UNIT 2: PROGRAMMING CONCEPTS OF BASIC JAVA

Unit Structure

2.0 Learning Objectives

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

- Describe Tokens, data types in Java
- Declare a variable, Java coding conventions
- Define typecasting
- Explain constants, operators
- Discuss precedence

2.1 Introduction

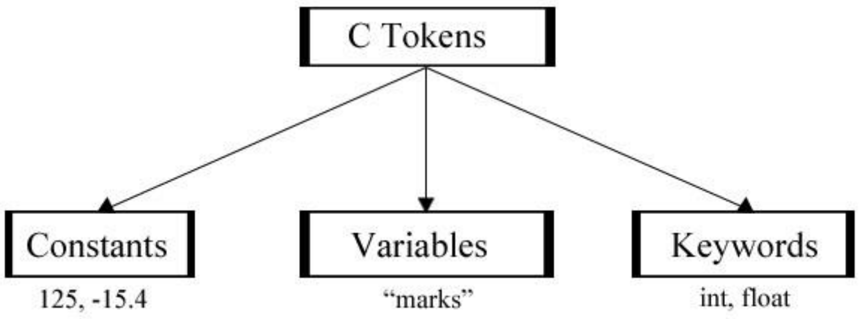
Java is an object-oriented programming language as it works totally on the concepts of Object oriented Programming concepts. Object oriented programming (called OOP for short) is the concept, which got its way after the procedural programming languages which was developed in 1970's.

The programming language that existed before was more of process oriented and this gave the concept of small entities called objects which could be made and reused as and when the need arises.

OOP hails from the idea of objects. All the real life entities are basically nothing but objects. In any program, every code will have an object having few characteristics features called the properties of that particular object or entity and then it will always have few actions to be performed over those entities or objects termed as Methods or Functions. To work with OOP, you should be able to identify three key characteristics of objects:

- The behavior of object: what can you do with this object, or what methods can you apply to it?
- The state of the object: how does the object react when you apply those methods?
- The identity of the object: how is the object distinguished from others that may have the same behavior and state?

2.2 Tokens



Tokens

The tokens of a language are the basic building blocks which can be put together to construct programs. A token can be a reserved word (such as int or while), an identifier (such as b or sum), a constant (such as 25 or "Alice in Wonderland"), a delimiter (such as {or **;**) or an operator (such as + or =). These tokens are explained below:

Identifiers - Identifiers are those words, which help to identify an entity. It can be used for class names, method names and variable names. It can be represented using uppercase and lowercase characters, numbers, underscore and dollar sign characters.

For Example,

a. Class _Name

b. B5

c. $name

d. This_is_program

Literals/Constants - Literals or constants are those quantities whose value may not change during execution of program.

For Example,

a. “This is Program”

b. ‘a’

c. 100.52

d. 56

In Java, the keyword final is used to denote a constant. The value of final variable cannot change after it has been initialised.

For example,

final int x=0;

The above statement declares a final variable and initialises it, all at once. While attempting to assign a value to variable x would result in compilation error.

Keywords - Keywords are also called reserved words. These are those words whose meaning is already explained to the compiler, so that when they are used in the program the compiler never generates error. There are mainly 49 keywords used in Java language. These keywords are given in the following table:

Table 2.1: Keywords

Abstract	continue	goto	package	synchronised
Assert	default	if	private	this
Boolean	do	implements	protected	throw
Break	double	import	public	throws
Byte	else	Instance of	return	transient
Case	extends	int	short	try
Catch	final	interface	static	void
Char	finally	long	strictfp	volatile
Class	float	native	super	while
Const	for	new	switch	

In addition to the above keywords, true, false and null are also the reserved words whose values are defined by Java.

Check your progress 1

1. Write a note on identifiers.
2. Explain constants

....

....

....

