Unit 03

OPERATORS AND PRECEDENCE

UNIT STRUCTURE

3.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Arithmetic Operator
- Assignment Operator
- Bitwise Operator
- Relation Operator
- Logical Operator
- Ternary Operator
- Operator Precedence

3.1 Introduction :

An Operator is a symbol that allows a us to perform arithmetic or logical operations on data. It operate on operands and cause changes in the operand value. Java provides a rich set of operators for manipulating programs. Commonly operator performs a function on one, two or three operands.

Unary operator perform operation based on one operand. The ++ or – – are the Unary operators. Binary operator perform operation based on two operands. The +, <, = are Binary operators. An operator that required three operands are called Ternary operator. The "expression1 ? expression2 : expression3" is an example of Ternary operator.

In Java, Operators are divided in to six categories. :

- Arithmetic Operators
- Assignment Operators
- Bitwise Operators
- Relational Operators
- Logical Operators
- Ternary Operator

3.2 Arithmetic Operator :

The arithmetic operators are used to perform arithmetical operations. For example, addition, subtraction, multiplication, division and modulo.

These arithmetic operators can be unary or binary type. If the operator is specified with two operands then it is called binary operator and if the operators are used with single operand then they are called unary operators. The given table gives a brief description of binary operators :

Here the values of Variables A = 10 and B = 20

Table 3.1 : Binary Operators (Arithmetic Operator)

Operator	Description	Example
+	Addition – Adds values of either side of the operator	A + B will give 30
–	Subtraction – Subtracts right hand operand from left hand operand	A – B will give –10
*	Multiplication – Multiplies values on either side of the operator	A * B will give 200
/	Division – Divides left hand operand by right hand operand	B / A will give 2
%	Modulus – Divides left hand operand by right hand operand and returns remainder	B % A will give 0

❑ **Check Your Progress – 1 :**

1. Write a note on Arithmetic operators.

....

....

Following program shows the use of Arithmetic Operator

```
class ArithmaticDemo
{
   public static void main(String args[])
   {
      int a = 40;
      int b = 20;
```

```
        //Addition (+) operator
        System.out.println("Addition of a and b = " + (a + b));

        //Subtraction (-) operator
        System.out.println("Subtraction of a and b = " + (a - b));
        //Multiplication (*) operator
        System.out.println("Multiplication of a and b = " + (a * b));

        //Division (/) operator
        System.out.println("Division of a and b = " + (a / b));

        //Modulo (%) operator
        System.out.println("Modulo of a and b = " + (a % b));

    }
}
```

OUTPUT :

```
Addition of a and b = 60
Subtraction of a and b = 20
Multiplication of a and b = 800
Division of a and b = 2
Modulo of a and b = 0
```

3.3 Increment / Decrement Operator :

The binary forms of ++ and – – are unary operator. It is also know as increment / decrement operator. Each of these operator has unary versions that perform the following operations :

Table 3.2 : Increment / Decrement Operators

Operator	Description	Example
+ +	Increment by one value	+ +a, b+ +
– –	Decrement by one value	– –a, b– –

The unary operator also known as increment and decrement operators. The Increment and decrement operators can be used in two ways, either before or after an operand. Depending on the placement of these operators, it can be called as prefix or postfix operation. In the prefix notations, the value of the operands is incremented or decremented before assigning it to another variable, while in the postfix notation, the value of the operands is incremented or decremented after assigning it to another variable.

There are two types of increment and decrement operators :

1. Pre–increment/Decrement
2. Post–increment/Decrement

1. **Pre–Increment/Decrement –** The pre–increment/decrement operator increases/decreases the value of a variable first and then evaluates the expression.

   For example,

   If a = 2, then b = ++a will first increase the value of a that is make it 3 and then assign it to variable a.

   So, the final value of b becomes 3.

   Similarly, if a = 2 then b = – –a will first decrease the value of variable a by 1 and then assign it to variable b.

2. **Post–Increment/Decrement –** The post–increment/decrement operator first assigns the value to the variable and then increases/decreases it.

   For example, If a = 2; And b = a++;

   Then, first the value of a is assigned to b and then it is incremented/decremented by 1.

   As in the above example, first value of a is assigned to b and then it gets increased. So, the value of b becomes 2 and a becomes 3.

   Similarly, if a = 2 and b = a– –, then value of b becomes 2 and a becomes 1.

