void main (String args[ ] )
5.                {
6.                    System.out.println("This is My First Java Application");
7.                } // main ends here
8.        }// Code ends here
```

Let us see the above program line by line.

Line 1. /* This is my First Java Program. Save this file as Application.Java same as class name */

Comments in Java are similar to C++. Everything between /* and */ is ignored by the compiler but Comments allow you to describe the details of the program. This is useful for developing the understandability in your program.

Line 2. class Application

Second line in the program defines a class named Application using a keyword class. After that, class definition is specified within curly braces. More about Java classes you can read in Unit 1 Block 1 of this course.

class Application
{
.
class definition
}
Line 4. public static void main (String args[ ])

Line number 3 is not important to discuss here as it contains only the separator. What use is the separator you will see later.

“public static void main (String args[ ])” This is the point from where the program will start the execution. This program starts the execution by calling main () method. In this line public, static, and void all are keywords. May be you are thinking of the meaning of ‘keyword’. Keywords are nothing but some reserved words, details you will see in the later section of this unit.

The public keyword is used to control the access of various class members. If member is public it can be accessed outside the class. So we have to declare main () as public because it has to be invoked by the code outside the class when program is executed. Static key word allows the main () method to be executed without creating an object of that class, and void means main () method does not return any value.

Separators define the structure of a program. The separators used in Application are parentheses, ( ), braces, { }, the period, ., and the semicolon, ;. Java contains six separators and all are widely used by programmers. Let us see what the proper meaning of each separator is.

Parentheses ( ), generally it encloses arguments in method definitions or calling and used to delimits test expressions control statements.

Braces { }, defines blocks of code and it also defines automatically initializes arrays. Square bracket [ ], declares array types.

Semicolons; are used to terminate statements.

Single coma “,” is use to separates successive identifiers in variable declarations.

Single dot “.” Selects a method from an object and separates package names from sub-package and class names. And in last “:” is used in loop labels.

Line 6. System.out.println(“This is My First Java Application.”);

This line prints the string “This is My First Java Application”. In this line println () function is used to display this line. The println () function accepts any string and display its value on console.

Typing and saving program

To write the program you need a text editor like Notepad, Brief, or vi or any other. You should not use a word processor like Microsoft Word because word processors save their files in a proprietary format and not in pure ASCII text. Now type the above program into a new file but remember to type it exactly as it written here because Java is case sensitive, so **System** is not the same as **system** and **CLASS** is not the same as **class**. But white space is meaningless except inside string literals.

Assume we are using Notepad in windows. Save this program in a file called Application.Java. Windows text editors Notepad add a three letter ".txt" extension to all the files saved without informing user. You may get unexpectedly file called " Application.Java.txt." because it in not “.Java” compiler will not compile this file. If your editor has this problem, you can change to better editor.

Compiling and running Application.Java

Before compiling ensure that your Java environment is correctly configured. To set PATH and CLASSPATH see MCS 025 ‘s Java Programming section.

```
C:> Javac Application.Java
C:> Java Application
This is My First Java Application
C:>
```

Java compiler named Javac that takes your source file and translates its instructions into a bytecode file. Java interpreter named Java that takes your bytecode file and translates them into instructions that your computer can understand. When you compile the program you need to use Application.Java but when you run the program you need not use Application.class.

A Simple Java Applet

As you know Java programs can be classified into applets, application and servlets. *Java servlets are similar programs like applets except they execute on servers side*. **Servlets** are Java programs answer to traditional CGI programming. They are programs that run on a Web server and build Web pages.

Now let us try to write a code for Java applet, When we write a program for applet, we must import two Java groups. Import is a process to tell the compiler where to find the methods of classes from library we will use in the program. These two important groups are Java.awt and Java.applet. Java.awt is called the Java abstract windows tool kit and Java.applet is the applet group.

To create an applet, which will display “Hello IGNOU”, follow the three steps that are used for creating Java application as given below.

1. Create a Java source file.
2. Compile the source file.
3. Run the program.

Write the following Java source code into text editor and save as MyFirstApplet.Java

```
// Code start here… don’t include these line numbers in your program
1. import Java.applet.Applet;
2. import Java.awt.Graphics;
3. public class MyFirstApplet extends Applet
    {
4.    public void paint (Graphics g)
      {
5.        g.drawString("Hello IGNOU !", 50, 25);
6     }
7   }  // please note in applets there is no ‘ main () ‘.
// code ends here….