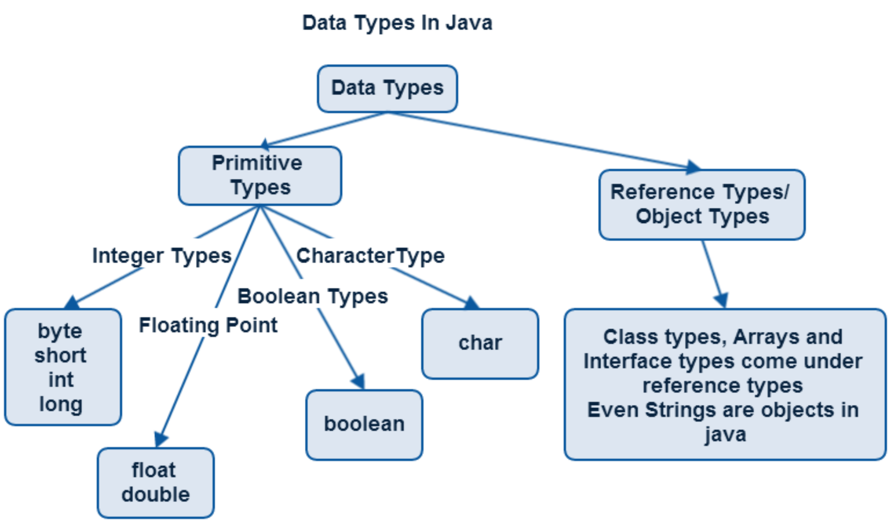
....

....

....

....

2.3 Data Types in Java



Data Types in Java

- Java is a strongly typed language. Every variable, expression has a type and these types are strictly checked. Unlike C, type checking is strictly enforced at run time.
- Impossible to typecast incompatible types.

In Java, a floating point value can be assigned to an integer.

Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the memory. Therefore, by assigning different data types to variables, you can store int, float, char, boolean (any of the eight primitive data types in these variables.

There are two data types available in Java:

1. Primitive Data Types
2. Reference/Object Data Types

Primitive Data Types:

There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named by a key word. Let us now look into detail about the eight primitive data types.

Byte:

- Byte data type is an 8-bit signed.
- Minimum value is -128
- Maximum value is 127
- Default value is 0
- Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an int.
- Example : byte a = 100 , byte b = -50

Short:

- Short data type is a 16-bit signed.
- Minimum value is -32,768
- Maximum value is 32,767
- Short data type can also be used to save memory as byte data type.
- Default value is 0.
- Example : short s= 10000 , short r = -20000

int:

- int data type is a 32-bit signed.
- Minimum value is 2,147,483,648.
- Maximum value is 2,147,483,647.
- int is generally used as the default data type for integral values unless there is a concern about memory.
- The default value is 0.
- Example : int a = 200, int b = -400

Long:

- Long data type is a 64-bit signed.
- Minimum value is -9,223,372,036,854,775,808.
- Maximum value is 9,223,372,036,854,775,807.
- This type is used when a wider range than int is needed.
- Default value is 0L.
- Example : int a = 200L, int b = -400L

Float:

- Float data type is a 32 bit floating point numbers.
- Float is mainly used to save memory in large arrays of floating point numbers.
- Default value is 0.0f.
- Float data type is never used for precise values such as currency.
- Example: float f1 = 134.5f

Double:

- Double data type is a double-precision 64-bit floating point.
- This data type is generally used as the default data type for decimal values.

- Default value is 0.0d.
- Example: double d1 = 113.4

Boolean:

- boolean data type represents one bit of information.
- There are only two possible values: true and false.
- This data type is used for simple flags that track true/false conditions.
- Default value is false.
- Example: boolean b = false

Char:

- Char data type is a single 16-bit Unicode character.
- Minimum value is '\u0000' (or 0).
- Maximum value is '\uffff' (or 65,535 inclusive).
- Char data type is used to store any character.
- Example: char letter ='C'

Reference Data Types:

- Hold the reference of dynamically created objects which are in the heap memory
- Can hold three types of values:
- Class type: Points to an object / instance of a class
- Interface type: Points to an object, which is implementing the corresponding interface
- Array type: Points to an array object or “null”
- Difference between Primitive & Reference data types:
- Primitive data types hold values themselves
- Reference data types hold reference to objects, i.e. they are not objects, but reference or pointers to objects

- Reference variables are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed. For example, employee, puppy etc.
- Class objects and various types of array variables come under reference data type.
- Default value of any reference variable is null.
- A reference variable can be used to refer to any object of the declared type or any compatible type.

Example: Animal animal = new Animal("elephant").

Check your progress 2

1. Explain float data type.
2. Write a note on reference data type.

....

....

....

....

....

....

....

....

....

....

