Chapter-2
Language Reference

Introduction

We can develop two types of java programs:

- **Stand-alone applications:**are programs written in Java to carry out certain tasks on a standalone local computer.

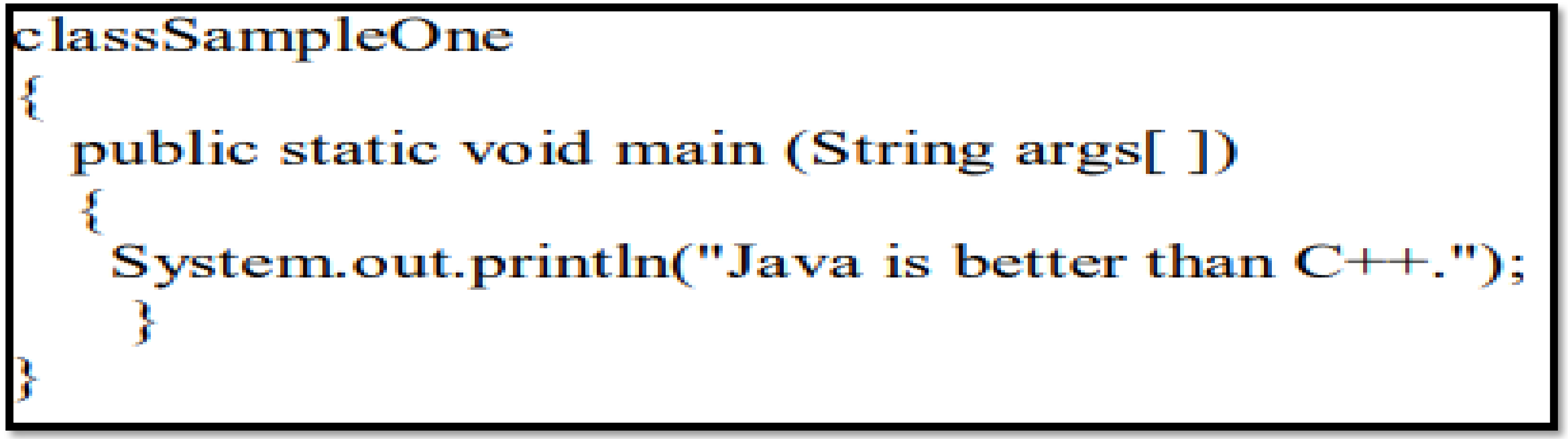
Executing a stand-alone Java program involves two steps:

1. Compiling source code into byte code using javac compiler.
2. Executing the bytecode program using java interpreter.

- **Web applets :** Applets are small Java programs developed for Internet applications. An applet located on a distant computer (Server) can be downloaded via Internet and executed on a local computer (Client) using a Java-capable browser. We can develop applets for doing everything from simple animated graphics to complex games and utilities.

- **Stand-alone programs** can read and write files and perform certain operations that applets cannot do.
- **An applet** can only run within a Web browser.

A simple Java program:



```
classSampleOne
{
  public static void main (String args[ ])
   {
    System.out.println("Java is better than C++.");
     }
}
```

Class Declaration : The first line .

Class Sample One : Declares a class, which is an object-oriented construct.

class is a keyword and declares that a new class definition. **SampleOne** is a Java identifier.

Opening Brace: Every class definition in Java begins with an opening brace "{" and ends with a matching closing brace"}" appearing in the last line.

The main Line: The third line .

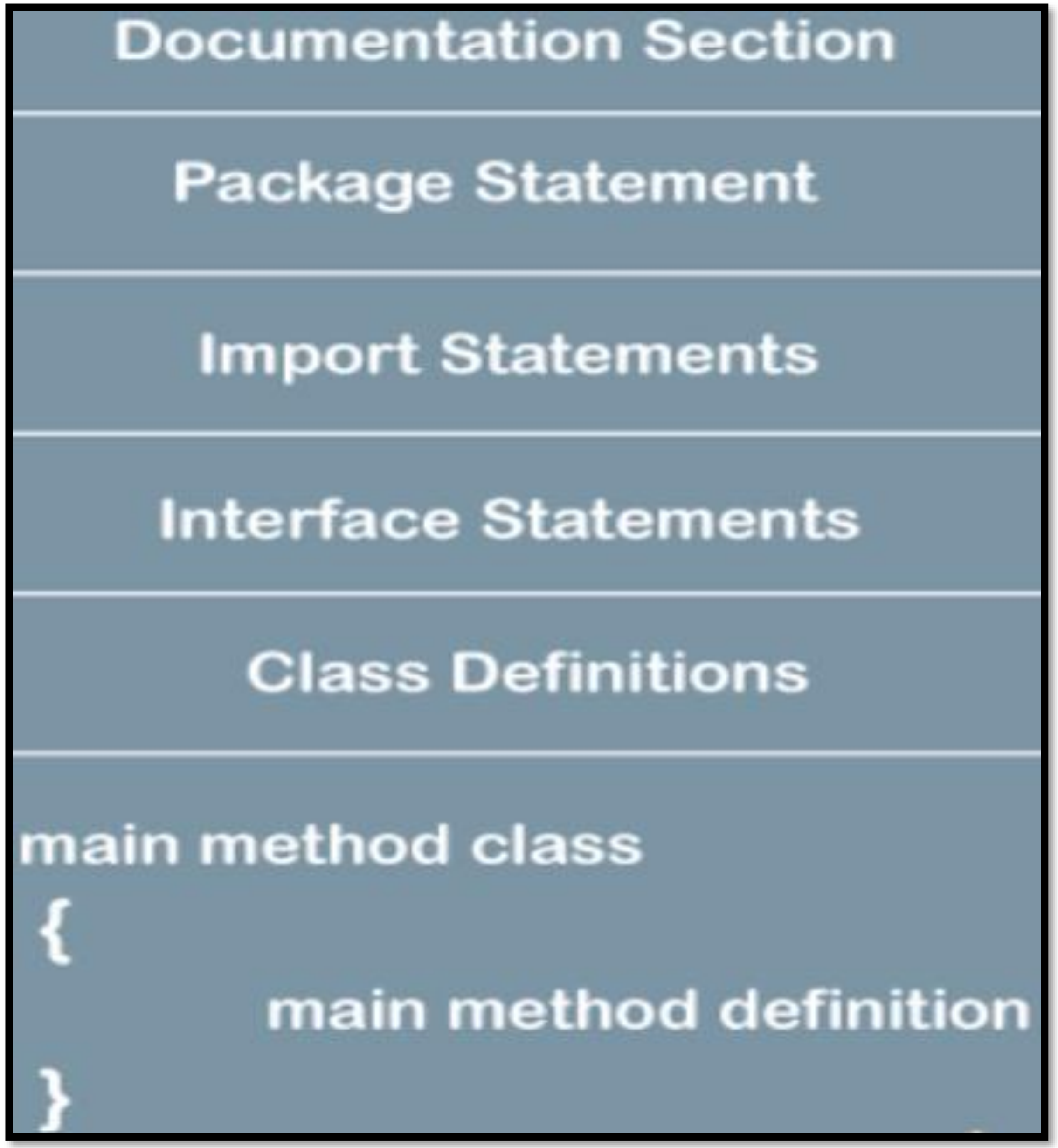
<u>public static void main (String args[ ]) :</u>

- Defines a method named main. Every Java application program must include the main( ) method.
- This is the starting point for the interpreter to begin the execution of the program.
- This line contains a number of keywords, public, static and void.

<u>The Output Line :</u>

- The only executable statement in the program is System.out.println("Java is better than C++.");
- The println method prints the string Java is better than C++ to the screen.
- Every Java statement must end with a semicolon.

JAVA PROGRAM STRUCTURE:



JAVA PROGRAM STRUCTURE:

Documentation Section:

- The documentation section comprises a set of comment lines giving the name of the program, the author and other details,.
- Java permits both the single-line comments and multi-line comments.
- The single-line comments begin with // and end at the end of the line.
- multi-line comments begins with /* and ending with*/.
- /**…............*/ known as documentation comment. This form of comment is used for generating documentation automatically.

Package Statement :

- The first statement allowed in a Java file is a package statement.
- This statement declares a package name and informs the compiler that the classes defined here belong to this package.
- **For example:** package student; The package statement is optional.

Import Statements:

- The next thing after a package statement (but before any class definitions) may be a number of import statements.
- For example import student test. This statement instructs the interpreter to load the test class contained in the package student.