```

Most of the symatx of this program is similar to previous the Java application program you have just completed. The important steps for developing basic Java applet are explained here step by step.

1. import Java.applet.Applet
2. import Java.awt.Graphics;

The above two lines import two important Java groups. Java.awt and Java.applet are necessary whenever you develop applets. java .awt is called the Java abstract windows tool kit and Java.applet is the applet group. Here in the code java.applet.Applet is required to create an applet and java.awt.Graphics is required to paint on the screen.

3. public class MyFirstApplet extends Applet

My First Applet is the executable class. It is public class and it extends applet class which means that a programmer builds the code on the standard Applet class.

4. public void paint (Graphics g)

In our MyFirstApplet class we have defined paint. Method inside which we are using object of Graphics class. Graphics class is available in Java, group Java.awt.

5. g.drawString ("Hello IGNOU !", 50, 25);

Now in this line drawString method is used to write "Hello Ignou!" message on the screen. It is a method of Graphic class. This drawString takes three arguments. First argument is a message and the other two are pixel positions on X-axis and Y-axis respectively, from where the printing of the string on the screen will begin.

Compiling and running MyFirstApplet.Java

When Java applet is ready, we can compile it using Java compiler named 'Javac'.

To compile this program write:
C:> Javac MyFirstApplet.Java

When compilation is over MyFirstApplet.class is created as an output of compilation. To execute the program you have to create a HTML file which includes the given applet class MyFirstApplet.class, and then run the applet using Web browser or applet viewer.

For example:

```
<HTML>
    <HEAD>
    <TITLE> A Simple Applet </TITLE>
    </HEAD>
    <BODY>
    Here is the output of my program:
    <APPLET CODE=" MyFirstApplet.class" WIDTH=150
    HEIGHT=50>
    </APPLET>
    </BODY>
</HTML>
```

When we open this HTML file using Internet Explorer, you will get window showing the following

Here, the string "Hello IGNOU!" is displayed inside a window of size 150 pixel wide and 50 pixel in height. You will find more detailed discussion on applet programming in Unit 1 Block 4 of this course.

After getting flavors of both application and applet programming you will be thinking that Java application programs and applets are similar but there are many differences between them. **What are these differences?** The differences between them are given below in *Table 1*.

Table 1: Main differences between Java applets and application programs

Java Application Programs	Java Applets
Java Application Program can be executed independently. Applications are executed at command line by Java.exe	Applet cannot be executed independently. Applets can only be executed inside a Java compatible container, such as a browser or appletviewer.
It contain main () method. It has a single point for entry to execution	It does not contain main () method, and does not have a single point of entry for execution.
Applications have no inherent security restrictions, and can perform read/write to files in local System.	Applets cannot perform read/write to files in local system This is to provide security.
Applications have no special support in HTML for embedding or downloading	Applets can be embedded in HTML pages and downloaded over the Internet

☞ Check Your Progress 2

1) Compile and Execute a Java Application that will display the statement " I am Learning Java".

……

……

2) Compile and run a Java Applet of WIDTH=250 and HEIGHT=100, applet will display a message " Soon I will write big programs in Java".

……

……

3) What are the different arguments of 'drawstring'?

……

……

Now you are able to write and execute your own simple programs but going in of programming concepts you need to know the different Keywords and data types available in Java. In the next two sections we will discuss different data types.

3.3 PRIMITIVE DATA TYPE AND VARIABLES

Java supports different primitive types given in *Table 2 where* each primitive data type *is given with its description, group and example.*

Table 2: Java Primitive Types

Data Type	Description	Example	Groups
Byte	Byte-length integer	10, 24	Integer
Short	Short integer	500, 786	
Int	Integer	89, 945, 37865	

Long	Long integer	89L, -945L	
Float	Single-precision floating point	89.5f, -32.5f,	Floating point numbers
Double	Double-precision floating point	89.5, -35.5, 87.6E45	
Char	A single character	‘c’, ‘9’, ‘t’	Characters
Boolean	A boolean value (0 or 1)	True or false	Boolean

In most of the languages, the format and size of primitive data types depend on the platform on which a code is running. But in Java programming language, size and format of its primitive data types is specified. So programmers need not worry about the system-dependencies while using data types. For example, an **int** is always 32 bits, regardless of the particular platform. This allows programs to be written that are guaranteed to run *without porting* on any machine architecture. While strictly specifying the size of an integer may cause a small loss of performance in some environments, it is necessary in order to achieve portability. All the primitive data types can be divided into four major groups as shown in *Table”.*

3.3.1 Java Keywords

A keyword or reserved word in Java has a special meaning and cannot be used as a user defined identifier, because they are used by the compiler and if you use them as variable names, compiler will generate errors. The following *Table 3* gives a list of Java keywords and their purpose in programming. These keywords cannot be used as variable name or function name in any Java program.