   Following program shows the use of Unary Operator ++ :

```
class UnaryDemo
{
    public static void main(String args[])
    {
        int i, j, k;
        i = 10;
        j = i++;

        //prefix increment
        System.out.println("i = " + i);
        System.out.println("j = " + j);

        i = 10;
        j = 0;
        j = ++j;

        //postfix increment
        System.out.println("i = " + i);
        System.out.println("j = " + j);
    }
}
```

OUTPUT

```
i = 11
j = 11

i = 11
j = 11
```

❑ **Check Your Progress – 2 :**

1. Write a note on increment and decrement operators.

....

....

3.4 Assignment Operator :

Assignment operators are used to assign a value to operand (variable). Assignment Operator is a Binary operator. General form of operators is as follow :

varname = expr;

Where, varname is variable name and expr is an expression.

Table 3.3 : Assignment Operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assigne value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
– =	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C –= A is equivalent to C = C – A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2

>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator	C &= 2 is same as C = C & 2
^=	Bitwise exclusive OR and assignment operator	C ^= 2 is same as C = C ^ 2
|=	Bitwise inclusive OR and assignment operator	C |= 2 is same as C = C | 2

3.5 Bitwise Operator :

The bitwise operator is used when the manipulation is to be done bit by bit, i.e., in terms of 0's and 1's. These are faster in execution than arithmetic operators. The given table displays the bitwise operators along with their meanings :

Table 3.4 : Bitwise Operators and their meanings

Operators	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	One's complement
<<	Left Shift
>>	Right Shift
>>>	Right Shift with zero fill

❖ **Bitwise Not (~)**

The bitwise operators perform operations on integer value. In the above table, the ~ operator is unary operator and the rest of the operators are binary operators.

The ~ (one's complement) operator inverts the bits which make up the integer value, that is, 0 bits becomes 1 and 1 bits becomes 0.

For example, if ~3 is written, then it will give the value of ~4. The same can be calculated as follows :

The Numerical value 3 can be represented as binary integer value–

```
00000000     00000000     00000000     00000011
```

Inverting each bits would give

```
11111111     11111111     11111111     11111100
```

It is same as bit pattern for –4 as an int value.

The given table shows the operations performed at bit level :

Table 3.5 : Operations Performed at Bit Level

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101
^	Binary XOR operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give –60 which is 1100 0011
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111
>>>	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>> 2 will give 15 which is 0000 1111

Now, based on the above table and explanations, let us take an example to understand the use of these bitwise operators. Consider the expressions, 63 & 252, 63 | 252 and 63 ^ 252

❖ **Bitwise AND (&)**

First of all we will calculate 63 & 252. To do the same, first represent the 63 and 252 values in their bit patterns.

```
00000000    00000000    00000000    00111111    = 63
00000000    00000000    00000000    11111100    = 252
```

As you know that and returns a 1 bit if and only if the corresponding bit from each operand is 1, we calculate 63 and 252 to be 60 as follows :

```
00000000    00000000    00000000    00111111    = 63
00000000    00000000    00000000    11111100    = 252
00000000    00000000    00000000    00111100    = 60
```

❖ **Bitwise OR (|)**

Now we will calculate 63 | 252. To do the same, first represent the 63 and 252 values in their bit patterns.

```
00000000    00000000    00000000    00111111    = 63
00000000    00000000    00000000    11111100    = 252
```

As you know that for Bitwise OR, Operator returns a 0 bit if, and only if, the corresponding bit form of each operand is 0. Else it returns 1. We calculate 63 OR 252 to be 255 as follows :

00000000	00000000	00000000	00111111	= 63
00000000	00000000	00000000	11111100	= 252
00000000	00000000	00000000	11111111	= 255

Bitwise XOR (|)

Now we will calculate 63 ^ 252. To do the same, first represent the 63 and 252 values in their bit patterns.

00000000	00000000	00000000	00111111	= 63
00000000	00000000	00000000	11111100	= 252

As you know that for Bitwise XOR, Operator returns a 0 bit if, and only if, the corresponding bit form of each operand match. Else it returns 1. We calculate 63 XOR 252 to be 195 as follows :

00000000	00000000	00000000	00111111	= 63
00000000	00000000	00000000	11111100	= 252
00000000	00000000	00000000	11000011	= 195

❖ **Shift Operators**

The shift operators work on the bit level. When the left operand is an int, only the last 5 bit of the right operand is used to perform the shift. This is due to the fact that an int is a 32 bit value and can only be shifted 0 through 31 times. Similarly, when the left operand is a long value, only the last 6 bits of the right operand are used to perform the shift, as long values are 64 bit values, they can only be shifted 0 through 63 times.

The << operator (left shift) causes the bits of the left operand to be shifted to the left based on the value of the right operand. The shifted right bits will be filled with 0 values.