Interface Statements :

- An interface is like a class but includes a group of method declarations.
- This is also an optional section and is used only when we wish to implement the multiple inheritance features in the program.

Class Definitions:

- A Java program may contain multiple class definitions. Classes are the primary and essential elements of a Java program.

Main Method Class :

- Since every Java stand-alone program requires a main method as its starting point, this class is the essential part of a Java program.
- A simple Java program may contain only this part.

JAVA TOKENS:

- Smallest individual units in a program are known as tokens.
- In simple terms, a Java program is a collection of tokens, comments and white spaces.

Java language includes five types of tokens. They are:

Keywords

- Keywords are an essential part of a language definition. They implement specific features of the language.
- Java language has reserved 60 words as keywords.
- These keywords, combined with operators and separators according to syntax, form definition of the Java language
- All keywords are to be written in lower-case letters.

Identifiers

- Identifiers are programmer-designed tokens.
- They are used for naming classes, methods, variables, objects, labels, packages and interfaces in a program.

Java identifiers follow the following rules:

1. They can have alphabets, digits, and the underscore and dollar sign characters.
2. They must not begin with a digit.
3. Uppercase and lowercase letters are distinct.
4. They can be of any length.

Literals :

- Literals in Java are a sequence of characters (digits, letters, and other characters) that represent constant values to be stored in variables.

Java language specifies five major types of literals. They are:

Integer literals

Floating point literals

Character literals

String literals

Boolean literals

Operators :

An operator is a symbol that takes one or more arguments and operates on them to produce a result.

Separators:

Separators are symbols used to indicate where groups of code are divided and arranged.

They basically define the shape and function of our code.

Some of the separators are : **parentheses( ), braces{ },brackets[ ], semicolon; comma , period.**

Java statements :

A statement is an executable combination of tokens ending with a semicolon (;) mark.

1. Empty Statements:

- It is used when the syntax of a statement is required, but no action is desired or necessary.

2. Labeled Statements:

- A labeled statement in Java is a statement preceded by an identifier (label) and a colon.
- Labeled statements are used with break and continue statements to specify which loop or switch statement should be affected.

3. Expression Statements:

- The purpose of an expression statement is to evaluate the expression, and the result is often used for its side effects.

4. Iteration Statements:

- Iteration statements, also known as looping statements or loops, allow you to repeatedly execute a block of code as long as a certain condition is true.

5. Selection Statements:

- Selection statements in Java allow you to control the flow of your program based on certain conditions.

6. Jump Statements:

- **Jump** statements in java are break, continue, return and throw. They are used to alter the normal flow of a program.

7. Synchronization Statements:

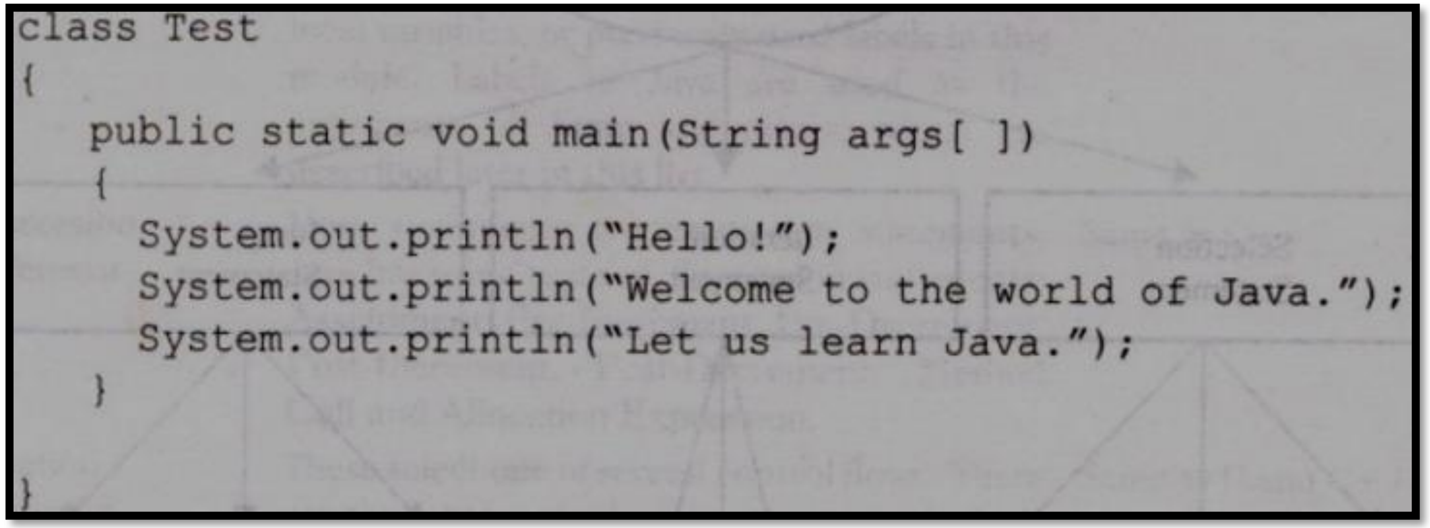
- In Java, synchronization is essential when dealing with multithreading to ensure that multiple threads can access shared resources in a coordinated way.

IMPLEMENTING A JAVAPROGRAM

- Creating the program
- Compiling the program
- Running the program

Creating the Program:

- We can create a program using any text editor. Assume that we have entered the following code



```
class Test
{
   public static void main(String args[ ])
   {
     System.out.println("Hello!");
     System.out.println("Welcome to the world of Java.");
     System.out.println("Let us learn Java.");
   }
}
```

- We must save this program in a file called Test.java ensuring that the filename contains the class name properly.
- This file is called the source file. All Java source files will have the extension java.
- If a program contains multiple classes, the file name must be the class name of the class containing the main method.

Compiling the Program:

To compile the program, we must run the Java Compiler javac, with the name of the source file on the command line

as javac Test.java . If everything is OK, the javac compiler creates a file called Test.class containing the bytecodes of the program.

The compiler automatically names the bytecode file as .class

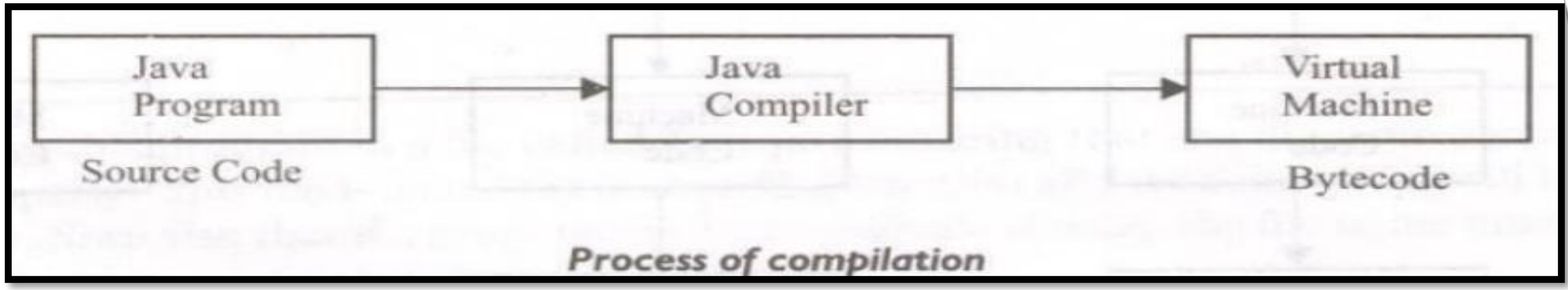
Running the Program :

We need to use the Java interpreter to run a stand-alone program. At the command prompt, type java Test. When executed program

displays the following: Hello! Welcome to the world of Java. Let us learn Java.