....

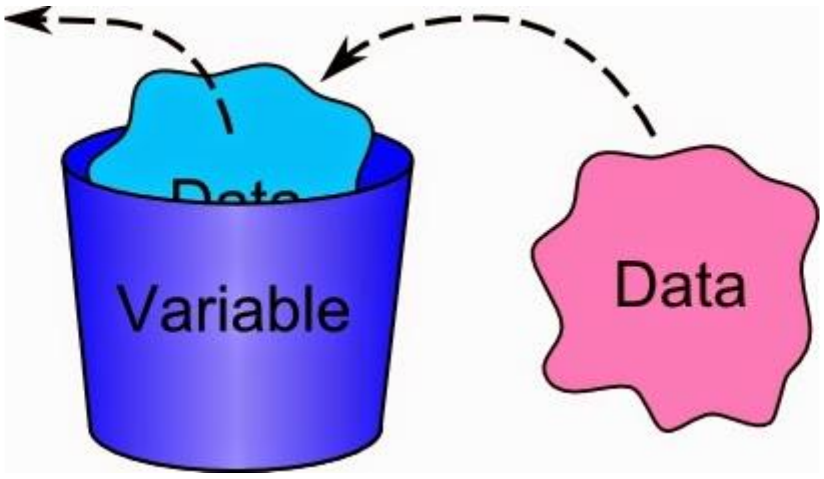
....

....

....

....

2.4 Declaring a Variable



Declaring a Variable

In Java, variables are those quantities whose value may change during execution of program. These variables should be initialised before they are used. The syntax of variable declaration is given below:

Syntax:

Type identifier [=value] [, identifier [=value]…];

Where,

Type is one of Java's atomic types, i.e., the name of class or interfaces.

For example, the given statements display the method of variable declaration.

```
int a, b, c;        //will declare three variables a, b and c of integer types
int x=5, y=10, z=25;   //will declare 3 variables x, y and z with 5, 10 and 25 values.
double pi=3.14;        //will declare an approximate value of pi
char x= 'a';           //will declare variable x with character value 'a'
```

Check your progress 3

1. What is a variable?
2. Write the syntax of declaring variables.

....

....

....

....

....

....

....

....

....

....

2.5 Java Coding Conventions

Reducing the cost of software maintenance is the significant reason for following coding conventions. In their introduction to code conventions for the Java Programming Language, Sun Microsystems provides the rationale.

Code conventions are important to programmers for a various reasons:

- 80% of the lifetime cost of a piece of software goes to maintenance.
- Hardly any software is maintained for its whole life by the same programmer.
- Code conventions improve the readability of the software, allowing software engineers to understand new code more quickly and thoroughly.
- If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product.
- Coding conventions allows simple scripts or programs whose job is to process source code for some purpose other than compiling it into an

executable. It is common practice to count the software size (Source lines of code) to track current project progress or establish a baseline for future project estimates.

- Consistent coding standards can, in turn, make the measurements more consistent. Special tags within source code comments are often used to process documentation.

Check your progress 4

1. Why code conventions are vital to programmers?
2. Explain the use of special tags in source code comments.

....

....

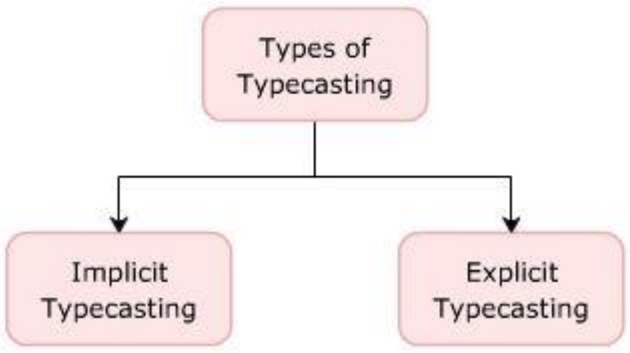
....

....

....

....

2.6 Typecasting



Typecasting

The conversion of one data type to another data type is called typecasting or type conversion.

There are mainly two types of conversions; these are implicit or automatic and explicit conversions.

a. Automatic Conversion

The type conversions are automatically performed when the type of the expression on the right hand side of an assignment operation can be safely promoted to the type of the variable on the left hand side of the assignment.