The >> (right shift) operator causes the bits to the left operand to be shifted to the right, based on the value of the right operand. The bits that fill in the shifted left bits have the value of the leftmost bit (before the shift operation). This operator is also called signed shift as it preserves the sign (positive or negative) of the operand.

The >>> operator is similar to >> (Right shift) operator, except that the bits that fill in the shifted left bits have the value of 0. It is also called an unsigned shift as it does not preserve the sign of the operand.

❑ **Check Your Progress – 3 :**

1. Write a note on Bitwise operators.

   ....

   ....

3.6 Relation Operator :

The relational operators are used to compare two quantities. It either returns true or false value only.

For example,

num1>num2

Here, the value of num1 is checked with num2, if num1 is greater than num2 then a true value is returned else false. The syntax for declaring the relational operators is given below :

expr1 <relational operator> expr2

In the above syntax, expr1 and expr2 are the arithmetic expressions which can be variables, constants or both. When the arithmetic expressions are used on either side of a relational operator, the arithmetic expression gets evaluated first.

The given table shows the list of relational operators with their meaning :

Table 3.6 : Relational Operators

Operator	Description	Example
= =	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A = = B) is not true.
|=	Check if the value of the operands are equal or not, if values are not equal then condition becomes true.	(A |= B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not ture.
<	Checks if the value of left oeprand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

3.7 Logical Operator :

Logical operators are used to combine more than one condition to perform logical operation. There are 3 logical operators, the given table shows the list of these operators

Table 3.7 : Logical Operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is false.
||	Called Logical OR operator. If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
!	Called Logical NOT operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

The && and || are used to form compound conditions. For example, num1>num2 && num1 >num3

In the above expression, first the value of num1 and num2 will be compared (to the left side of && operator) then the values of num1 and num3 gets compared (to the right of &&) and finally AND operation is performed on both the results.

Now let us see the functions of these logical operators, that is, AND, OR and NOT.

1. **AND operator (&&)** – The AND operator returns true value when both the operands are true. The truth table for AND operator is given below :

Table 3.8 : Logical AND Operators

operand1	operand2	operand1 && operand2
True	True	True
True	False	False
False	True	False
False	False	False

2. **OR Operator (||)** – The OR operator (||) produces true output when one of the input is true or when both the inputs are true. The truth table of OR (||) operator is given below :

Table 3.9 : Logical OR Operators

operand1	operand2	operand1 || operand2
True	True	True
True	False	True
False	True	True
False	False	False

3. **NOT Operator (!)** – The NOT (!) operator is used to negate the condition, that is, if the true value is specified as an input then it produces false output and if false value is given as an input then true output is produced.

The truth table for NOT (!) operator is given below :

Table 3.10 : Logical XOR Operators

operand1	**!operand**
True	False
False	True

4. **Assignment Operators** – The assignment operator is used to assign a value to a variable. The syntax of assignment operator is given below :

varname= expr;

Where, varname is variable name and expr is an expression.

❑ **Check Your Progress – 4 :**

1. Write a note on Bitwise operators.

....

....

2. Unary operator perform operation based on ________ operand.

(A) One (B) two (C) three (D) four

3. The ________ operators are used to perform arithmetical operations. .

(A) Bitwise (B) Arithmetic (C) Relational (D) Logical

4. ________ also known as increment / decrement operator.

(A) ++ & – – (B) + & – (C) += & –= (D) =+ & =–

5. The ________ operator is used when the manipulation is to be done bit by bit.

(A) Bitwise (B) Arithmetic (C) Relational (D) Logical

6. The ________ operator is also known as ternary operator.

(A) Bitwise (B) conditional (C) Relational (D) Logical

7. The ________ have states and behaviours.

(A) Class (B) Objects (C) Variable (D) Method

8. Assignment operators are used to assign a value to operand.

(A) True (B) False

9. Bitwise Not operator is denote by ~ sign.

(A) True (B) False

10. The << operator (left shift) causes the bits of the left operand to be shifted to the left based on the value of the right operand.

(A) True (B) False

11. Logical operators are used to combine more than one condition to perform logical operation.

(A) True (B) False

3.8 Ternary Operator :

The conditional operator is also known as ternary operator. It is denoted by ? and :. The syntax of same can be given as :

```
expr1 ? expr2  :expr3
```

In the above syntax, expr1, expr2 and expr3 are expressions and expr1 must be of boolean type.