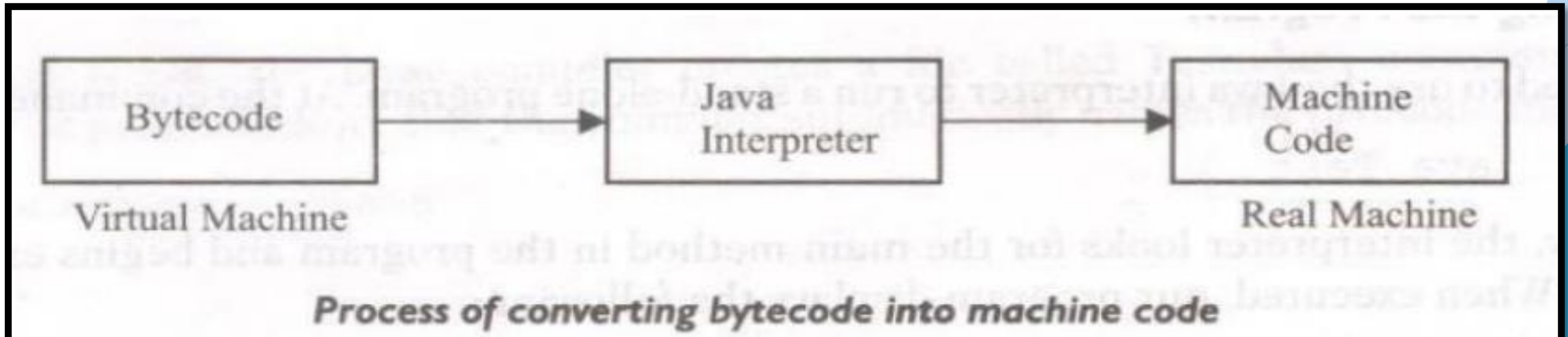
JAVA VIRTUAL MACHINE:

- All language compilers translate source code into machine code for a specific computer. Java compiler also does the same thing.
- The Java compiler produces an intermediate code known as bytecode for a machine that does not exist. This machine is called the Java Virtual Machine and it exists only inside the computer memory.
- It is a simulated computer within the computer and does all major functions of a real computer.



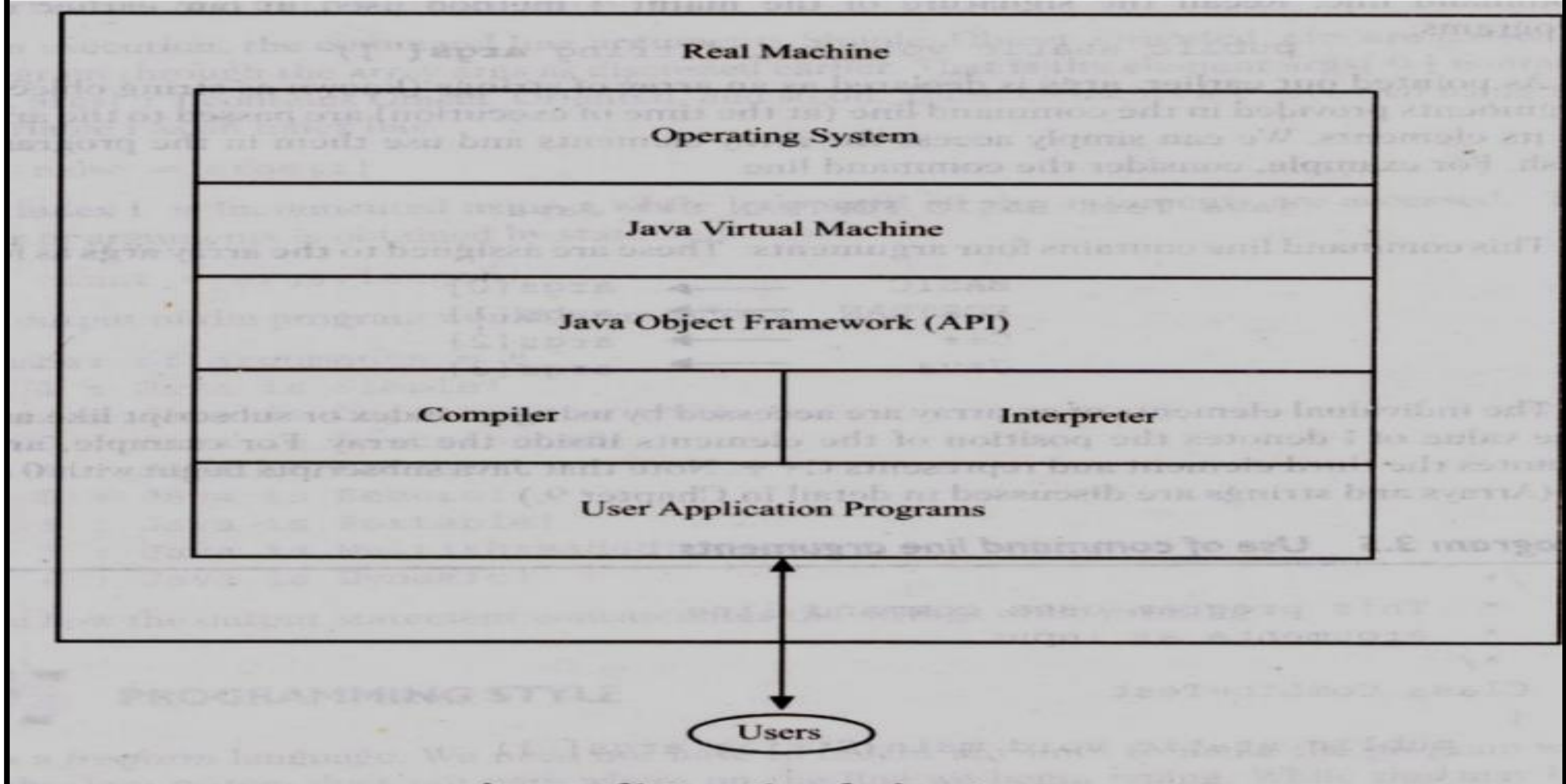
Process of compilation

- The virtual machine code is not machine specific. The machine code is generated by the Java interpreter by acting as an intermediary between the virtual machine and the real machine.



Process of converting bytecode into machine code

The Java object framework (Java API) acts as the intermediary between the user programs and the virtual machine which in turn acts as the intermediary between the operating system and the Java objectframework.

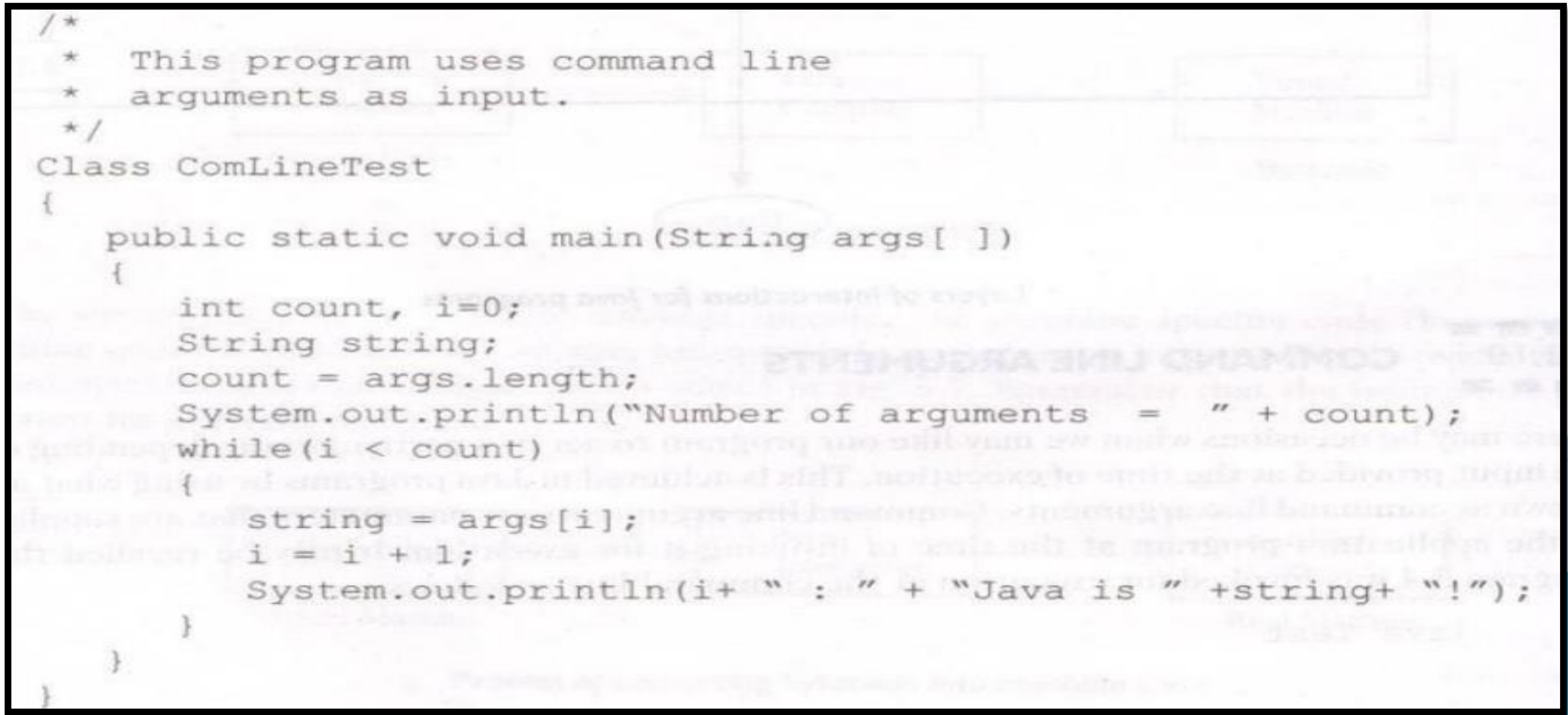


Layers of interactions for Java programs

COMMAND LINE ARGUMENTS:

- There may be occasions when we may like our program to act in a particular way depending on the input provided at the time of execution.
- This is achieved in Java programs by using what are known as **command line arguments.**
- Command line arguments are parameters that are supplied to the application program at the time of invoking it for execution.
- We can write Java programs that can receive and use the arguments provided in the command line.