Table 3: List of Java keywords

Keyword	Purpose
abstract	It declares that a class or method is abstract
assert	Used to specify assertions
boolean	Declares that variable is Boolean type
break	Used to exit a loop before end of the loop is reached
byte	Declares that variable is byte type
case	To represent different conditions in switch statement
catch	Used for handling exceptions, i.e., capture the exceptions thrown by some actions
char	Declares that variable is character type
class	Signals the beginning of a class definition
const	This keyword is reserved by Java but now it is not in use
continue	Prematurely return to the beginning of a loop
default	Default action in switch statement
do	Begins a do while loop
double	Declares that variable is double type

else	Signals the code to be executed when if condition executes to false
extends	Specifies the class base from which the correct class is inherited
final	Declares that a class may not be extended or that a field or method may not be overridden
finally	Declares a block of code guaranteed to be executed
float	Declares a floating point variable
for	Start a for loop
goto	This keyword is reserved by Java but now it is not in use
if	Keyword to represent conditional statement
implements	Declares that this class implements the given interface
import	permits access to a class or group of classes in a package
Instance of	tests whether an object is an instance of a class
int	Declares an integer variable
interface	signals the beginning of an interface definition
long	Declares a long integer variable
native	Declares that a method that is implemented in native code
new	Allocates memory to an object dynamically
package	Defines the package to which this source code file belongs
private	Declares a method or member variable to be private
protected	Declares a class, method or member variable to be protected
public	Declares a class, method or member variable to be public
return	Returns a value from a method
short	Declares a short integer variable
static	Declares that a field or a method belongs to a class rather than an object
strictfp	To declare that a method or class must be run with exact IEEE 754 semantics
super	A reference to the parent of the current object
switch	Tests for the truth of various possible cases
synchronized	Indicates that a section of code is not thread-safe
this	A reference to the current object
throws	Declares the exceptions thrown by a method
transient	Data should not be serialized
try	Attempt an operation that may throw an exception

void	Declare that a method does not return a value
volatile	Warns the compiler that a variable changes asynchronously
while	Begins a while loop

It is important that Java is case-sensitive, so even though break is a keyword, Break is not a keyword at all.

For the C++ programmers case sensitivity in Java cause problems, but at the time of designing Java its designers decided to **make it case sensitive because** first of all this enhances the readability of cod, secondly it reduces the compilation time and increases the efficiency of compiler. All the Java technology keywords are in lower case. The words true, false, and null are reserved words. Keywords **'const'** and **'goto'** are currently not in use. In this list of Keywords the new Keywords have also been added, for exapmle **Java 1.2** adds the **strictfp** keyword to declare that a method or class must run with exact IEEE 754 semantics. **Java 1.4** adds the **assert** keyword to specify assertions. Each keyword has a specific and well-defined purpose. The following list explains the purpose and meaning of available keywords.

3.3.2 Integer and Floating Point Data Type

Integer & Floating-point numbers fall under Numeric datatypes. Java has six numeric datatypes that differ in size and precision of the numbers they can hold. The Size means how many bits are needed to represent the data type. The Range of a data type expresses the precision of numbers.

Integers

Integers are whole numbers. Depending on range, as given in *Table 4,* integer data type can be further divided into four categories:

Table 4: Four categories of Integer data types

Type	Values	Default	Size	Range
Byte	signed integers	0	8 bits	-128 to 127
Short	signed integers	0	16 bits	-32768 to 32767
Int	signed integers	0	32 bits	-2147483648 to 2147483647
Long	signed integers	0	64 bits	-9223372036854775808 to 9223372036854775807

Floating Point Numbers

A number containing a fractional part is called real number or floating point Number. As given in *Table 5*, Real Number can also be divided into two categories: float and double. But double is used where more accuracy is required for fractional part.