For example,

```
If a = 10
   b = 2
   z = (a > b) ? a : b
```

In the above example, the value of a > b gets evaluated first, if the condition is true then value of a gets assigned to z otherwise the value of b.

```
class TarnaryDemo
{
   public static void main(String args[])
   {
      int a = 12;
      int b = 8;
      int c;

      c =  (a > b) ? a : b;

      System.out.println(c);
   }
}
```

OUTPUT

```
12
```

3.9 Operator Precedence :

The precedence of operators is useful when there are several operators in an expression. Java has specific rules for determining the order of evaluation of an expression. The given table displays the list of operators in the order of precedence. The hierarchy of Java operators with highest precedence is shown first.

a. All those expressions which are inside parenthesis are first evaluated, the nested parenthesis are evaluated from the innermost parenthesis to the outer.

b. All the operators which are in the same row have equal precedence.

c. The given table shows the list of operators with their order of evaluation–

Table 3.11 : Operators and their Evaluation

Operator	Type	Order of Evaluation
( ) [ ] .	Parenthesis Array Subscript Member Access	Left to right
++, – –	Prefix increment, decrement	Right to left
++, – – –	Postfix Increment, decrement Unary minus	Right to left
*,/,%	Multiplicative	Left to right
+, –	Additive	Left to right
<, >, <=, >=	Relational	Left to right
==,!=	Equality	Left to right
&&	AND	Left to right
||	OR	Left to right
? :	Conditional	Right to left
=, +=, –+, *=,/+,%=	Assignment	Right to left

❖ **Java Programs :**

When we consider a Java program it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods and instant variables mean.

- **Object** – Objects have states and behaviors. For example : A dog has states–color, name, and breed as well as behaviors –wagging, barking and eating. An object is an instance of a class.
- **Class** – A class can be defined as a template/ blue print that describe the behaviors/states that object of its type support.
- **Methods** – A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instant Variables** – Each object has its unique set of instant variables. An object's state is created by the values assigned to these instant variables.
- **Writing and Compiling Programs** – Let us look at a simple code that would print the word Welcome.

```
public class Welcome
   {
      /*This program will print Welcome */
      public static void main (String args [ ])
      {
         System.out.println ("Welcome"); //Print Welcome
      }
   }
```

Let us look at how to save the file, compile and run the program. Please follow the steps given below :

1. Open notepad and add the code as above.
2. Save the file as : Welcome.java.
3. Open a command prompt window and go to the directory where you saved the class. Assume its C:\.
4. Type ' javac Welcome.java ' and press enter to compile your code. If there are no syntax errors in your code the command prompt will take you to the next line (Assumption : The path variable is set).
5. Now type ' java Welcome ' to run your program.
6. You will be able to see Welcome' printed on the window. C:> javac Welcome.java

   C:>java Welcome Welcome

- **Basic Syntax –** About Java programs, it is very important to keep in mind the following points.
- **Case Sensitivity –** Java is case sensitive which means identifier Hi and hi would have different meaning in Java.
- **Class Names –** For all class names the first letter should be in Upper Case.

  If several words are used to form a name of the class, each inner word's first letter should be in upper case.

  Example : class Welcome
- **Method Names –** All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

  Example : public void myMethodName()
- **Program File Name –** Name of the program file should exactly match the class name.

When saving the file you should save it using the class name (Remember java is case sensitive) and append '.java' to the end of the name. (If the file name and the class name do not match your program will not compile).

Example : Assume 'Welcome' is the class name. Then the file should be saved as 'Welcome.java'

- **Public static void main(String args[ ]) –** Processing of java program starts from the main() method which is a mandatory part of every java program.

Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are :

1. Visible to the package, its default modifier.
2. Visible to the class only (private)
3. Visible to the world (public)
4. Visible to the package and all subclasses (protected).

Default Access Modifier – No keyword – Default access modifier means we do not explicitly declare an access modifier for a class, field, method etc.

A variable or method declared without any access control modifier is available to any class in the same package. The default modifier cannot be used for methods, fields in an interface.

Example :

Variables and methods can be declared without any modifiers, as in the following example :

```
String x= "123";
boolean processorder ( )
{
return true;
}
```

Private Access Modifier – private – Methods, Variables and Constructors that are declared private can only be accessed within the declared class itself.

Private access modifier is the most restrictive access level. Class and interfaces cannot be private.

Variables that are declared private can be accessed outside the class if public getter methods are present in the class.

Using the private access modifier is the main way that an object encapsulates itself and hides data from the outside world.

So to make the variable available to the outside world, we defined two public methods : get Format(), which returns the value of format and set Format(String), which sets its value.

Public Access Modifier – public – A class, method, constructor, interface etc declared public can be accessed from any other class. Therefore fields, methods, blocks declared inside a public class can be accessed from any class belonging to the Java Universe.

However, if the public class we are trying to access is in a different package then the public class still need to be imported.

Because of class inheritance, all public methods and variables of a class are inherited by its subclasses.