Example:



```
/*
 *  This program uses command line
 *  arguments as input.
 */
Class ComLineTest
{
    public static void main(String args[ ])
    {
        int count, i=0;
        String string;
        count = args.length;
        System.out.println("Number of arguments  =  " + count);
        while(i < count)
        {
            string = args[i];
            i = i + 1;
            System.out.println(i+ " : " + "Java is " +string+ "!");
        }
    }
}
```

Compile and run the program with the command line as follows:

```
JavaComLineTest Simple Object_Oriented Distributed Robust Secure Portable Multithreaded Dynamic .
```

The number of arguments is obtained by statement count = args.length;

The output of the program would be as follows:

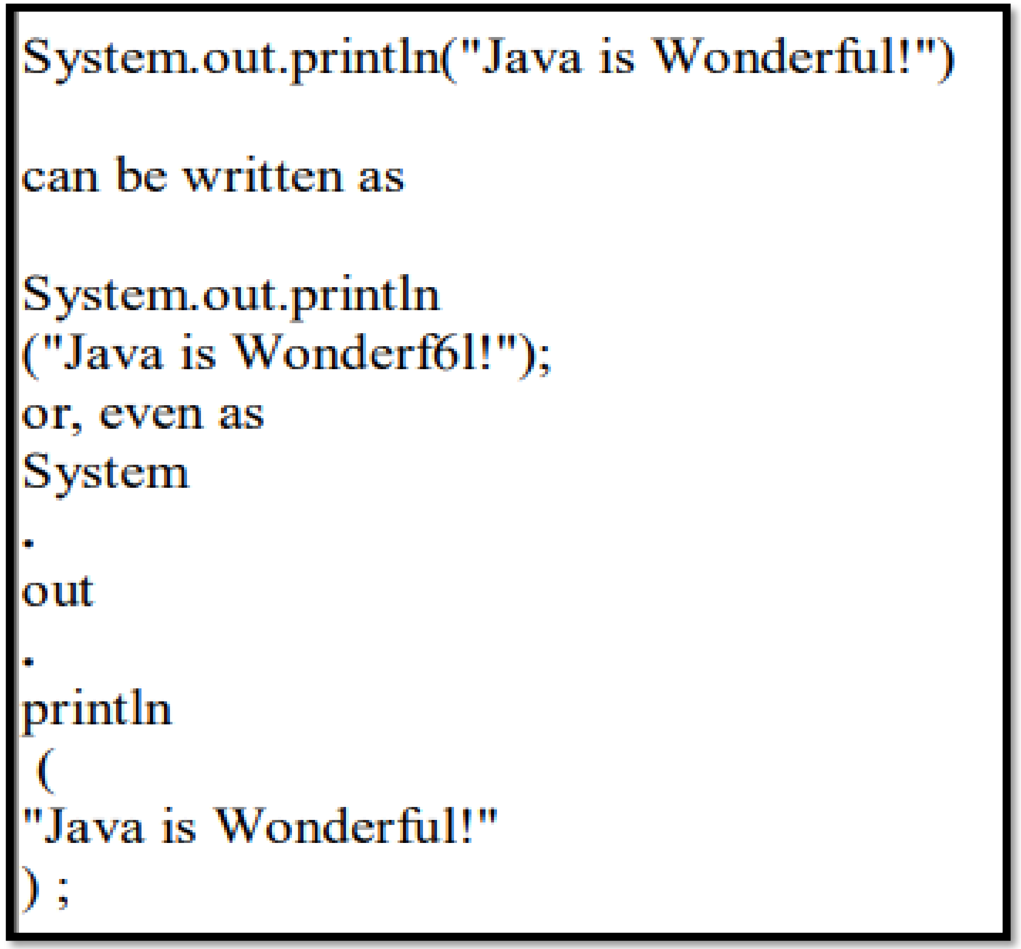
```
Number of arguments = 8
1 : Java is Simple!
2 : Java is  Object_Oriented!
3 : Java is Distributed!
4 : Java is Robust!
5 : Java is Secure!
6 : Java is Portable!
7 : Java is Multithreaded!
8 : Java is Dynamic!
```

Note how the output statement concatenates the strings while printing.

PROGRAMMING STYLE :

- Java is a freeform language.
- We need not have to indent any lines to make the program work properly.
- Java system does not care where on the line we begin typing.

Example:



```
System.out.println("Java is Wonderful!")

can be written as

System.out.println
("Java is Wonderf6l!");
or, even as
System
.
out
.
println
 (
"Java is Wonderful!"
) ;
```

Constants variables and Data Types

CONSTANTS:

- Constants in Java refer to fixed values that do not change during the execution of a program.

Integer Constants: An integer constant refers to a sequence of digits.

There are three types of integers:

1. **Decimal integers**

Examples:123   -321   0   654321

Octal integer:

- combination of digits from the set 0 through 7, with a leading O.
- **Examples :** 037 0 0435 0551

Hexadecimal integer :

- A sequence of digits preceded by Ox or OX .
- Include alphabets A through F or a through f.
- A through F represents the numbers 10 through 15.

Examples: OX2 OX9F Oxbcd Ox.

Real Constants /floating point Constants:

- Contain fractional parts like 17.548.

Examples :

0.0083 -0.75 435.36

A floating point constant has four parts:

(1)A whole number

(2)A decimal point

(3)A fractional part

(4)An exponent

Single Character Constants:

A single character contains a single character enclosed within a pair of single quote marks.

Examples:’5’ 'X'

String Constants:

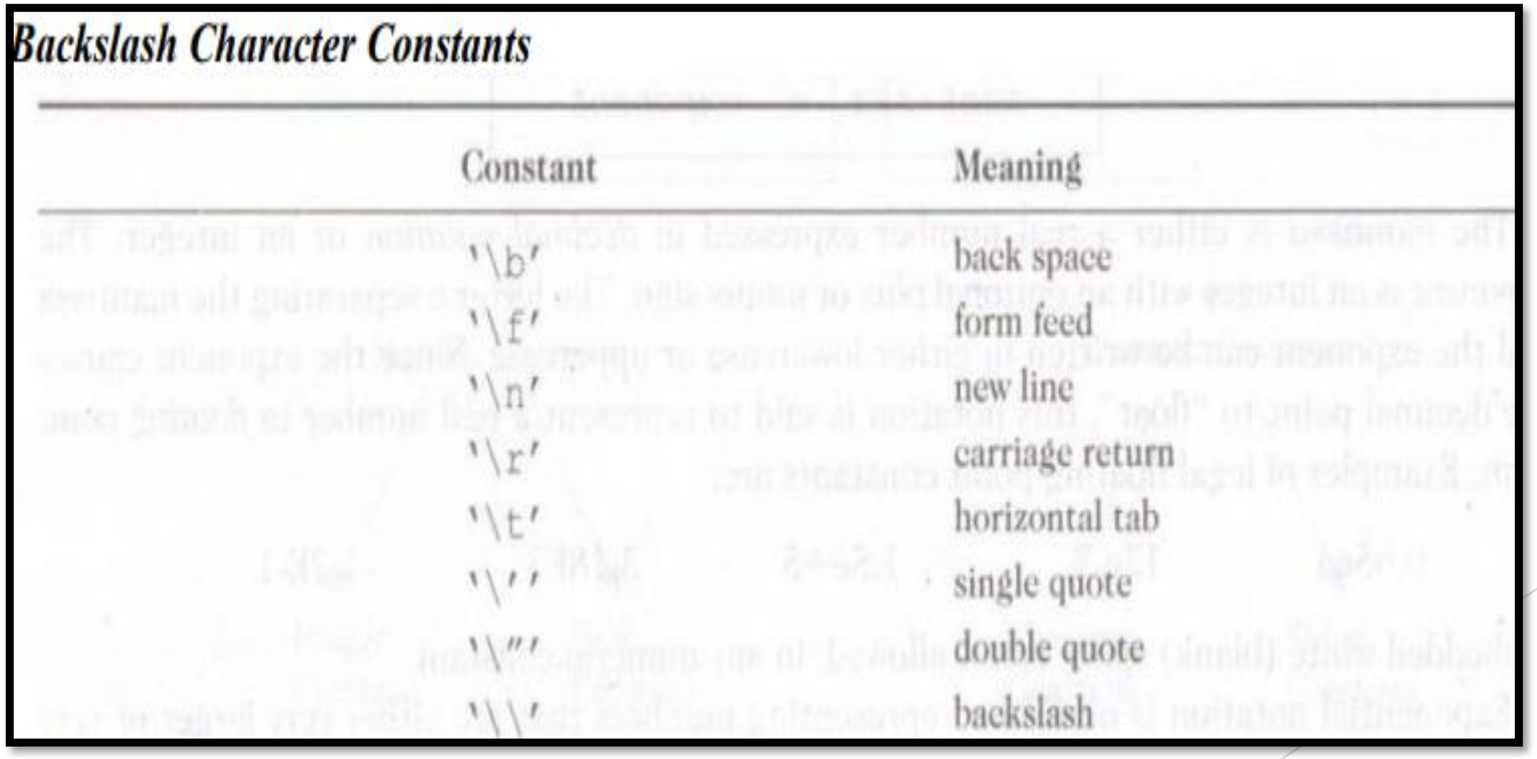
A string constant is a sequence of characters enclosed between double quotes.