Thus, the values can be safely assigned as:

byte->short->int->long->float->double

The extra storage associated with long integer as shown in the above example will be padded with extra zeroes.

b. Explicit Conversion

If we want to assign the value of long to an integer then an integer variable will require more storage and may result in loss of data. To force such conversion, an explicit conversion is performed and the process is called explicit typecasting.

For example,

If

```
int x;

long y;
```

Then,

```
x= (int) y;
```

The above statement tells the compiler that the type of variable y must be temporarily changed to an int when the given assignment statement is processed. Thus, the cast only lasts for the duration of the assignment.

So, the syntax of explicit typecasting can be given as:

(T) N

Here, T is the name of a numeric type and N is a data item of another numeric type. The result is of type T

Check your progress 5

1. What do you mean by the term typecasting?
2. Explain explicit conversion with suitable example.

....

....

....

....

....

....

....

....

....

....

....

....

....

2.7 Constants

Java is a programming language used to create programs that can run on a variety of Operating Systems. A Java constant is a variable with a pre-defined value. Although Java does not have a constant type, you can effectively obtain the same effect with a final variable. This enables you to have a control of what is constant and what is not.

You can have a constant effect by declaring and initializing public, static and final variables. The static modifier makes the variable obtainable without loading an occurrence of the class where it is defined.

Once you have 34 initialized the constant variables, their value cannot be changed anymore. After initializing, you can gain access to the constant value with the variable's name and the name of its class with a period.

Standard Naming Convention

In declaring Java constant variables, you should declare the variable names in ALL CAPS (notice each letter of the variable name above). The words in Java constants are typically separated with underscores (as in the example above). This format indicates that these values are constants. It will be easier for an individual to read a code if this standard naming convention is followed.

Check your progress 6

1. How can a constant effect be achieved?
2. Illustrate an example of Java constant.

....

....

....

....

....

....

....

....

....

....

2.8 Operators

An operator is used to perform specific operation on two or more operands. The operators are classified as given below:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators

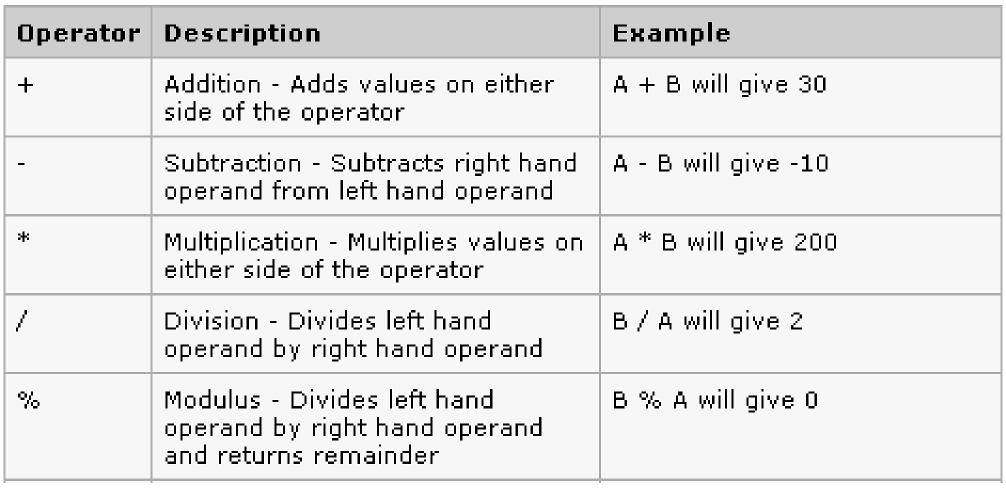
5. Increment and Decrement Operators
6. Conditional Operators
7. Bitwise Operators
8. Special Operators

Let us discuss these operators in detail:

a. **Arithmetic Operators -** 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:

Table 2.2: Binary Operators



Operator	Description	Example
+	Addition - Adds values on 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

b. **Relational Operators -** The relational operators are used to compare two quantities. It either returns true or false value only.