Example :

The following function uses public access control :

```
Public static void main (String args [ ])
{
//………..
}
```

The main ( ) method of an application has to be public. Otherwise, it could not be called by a Java interpreter (such as java) to run the class.

Protected Access Modifier – protected – Variables, methods and constructors which are declared protected in a superclass can be accessed only by the subclasses in other package or any class within the package of the protected members' class.

The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared protected; however, methods and fields in an interface cannot be declared protected.

Protected access gives the subclass a chance to use the helper method or variable, while preventing a nonrelated class from trying to use it.

3.10 Let Us Sum Up :

An operator is used to perform specific operation on two or more operands. The operators are classified as given are Arithmetic Operators, Relational Operators, Logical Operators, Assignment Operators, Increment and Decrement Operators, Conditional Operators, Bitwise Operators, Special Operators

The precedence of operators is useful when there are several operators in an expression. Java has specific rules for determining the order of evaluation of an expression. The given table displays the list of operators in the order of precedence. The hierarchy of Java operators with highest precedence is shown first as all those expressions which are inside parenthesis are first evaluated, the nested parenthesis are evaluated from the innermost parenthesis to the outer. Secondly all the operators which are in the same row have equal precedence.

Let us talk about our understanding related to Java program it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods and instant variables mean.

- **Object –** Objects have states and behaviors. For example : A dog has states–color, name, breed as well as behaviors –wagging, barking and eating. An object is an instance of a class.
- **Class –** A class can be defined as a template/ blue print that describe the behaviors/states that object of its type support.
- **Methods –** A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instant Variables –** Each object has its unique set of instant variables. An object's state is created by the values assigned to these instant variables

3.11 Suggested Answer for Check Your Progress :

❑ **Check Your Progress 1 :**

See Section 3.2

❑ **Check Your Progress 2 :**

See Section 3.3

❑ **Check Your Progress 3 :**

See Section 3.5

❑ **Check Your Progress 4 :**

1 : See Section 3.7		**2 :** A	**3 :** B	**4 :** A
5 : A	**6 :** B	**7 :** B	**8 :** A	**9 :** A
10 : A	**11 :** A			

3.12 Glossary :

1. **Operators** – java provides six kind of operators. Arithmetic's operatos uses for the mathematic operation. It required two operands to perform operation. Increment and Decrement are the unary operators. It will either increase or decrease operand value by one. Assignment operator is responsible to assign value of right hand operand to left hand side operand. Bitwise operators work on bit. It basically use for binary operation. Relation operator is uses for the compression between two operands. Logical operator's uses to take decision based on operands values. Ternary operators required three operands and uses for the decision making.
2. **Object** – Objects have states and behaviors. For example : A dog has states–color, name, breed as well as behaviors –wagging, barking and eating. An object is an instance of a class.
3. **Class** – A class can be defined as a template/ blue print that describe the behaviors/states that object of its type support.
4. **Methods** – A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

3.13 Assignment :

Write a program to explain the use of operators.

3.14 Activities :

1. Explain the concept of Pre-Increment/Decrement and Post Increment/ Decrement.
2. Explain class, object, methods and instant variables

3.15 Case Study :

For our case study, we will be creating two classes. They are Student and Student Details.

First open notepad and add the following code. Remember this is the Student class and the class is a public class. Now, save this source file with the name Student.java.

The Student class has four instance variables name, age, course and semester.

The class has one parameterised constructor, which takes parameters.

The class should also comprise with one public method display(), which displays the details of the student.

Processing starts from the main method. Therefore in–order for us to run this Student class there should be main method and objects should be created. We will be creating a separate class for these tasks.

Student Details class, which creates two instances of the class Student and invokes the methods for each object to assign values for each variable.

3.16 Further Readings :

1. Core Java 2, 2 volumes, Cay S. Horstmann, Gary Cornell, The Sun Microsystems Press, 1999, Indian reprint 2000
2. Java 2, The Complete Reference, Patrick Naughton and Herbert Schildt, Tata McGraw Hill, 1999
3. Programming with Java, Ed. 2, E. Balagurusamy, Tata McGraw Hill, 1998, reprint, 2000
4. The Java Tutorial, Ed. 2, 2 volumes, Mary Campione and Kathy Walrath, Addison Wesley Longmans, 1998
5. The Java Language Specification, Ed. 2, James Gosling, Bill Joy, Guy Steele & Gilad Bracha, (Ed. 1 1996, Ed. 2 2000), Sun Microsystems, 2000
6. Using Java 2, Joseph L. Weber, Prentice Hall, Eastern Economy Edition, 2000