The characters may be alphabets, digits, special characters and blank spaces.

Examples : "Hello Java" "1997" "WELL DONE" "?..!" "5+3""X“

Backslash Character Constants :

Java supports some special backslash character constants that are used in output methods.



Backslash Character Constants

Constant	Meaning
'\b'	back space
'\f'	form feed
'\n'	new line
'\r'	carriage return
'\t'	horizontal tab
'\''	single quote
'\"'	double quote
'\\'	backslash

Data Types :

- Every variable in Java has a data type. Data types specify the size and type of values that can be stored.

Integer Types:

- Integer types can hold whole numbers such as 123, -96, and 5639.
- The size of the values that can be stored depends on the integer data type we choose.

Floating Point Types:

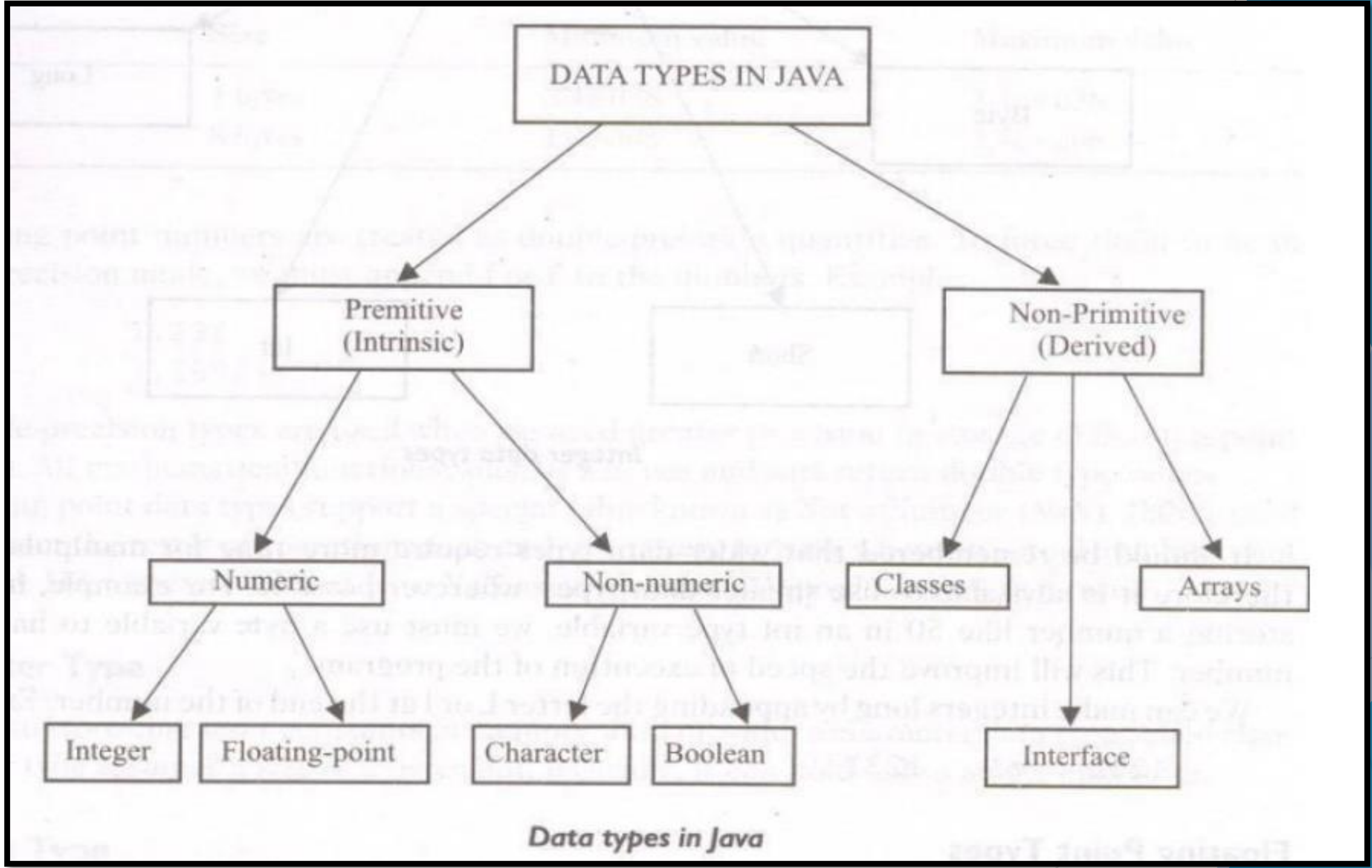
- Integer types can hold only whole numbers so we use another type known as floating point type to hold numbers containing fractional parts such as 27.59 and -1.

Character Type:

- In order to store character constants in memory, Java provides a character data type called char.
- The char type assumes a size of 2 bytes but, basically, it can hold only a single character.

Boolean Type:

- Boolean type is used when we want to test a particular condition during the execution of the program.
- Two values that a boolean type can take: **true or false**.



Data types in Java

Variables:

- A variable is an identifier that denotes a storage location used to store a data value.
- Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during the execution of the program .

Variable names may consist of alphabets, digits, the underscore (-) and dollar characters, subject to the following conditions:

1. They must not begin with a digit.
2. Uppercase and lowercase are distinct.
3. It should not be a keyword.
4. White space is not allowed.
5. Variable names can be of any length

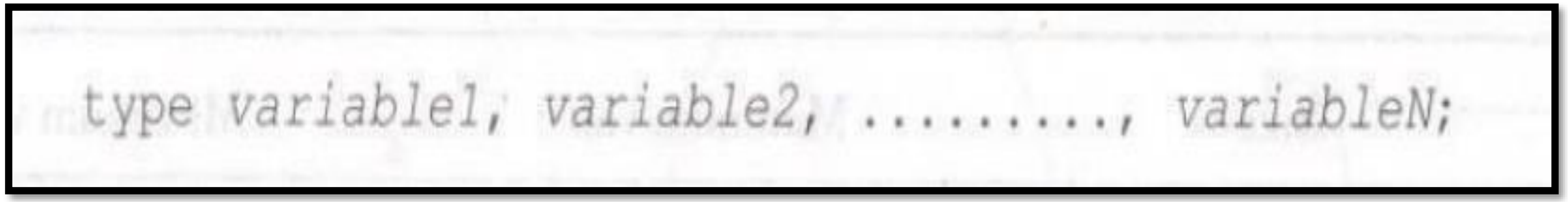
Declaration Of variables:

Declaration does three things:

1. It tells the compiler what the variable name is.
2. It specifies what type of data the variable will hold.
3. The place of declaration (in the program) decides the scope of the variable.