Table 5: Two categories of real numbers

Type	Values	Default	Size	Range

float	IEEE 754 floating point	0.0	32 bits	+/-1.4E-45 to +/- 3.4028235E+38, +/-infinity, +/-0, NAN
double	IEEE 754 floating point	0.0	64 bits	+/-4.9E-324 to +/- 1.7976931348623157E+308, +/-infinity, +/-0, NAN

3.3.3 Character and Boolean Types

Characters

A char is a single character that is a letter, a digit, a punctuation mark, a tab, a space or something similar. **A char** literal is a single one character enclosed in single quotes like
char myCharacter = ‘a’;
char doublequote = ‘ ” ’;

The character datatype, ***char***, holds a single character. Each character is a number or character code that belongs to a character set, an indexed list of symbols. For Example, **ASCII** (American Standard Code for Information Interchange) is a character set. In ASCII character set ranges from **0 to 127** and needs 8 bit to represent a character. Different attributes of character data type are given the *Table 6.*

Java uses the **Unicode** character set. Unicode defines a fully international character set that can represent all of the characters found in all human languages and writing systems around the world such as English, Arabic, Chinese, etc. Since there are a large number of languages, therefore a large set is a required and 8 bits, are not sufficient. This 16-bit character set fulfills the need. Thus, in Java **char** is a 16-bit type.

Table 6: Character data type

Type	Values	Default	Size	Range
Char	Unicode character	\u0000	16 bits	\u0000 to \uFFFF

Booleans

In Java’ Booleans are logical variables, which can contain the value either true or false. Any other value cannot be assigned to a Boolean variable. Different attributes of Boolean data type are given in *Table 7*.

Table 7: Boolean data type

Type	Values	Default	Size	Range
Boolean	True, False	False	1 bit used in 32 bit integer	NA

3.3.4 Declaring and Initialization Variables

Variables (Identifiers)

A variable is a basic unit of storage in a Java Program. Variables represent **memory location** in which value can be stored. Before using any variable, you have to declare it. After it is declared, you can then assign values to it (you can also declare and assign a value to a variable at the same time.)

Java actually has three kinds of variables:

- Instance variables
- Class variables
- Local Variables.

Instance variables are used to define the attributes of a particular object. Class Variables are similar to instance variables, except their values apply to all the instances of a class (and to the class itself) rather than having different values for each object. Local variables are declared and used inside methods, for example, for index counters in loops, temporary variables or to hold values that you need only inside the method definition itself.

Variable Declarations

Before using any variable, it must first be declared. A variable declaration specifies the datatype, the variable name and, optionally, the default value for the variable. A general variable declaration looks like the following:

datatype identifier [= default value] {, identifier [= defaultvalue] };
Consider the following examples for variable declaration:
byte b; short age;
boolean male;
Declare multiple variables of one type in one expression such as in the following example:
int age, enrollnum, numChildren;

Variable declarations can be put anywhere in your code, as long as they precede the first use of the variable. However, it is common practice to place the declarations at the top of each block of code. Variable names in Java can only start with a letter, an underscore (_), or a dollar sign ($). They cannot start with a number. After the first character, your variable names can include letters or numbers or a combination of both.

As you know, the Java language is case sensitive which implies that the variable x in lowercase is different from variable X in uppercase. In Java a variable name cannot start with a digit or hyphen. Variable names should help you understand what is happening in your program. Thus, it is useful to name your variables intelligently or according to its role in the program

The main restriction on the names you can give your variables is that they cannot contain any white space. There is no limit to the length of a Java variable name.

If you want to begin variable name with a digit, prefix underscore, with the name, e.g. _9cats. You can also use the underscore to act like a space in long variable names.

Variable Assignment and Initialization

Once you have declared the type of a variable, you can initialize it with some value.
variable name = some value.
For example, consider the following example:
int year;
year = 28; // here the value 28 is assigned to the variable ‘year’

Dynamic Initialization

We have seen how variables can be declared and assigned a value. However, we can declare and assign a value to the variable in a single statement. This is called **Dynamic initialization**. Variables can be dynamically initialized using any expression valid at the time the variable is declared. This can be made clearer by looking at the example code given below:
// Demonstrate dynamic initialization.

/* This Program calculates the hypotenuse of a triangle with height of the triangle, h base b, using formula: square root of [square of height h + square of base b] */

```
class DynamicInitialization   {
    public static void main (String args [ ] )
                        {
                          double  h = 3.0, b = 4.0;
                          double  c = Math.sqrt(h*h + b*b);   // here c is dynamically
                          initialized
                          System.out.println (" Hypotenuse is " + c);
                        }
                    )
```

Here, in this code, the variables h and b are declared and initialized to some values in a single statement. Then you find in the next line that another variable c, is declared as double and is assigned the output of an expression. The expression uses the sqrt() function which is one of the Java's built-in methods of the Math class.