The general form of declaration of a variableis:

A variable must be declared before it is used in the program. A variable can be used to store a value of any data type.



```
type variable1, variable2, ........., variableN;
```

Variables are separated by commas, A declaration statement must end with a semicolon.

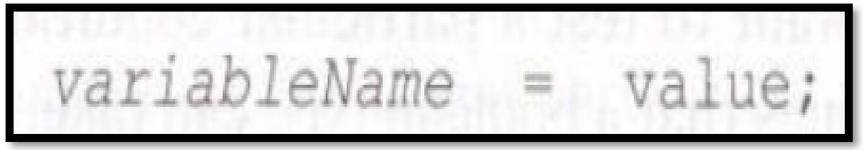
Some valid declarations are: int count; float x, y; double pi;

GIVING VALUES TO VARIABLES

This can be achieved in two ways:

1. By using an assignment statement
2. By using a read statement

Assignment Statement :



```
variableName  =  value;
```

//Giving value to the variable.

Example: i=0;

Read Statement :

- The **readLine()** method is used to read a line of text entered by the user.
- The entered line of text is then displayed.

SCOPE OF VARIABLES:

Java variables are classified into 3 Types:

1.instance variables

2.class variables

3.local variables

Instance and class variables are declared inside a class.

Instance variables:

- created when the objects are instantiated and therefore they are associated with the objects.
- They take different values for each object.

class variables :

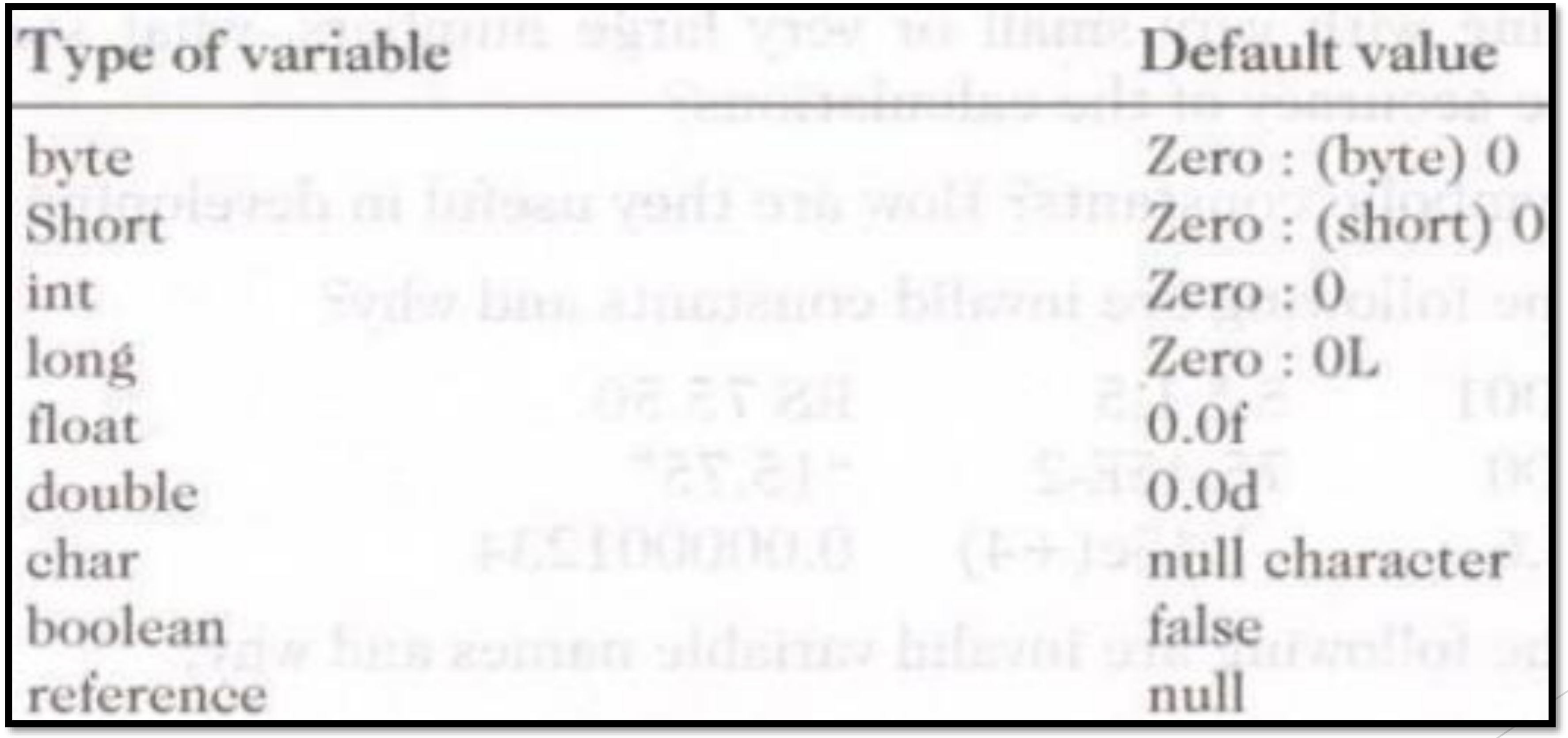
- Are global to a class and belong to the entire set of objects that class creates.
- Only one memory location is created for each class variable.

Local Variables:

- Are Variables declared and used inside methods .
- They are called so because they are not available for use outside the method definition.
- Local variables can be declared inside program blocks that are defined between an opening brace { and a closing brace }.

- These variables are visible to the program only from the beginning of its program block to the end of the program block.
- When the program control leaves a block, all the variables in the block will cease to exist.
- The area of the program where the variable is accessible is called its **scope.**

In Java, every variable has a default value.



Type of variable	Default value
byte	Zero : (byte) 0
Short	Zero : (short) 0
int	Zero : 0
long	Zero : 0L
float	0.0f
double	0.0d
char	null character
boolean	false
reference	null

SYMBOLIC CONSTANTS

- Symbolic constants are declared using the final keyword.
- We use unique constants in a program that may appear repeatedly in a number of places in the program.

Example:3.142, representing the value of the mathematical constant "pi".

Modifiability:

- We may like to change the value of "pi" from 3.142 to 3.14159 to improve the accuracy of calculations or the number 50 to 100 to process the test results of another class.
- In both the cases, we will have to search throughout the program and explicitly change the value of the constant wherever it has been used.
- If any value is left unchanged, the program may produce incorrect outputs.

Understandability :

- When a numeric value appears in a program, its use is not always clear, especially when the same value means different things in different places

Examples of constant declaration are: final int STRENGTH = 100; final int PASS MARK = 50;

Note :

1. Symbolic names take the same form as variable names.
2. After declaration of symbolic constants, they should not be assigned any other value within the program by using an assignment statement.

3. Symbolic constants are declared for types.

4. They can NOT be declared inside a method

TYPECASTING

Type casting in Java refers to the **conversion of one data type into another.**

Java has two types of casting:

1.**Implicit casting (also known as widening or automatic casting)**

- The process of assigning a **smaller type to a larger one** is known as widening

Example:

'int' to **'long', 'float'** to **'double'**

2.Explicit casting (also known as narrowing or manual casting).

- Assigning a **larger type to a smaller one** is known as narrowing.
- Narrowing may result in loss of information

Example

'double' to **'int', 'long'** to **'int'**

Automatic Conversion

- Automatic type conversion, also known as automatic casting, allows assigning a value of one data type to a variable of a different data type without the need for explicit casting.
- Automatic type conversion is possible only if the destination type has enough precision to store the source value.
- **For example,** int is large enough to hold a byte value. Therefore, byte b= 75; int a = b; are valid statements.

Changes are introduced during the final assignment.

1. float to int causes truncation of the fractional part.
2. double to float Causes rounding of digits.
3. long to int causes dropping of the excess higher order bits

GETTING VALUES OF VARIABLES :

A computer program is written to manipulate a given set of data and to display or print the results.

Java supports two output methods that can be used to send the results to the screen.

print ( ) method / / print and wait

println( ) method / / print a line and move to next line

print() method :

prints output on one line until a newline character is encountered.

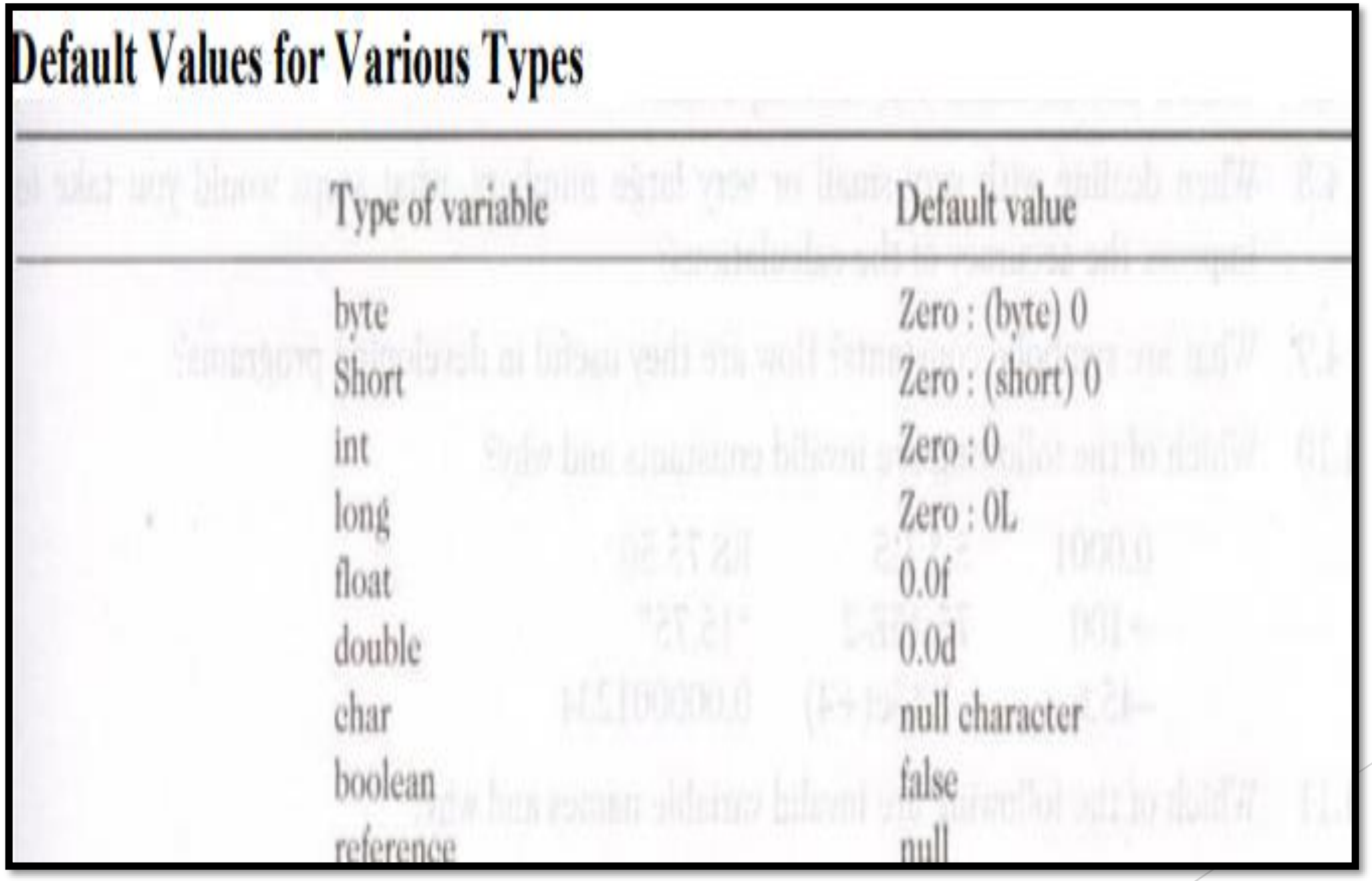
Example, the statements System.out.print("Hello ");

System.out.print("Java!");

It display the words Hello Java! on one line and waits for displaying further information on the same line.

STANDARD DEFAULT VALUES

In Java, every variable has a default value. If we don't initialize a variable when it is first created, Java provides default value to that variable type automatically.



Default Values for Various Types

Type of variable	Default value
byte	Zero : (byte) 0
Short	Zero : (short) 0
int	Zero : 0
long	Zero : 0L
float	0.0f
double	0.0d
char	null character
boolean	false
reference	null

Operators and Expressions :

Operator is a symbol that tells the computer to perform certain mathematical or logical manipulations.

Types Of Java Operators

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and decrement operators
6. Conditional operators.
7. Bitwise operators
8. Special operators

ARITHMETIC OPERATORS

- Java provides all the basic arithmetic operators. These can operate on any built-in numeric data type of Java.
- We cannot use these operators on boolean type

Arithmetic operators are used as shown below:

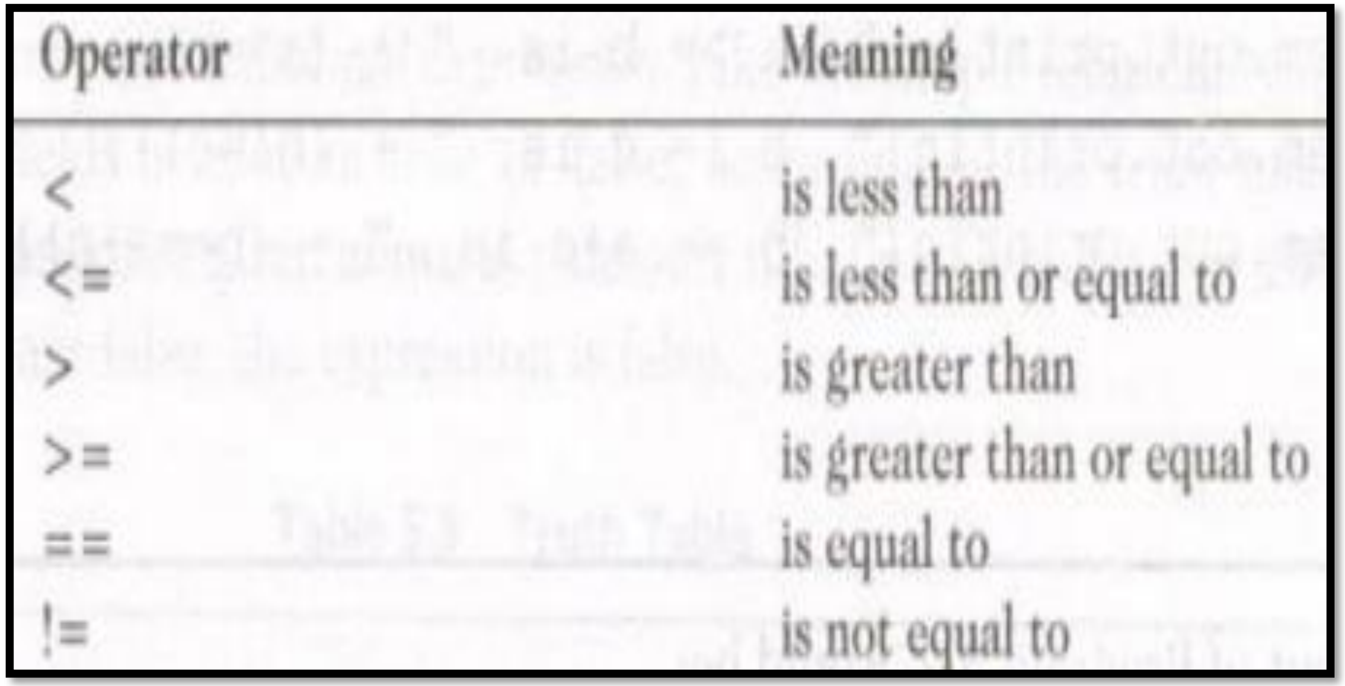
a-b   a*b   a%b   a+b   a/b   -a*b

Here a and b may be variables or constants and are known as operands.

RELATIONALOPERATORS

Example:

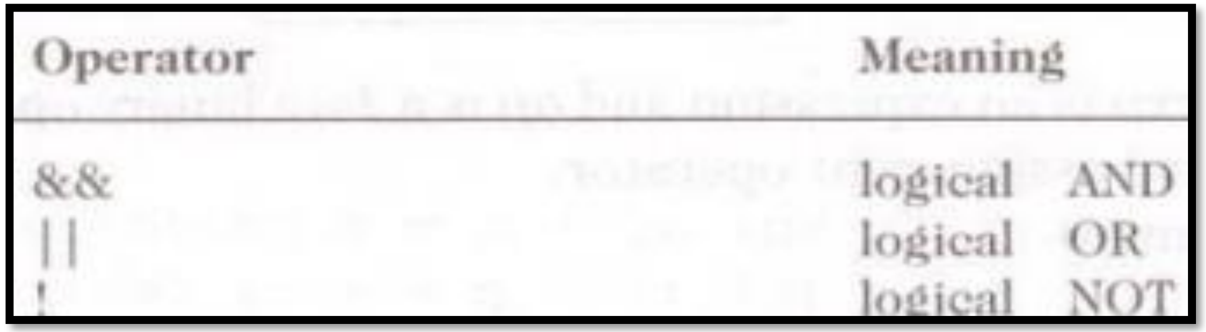
we may compare the age of two persons, or the price of two items, and so on. These comparisons can be done with the help of relational operators.