The key point here is that the initialization expression may use any element valid at the time of initialization, including calls to methods, other variables, or literals.

Literals

Literals are nothing but pieces of Java code that indicate explicit values. For example "Hello IGNOU!" is a String literal. The double quote marks indicate to the compiler that this is a string literal. The quotes indicate the start and the end of the string, but remember that the quote marks themselves are not a part of the string. Similarly, Character Literals are enclosed in single quotes and it must have exactly one character. TRUE and FALSE are boolean literals that mean true and false. Number, double, long and float literals also exits there. See examples of all types of literals in the following *Table 8*.

Table 8: Examples of literals

Types of literal	Example
Number Literals	-45, 4L, 0777, 0XFF, 2.56F, 10e45, .36E-2
Boolean Literals	TRUE, FALSE
Character Literals	'a', '#', '3', \n, \\, \"
String Literals	"A string with a \t tab in it"
Double Literals	1.5, 45.6, 76.4E8
Long Literals	34L
Float Literals	45.6f, 76.4E8F, 1.5F

Constants

Constants are used for fixed values. Their value never changes during the program execution. To declare constants in Java keyword final is used. The following statement defines an integer constant x containing a value of 10.

```
final int x =10;
```

Identifiers

In simple words Identifiers are nothing but the names of variables, methods, classes, packages and interfaces. In the Application program, String, args, main and System.out.println are identifiers. Identifiers must be composed of letters, numbers, the underscore '_' and the dollar sign $. But identifiers should begin with a letter, the underscore or a dollar sign.

☞ Check Your Progress 3

1) Write a program in Java to calculate the area and the circumference of a circle. Show the use of 'final' keyword in your program

……

……

……

……

2) What are the kinds of variables in Java? What are their uses?

……

……

……

……

3) Why are the following are unacceptable as Java integer constant.

   a. 2,145

   b. –8.62

   c. 146E32

   d. 54-

4) Why is abstract keyword but Abstract is not?

……

……..……..

……

……

In the next section we will explore the different types of operators in Java with their meaning and uses in the programs.

3.4 JAVA OPERATORS

Arithmetic operators

You must be friendly with arithmetic expressions in mathematics like 'A – B'. In this expression A and B are operands and the subtraction sign ' – ' is the operator. The same terminology is also used here in the programming language. The following table lists the basic arithmetic operators provided by the Java programming language along with the description and uses. Here in the Table A and B are the operands. Operators

are divided into two categories **Binary and Unary.** Addition, subtraction, multiplication etc. are binary operators and applied only on two operands. Increment and decrement are unary operators and applied on single operand.

Table 9A: Description of arithmetic operators

Binary Operators		
Operator	**Use**	**Description**
+	A + B	Adds A and B
-	A - B	Subtracts B from A
*	A * B	Multiplies A by B
/	A / B	Divides A by B
%	A % B	Computes the remainder of dividing A by B

Unary Operators		
Operator	**Use**	**Description**
++ "Post-increment"	A++	Post-increment: The value is assigned before the increment is made, e.g. A = 1; B = A++; Then B will hold 1 and A will hold 2
-- "Post-decrement"	A--	Post-decrement: The value is assigned before the decrement is made, e.g. : A = 1; B = A--; Then B will hold 1 and A will hold 0.
++ "Pre-increment"	++A	Pre-increment: The value is assigned after the increment is made, e.g. A = 1; B = ++A; Then B will hold 2 and A will hold 2.
-- "Pre-decrement"	--A	Pre-decrement: The value is assigned after the decrement is made, e.g. A = 1; B = --A; Then B will hold 0 and A will hold 0.

Maybe you are thinking here that in the list we don't have any operator for exponentiation. If you want to do exponentiation you should import Java.lang class where math is subclass. Inside math class you have one function 'pow' similar to C++ which can be used for exponentiation. All the above operators can be used only on numeric values except for +, which is also used to concatenate strings.

Assignment Operators

The basic assignment operator '=' is used to assign value to the variables. For example A = B; in which we assign value of B to A. Similarly, let us see how to assign values to different data types.

int Integer = 100; float Float = 10.5; char Chararcter = 'S'; boolean aboolean = true;

With basic assignment operator, the Java programming language defines short cut assignment operators that allow you to perform arithmetic, shift, or bitwise operation with one operator.

Now let us see, if you want to add a number to a variable and assign the result back into the same variable, so you will write something like i = i + 2; but using shortcut operator '+=' You can shorten this statement, like i += 2. But remember, i = i + 2; and i += 2; both statements are the same for the compiler. As given in *Table10*, Java programming language provides different *short cut assignment operators*:

Table 10: Use of short cut assignment operators

Assignment Operators		
Operator	**Use**	**Equivalent to**
+=	A += B	A = A + B
-=	A -= B	A = A – B
*=	A *= B	A = A * B
/=	A /= B	A = A / B
%=	A %= B	A = A % B
&=	A &= B	A = A & B
|=	A |= B	A = A | B
^=	A ^= B	A = A ^ B
<<=	A <<= B	A = A << B
>>=	A >>= B	A = A >> B
>>>=	A >>>= B	A = A >>> B

Multiple Assignments

Multiple assignments are possible in the following format:
Identifier= Identifier B= Identifier C=…….=expression
In this case all the identifiers are assigned the value of expression. Lets see one example:

A=B=786+x+y+z; this is the same if you write: A=786+x+y+z; and B=786+x+y+z;

Relational Operators

Relational operators also called, comparison operators, are used to determine the relationship between two values in an expression. As given in *Table 11*, the return value depends on the operation.

Table 11: Use of relational operators

Relational operators		
Operator	**Use**	**Returns true if**
>	A > B	A is greater than B
>=	A >= B	A is greater than or equal to B

<	A < B	A is less than B
<=	A <= B	A is less than or equal to B
==	A == B	A and B are equal
!=	A != B	A and B are not equal

Boolean Operators

Boolean operators allow you to combine the results of multiple expressions to return a single value that evaluates to either true or false. *Table 12* describes the use of each boolean operator.

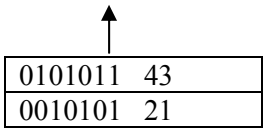
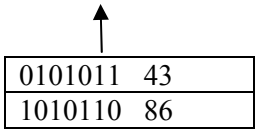
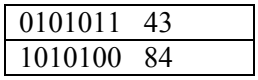
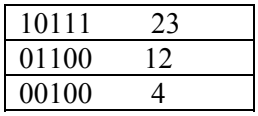
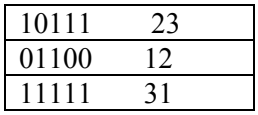
Table 12: Description of Boolean operators

Boolean Operators		
Operator	**Use**	**Description**
&&	A && B	Conditional AND :If both A and B are true, result is true. If either A or B are false, the result is false. But if A is false, B will not be evaluated. For example, (A > 4 && B <= 10) will evaluate to true if A is greater than 4, and B is less than or equal to 10.
||	A || B	Conditional OR: If either A or B are true, the result is true. But if A is true, B will not be evaluated. For example, (A > 10 || B > 10) will evaluate to true if either A or B are greater than 10.
!	! A	Boolean NOT: If A is true, the result is false. If A is false, the result is true.
&	A & B	Boolean AND: If both A and B are true, the result is true. If either A or B are false, the result is false and both A and B are evaluated before the test.
|	A | B	Boolean OR: If either A or B are true, the result is true. Both A & B are evaluated before the test.
^	A ^ B	Boolean XOR: If A is true and B is false, the result is true. If A is false and B is true, the result is true. Otherwise, the result is false. Both A and B are evaluated before the test.

Bitwise operators

As you know in computers data is represented in binary (1's and 0's) form. The binary representation of the number 43 is 0101011. The first bit from the right to left in the binary representation is the least significant bit, i.e. here value is 1. Each Bitwise operator allows you to manipulate integer variables at bit level. *Table 13* given below describes the use of bitwise operator with help of suitable example for each. Similarly, the use of class and object operators is given in *Table 14.*

Table 13: Description and use of Bitwise operators

Bitwise operators		
Operator	**Use**	**Description**
>>	A >> B	Right shift: Shift bits of A right by distance B, 0 is introduced to the vacated most significant bits, and the vacated least significant bits are lost. The following shows the number 43-shifted right once (42 >> 1). 
<<	A << B	Left shift: Shift bits of A left by distance B , the most significant bits are lost as the number moves left, and the vacated least significant bits are 0. The following shows the number 43 shifted left once (42 << 1). 
>>>	A >>> B	Right shift unsigned: Shift A to the right by B bits. Low order bits are lost. Zeros fill in left bits regardless of sign example.
~	~B	Bitwise complement: The bitwise Complemnt changes the bits. 1, into 0, and bit 0, to 1. The following shows the complement of number 43. 