Operator	Meaning
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

LOGICAL OPERATORS

In addition to the relational operators, Java has three logical operators,



Operator	Meaning
&&	logical AND
||	logical OR
!	logical NOT

The logical operators && and | | are used when we want to form compound conditions by combining two or more relations.

Example: x > 3 && y < 10

ASSIGNMENT OPERATORS

Assignment operators are used to assign the value of an expression to a variable.

assignment operator **' = '.**

INCREMENT AND DECREMENTOPERATORS

Java has two very useful operators not generally found in many other languages. These are the increment and decrement operators:

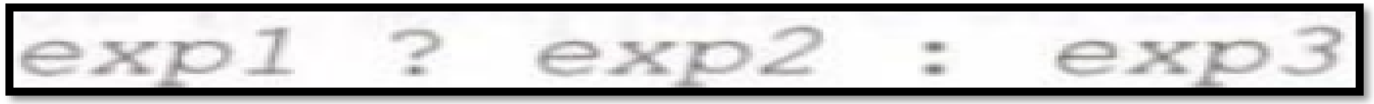
++ and—.

The operator + + adds 1 to the operand while - - subtracts 1.

CONDITIONAL OPERATOR

- The character pair ?: is a ternary operator available in Java.
- This operator is used to construct conditional expressions of the form

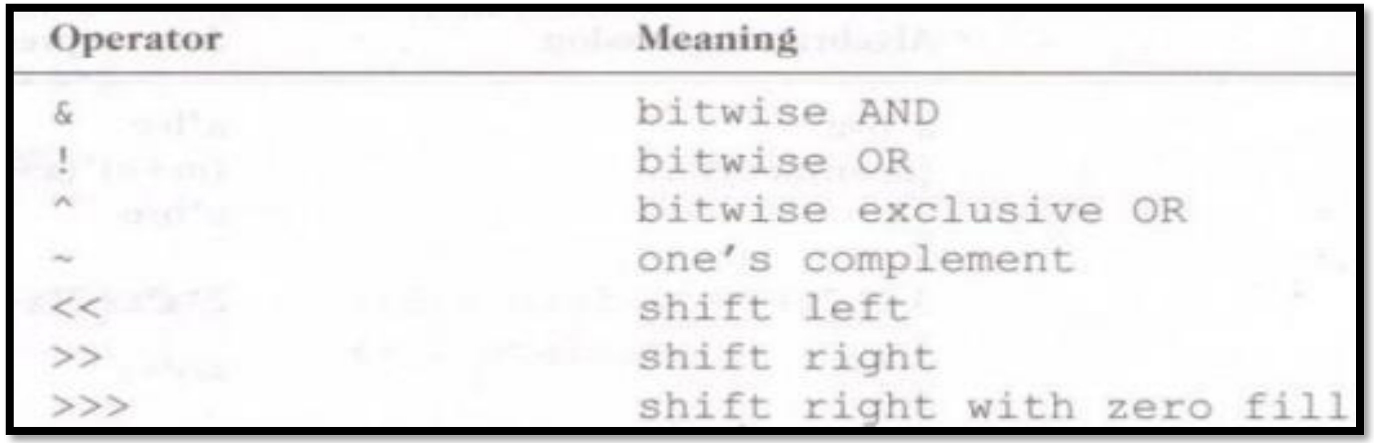
Where expl, exp2, and exp3 are expressions.



```
exp1 ? exp2 : exp3
```

BITWISE OPERATORS

Java has a distinction of supporting special operators known as bitwise operators for manipulation of data at values of bit level.



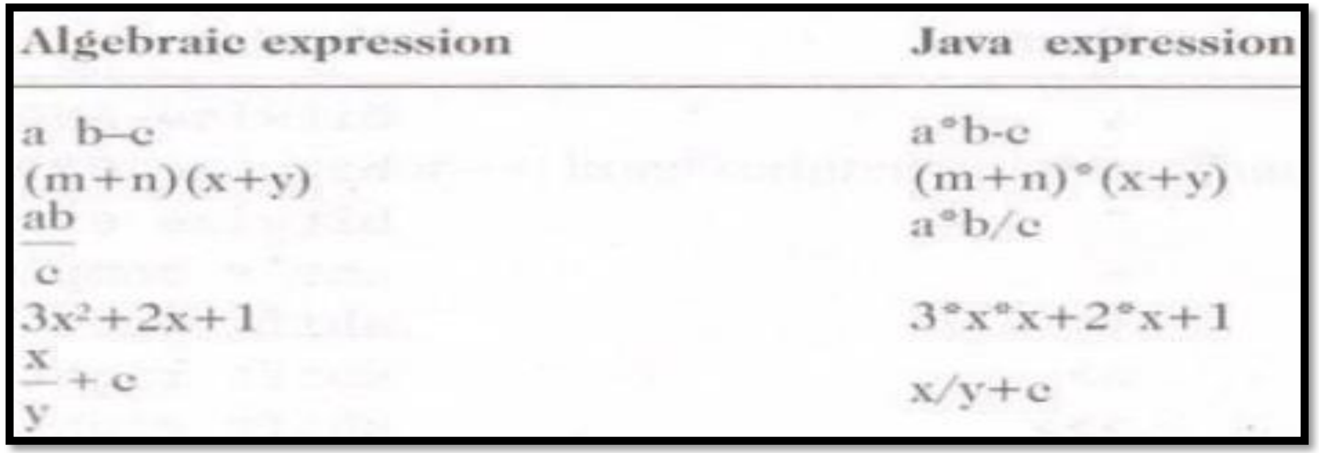
Operator	Meaning
&	bitwise AND
!	bitwise OR
^	bitwise exclusive OR
~	one's complement
<<	shift left
>>	shift right
>>>	shift right with zero fill

SPECIAL OPERATORS

Java supports some special operators of interest such as **instance of operator and member selection operator(.).**

ARITHMETIC EXPRESSIONS :

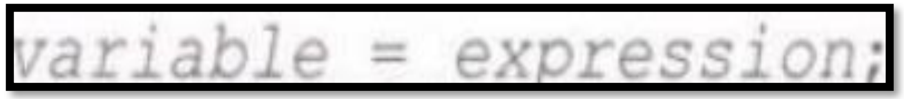
An arithmetic expression is a **combination of variables, constants, and operators** arranged as per the syntax of the language. Java can handle any complex mathematical expressions.



Algebraic expression	Java expression
a b–c	a*b-c
(m+n)(x+y)	(m+n)*(x+y)
$\frac{ab}{c}$	a*b/c
$3x^2+2x+1$	3*x*x+2*x+1
$\frac{x}{y}+c$	x/y+c

EVALUATION OF EXPRESSIONS

- Expressions are evaluated using an assignment statement of the form



```
variable = expression;
```

- Variable is any valid Java variable name. When the statement is encountered, the expression is evaluated first and the result then replaces the previous value of the variable on the left- hand side. All variables used in the expression must be assigned values before evaluation is attempted.
- **Examples of evaluation statements are :** x= a*b-c; y = b/c*a; z= a-b/c+d;

OPERATOR PRECEDENCE AND ASSOCIATIVITY:

- Each operator in Java has a level of importance known as **precedence.**
- Higher precedence means that an operator will be evaluated before operators with lower precedence.
- For example, multiplication (‘*’) has higher precedence than addition (‘+’) so multiplication is performed before addition.

Example: int result = 2 + 3 * 4;

Here, Multiplication has higher precedence, so ‘3*4’ is evaluated first, and then the result is added to ‘2’

Associativity:

- When operators have the same precedence, associativity determines the order of evaluation.
- Associativity can be left-to-right or right-to-left.
- For example, addition and subtraction (‘+’ and ‘-’) have the same precedence and left-to-right associativity.

Example:

int result = 10 - 5 + 3;

In this example, with left-to-right associativity, (‘10’-’5’) is evaluated first, and then the result is added to ‘3’.