&	A & B	Bitwise AND: AND is 1 if both the bits are 1 otherwise AND is 0. The bitwise AND is true only if both bits are set. Consider 23 & 12: 
|	A | B	Bitwise OR: The bitwise OR is true if either bits are set. Or if the bit is 1 if either of the two bits is 1, otherwise it is 0. Consider 23 | 12: 

^	A ^ B	Bitwise Exclusive OR: The bitwise Exclusive OR is true if either bits are set, but not both. XOR of the bits is 1 if one bit is 1 & other bit is 0, otherwise, it is 0. Consider 23 ^ 12: 10111 23 01100 12 11011 27

Class and Object Operators

Table 14: Class and Object Operators

Operator	Name	Description
instance of	Class Test Operator	The first operand must be an object reference. For example 'A instance of B', Returns true if A is an instance of B. Otherwise, it return false.
new	Class Instantiation	Creates a new object. For example: new A, in this A is either a call to a constructor, or an array specification.
"."	Class Member Access	It accesses a method or field of a class or object. For example A.B used for 'field access for object A' and A.B() used for 'method access for object A'
()	Method Invocation	For example: A(parameters) , Declares or calls the method named A with the specified parameters.
(type)	Object Cast	(type) A, in this example () operator Cost (convert) A to specific type. An exception will be thrown if the type of A is incompatible with specified type. In this type can be object or any primitive data type.

Other Operators

There are some other operators that cannot fit in the above categories but are very important for programmers. These operators are explained in *Table 15* given below.

Table 15: Other operators

Operator	Use	Description
?:	A ? B : C	If A is true, returns B. Otherwise, returns C.
[ ]	type []	Declares an array of unknown length, which contains type elements.
[ ]	type[A]	Creates and array with A elements. Must be used with the new operator.
[ ]	A[B]	Accesses the element at index B within the

		array A. Indices begin at 0 and extend through the length of the array minus one.
+	A+B	This binary operator concatenates one string to another. For example: String str1 = "IG"; String str2 = "NOU"; String str3 = str1 + str2 results in str3 holding "IGNOU".

☞ Check Your Progress 4

1) What is a literal? How many types of literals are there in Java?

……

……

2) What is synchronization and why is it important?

……

……

3) What is the difference between the Boolean & operator and the & operator?

……

……

……

……

4) What is the difference between a “compiler” and an “interpreter”?

……

……

……

3.5 SUMMARY

In this unit you learnt that Java was developed by SUN Microsystems in 1991 under the guidance of James Gosling. It’s an Object Oriented, general-purpose programming language. After its birth it became popular because of many reasons *like* security, robustness and multithreadedness but mainly because of its characteristic of *Architecture Neutral and platform independent*. The logic and magic behind its platform independence is “BYTECODE”. Java offers two flavors of programming, Java application and Java applet. Application can be executed independently while applet cannot be executed independently. When we compile a Java program we get Java bytecode (.class file) that can be executed on any other computer system, equipped with Java interpreter. To execute applets you can use applet viewer or explorer to run Java embedded HTML code.

You learnt the meaning of different Java keywords which have special meaning for compiler and cannot be used as a user defined identifier. Java supports eight primitive data types which can be classified into four categories. Each data type has its memory size and range of values it can store. Before using the Java variables in your program you should declare them and assign them some value. You learnt how to declare a variable dynamically with the help of an example. You have learnt different types of operators in the last section of this unit which are similar to C++ so you must be familiar with some of them. You must be tired after reading a long unit so now you can go and have a cup of coffee. In the next unit we will discuss about the classes and objects in Java.

3.6 SOLUTIONS /ANSWERS

Check Your Progress 1

1) Java Virtual Machine plays very important role in making Java portable and it executes the byte code generated by Java compiler. Java compiler translates Java program into a form called byte code. This byte code is kind of machine language for hypothetical machine or virtual machine. The name given 'virtual' is analysis to old system where compiler generates language which is understandable by hardware. In Java, compiler generates a language which is understandable by an imaginary machine which does not exist in hardware but is a software machine.

2) Java was designed for networking and a distributed environment so security was a major issue at the time of its development. Following are the two main characteristics that make Java secure.

- All the references to memory are 'symbolic references' which means that users do not know where the program is residing in memory and allocation of memory to program is totally handled by JVM of each machine.
- Java applets execute in runtime environment that restrict the intruder applet to spread virus and deleting and modifying files in host machine.

3) After developing Java language its team wanted to give a suitable name. They used to see an oak tree from their window and they thought to give the same name "Oak" to their new born language.

Check Your Progress 2

1) First of all using the text editor type Java source file of your program as given below. Save the file as Display Message.Java .

```
/* This is a program for simple display of message on output terminal */
class DisplayMessage
    {
     public static void main(String args[ ])
        {
           System.out.println("I am Learning Java ");
        }
    }
```

Now compile the source file-using compiler named Javac that takes your source file and translates its statements into a bytecode file.

C:> Javac DisplayMessage.Java

Run the program using interpreter named Java that takes your bytecode file and translates it them into instructions that our computer can understand.

C:> Java Display Message

2)

i. Write the following Java source code into text editor and save as BigProgram.Java

```
import Java.applet.Applet;
import Java.awt.Graphics;
public class BigProgram extends Applet {
public void paint(Graphics g) {
```

```
        g.drawString("Soon I will write big programs in Java ", 50, 25);
                                }
    .                                   }
```

ii. Compile the source file: Javac BigProgram.Java (This will generate BigProgram.class)

iii. Execute the Applet: include the code in HTML code.

```
HTML>
        <HEAD>
<TITLE> A Big Program </TITLE>
</HEAD>
<BODY>
Here is the output of my program:
<APPLET CODE=" BigProgram.class " WIDTH=250 HEIGHT=100>
</APPLET>
</BODY>
</HTML>
```

3) This drawString takes three arguments the first is message and the other two are pixal positions X-axis and Y-axis respectively.
For example; g.drawString("Hello IGNOU !", 50, 25);
Here, in the example," Hello IGNOU! Message is first argument, 50 and 25 are other two arguments that define X-position and Y-position where the message will be displayed.

Check Your Progress 3

1) If we get the error " Exception in thread "main"

Java.lang.NoClassDefFoundError", this means Java is unable to find bytecode file. Java tries to find bytecode file in current directory. So, if bytecode file is in C:\Java, we should change our current directory to that. The prompt should change to C:\Java>.

If still there is a problems, in this case may be CLASSPATH environment variable is not set up properly. You can set your CLASSPATH environment variable using DOS command like

C:\> SET CLASSPATH=C:\JDK\JAVA\CLASSES;c:\Java\lib\classes.zip

// program to calculate area of circle and circumference of circle.

```
class AreaTriangle   {
    public static void main (String args [ ] )
                    {
                       final double  pi = 3.14159;  // pi is defined as final here
                         double  radius = 4.0;
                         double  circumference = 2*pi*radius;   // here c is dynamically
                         initialized double  area = pi*radius*radius;
                         System.out.println ("circumference of circle of radius 4 unit is
                         " + circumference);
                         System.out.println ("Area of circle of radius 4 unit is " +area);
                }
               }
```

2) Java has three kinds of variables, namely, the instance variable, the local variable and the class variable.

- Local variables are used inside blocks as counters or in methods as temporary variables to store information needed by a single method.

- Instance variables are used to define attributes or the state of a particular object and are used to store information needed by multiple methods in the objects.

- Class variables are global to a class and to all the instances of the class. They are useful for communicating between different objects of the same class or keeping track of global states.

3) The given integer constants are unacceptable because:

   a. The comma is not allowed in the integer.
   b. The decimal point is not allowed in integers constants.
   c. The character E is not allowed in integer constant.
   d. The symbol '–' can appear in front of integer constant not in last or in-between.

4) Because Java is case-sensitive, so even though abstract is a keyword but Abstract is not a keyword at all.

Check Your Progress 4

1) A literal represents a value of a certain type where the type describes how that value behaves.

There are different types of literals namely number literals, character literals, boolean literals, and string literals.

2) With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

3) If an expression involving the Boolean & operator is evaluated, both operands are evaluated. Then the && operator is applied to the operand. When an expression involving the & operator is evaluated, the first operand is evaluated. If the first operand returns a value of true then the second operand is evaluated. The && operator is then applied to the first and second operands. If the first operand evaluates to false, the evaluation of the second operand is skipped.

4) Compilers and interpreters have similar functions: They take a program written in some programming language and translate it into machine language. A compiler does the translation all at once. It produces a complete machine language program that can then be executed. An interpreter, on the other hand, just translates one instruction at a time, and then executes that instruction immediately. (Java uses a compiler to translate Java programs into Java Bytecode, which is a machine language for the imaginary Java Virtual Machine. An interpreter then executes Java Bytecode programs.)