For example,

num1>num2

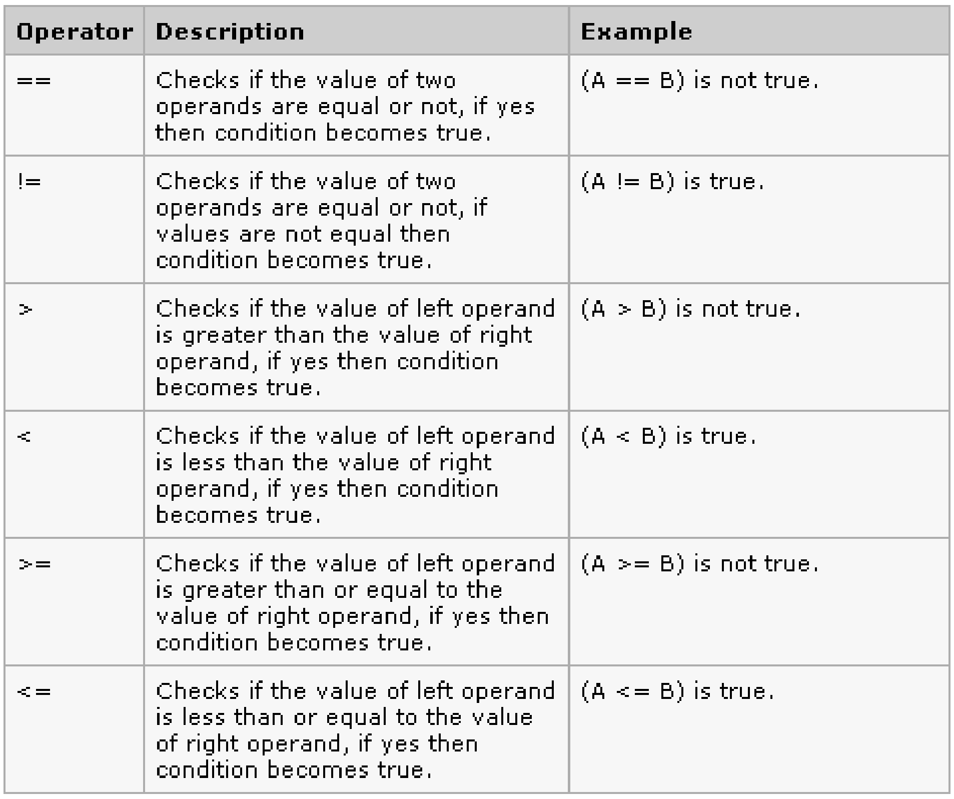
Here, the value of num1 is checked with num2, if num1 is greater than num2 then a true value is returned else false. The syntax for declaring the relational operators is given below:

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:

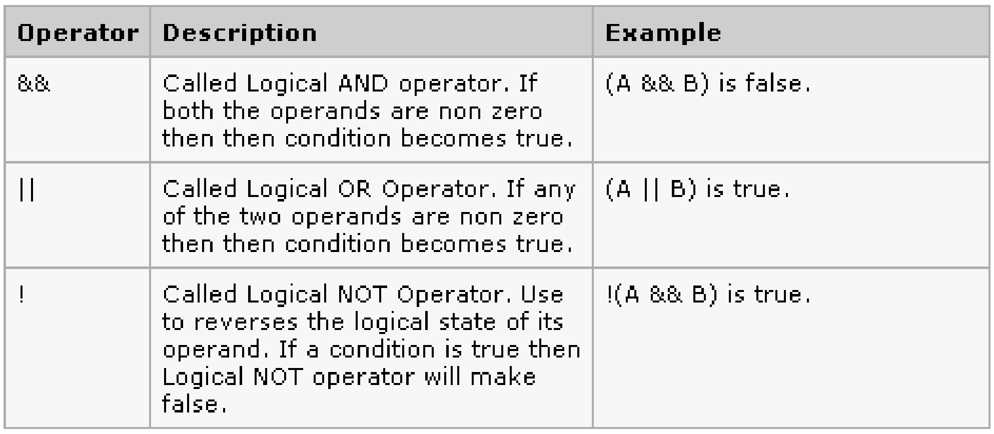
Table 2.3: Relational operators



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.
!=	Checks if the value of two 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 true.
<	Checks if the value of left operand 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.

c. **Logical Operators -** 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 2.4: 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:

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:

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:

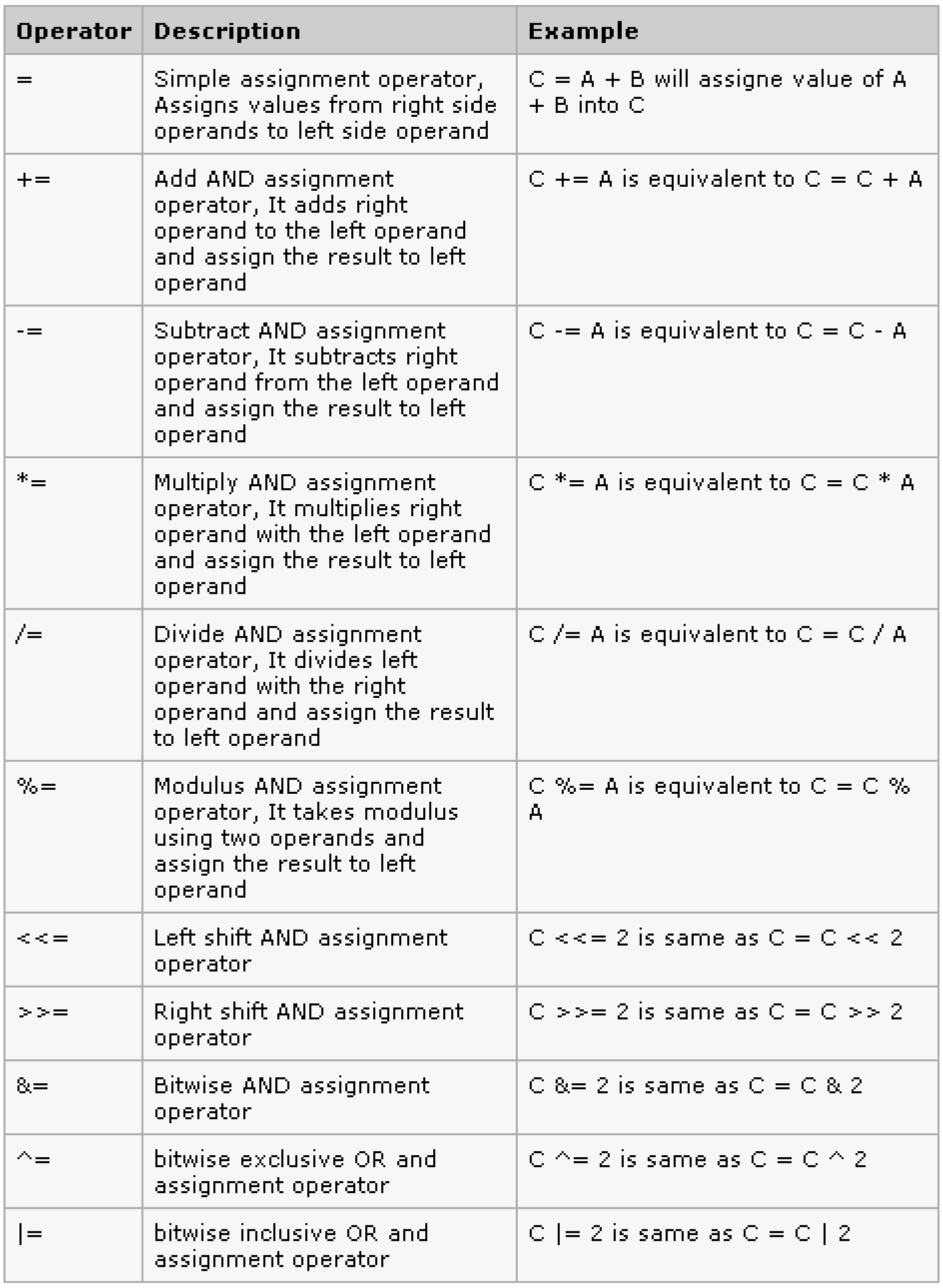
operand1	**!operand**
True	False
False	True

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.

Table2.5: 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 to 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

5. **Increment and Decrement Operators -** The increment and decrement operators are used to increase and decrease the value of variable by 1. The operator ‘++’ adds 1 to the operand value while the operator ‘—‘ subtracts 1. Both the ++ and – operators are unary operators.

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.

6. **Conditional Operators (Ternary Operators) -** 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.

7. **Bitwise and Shift Operators**

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 2.6: 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

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:

3 can be represented as binary integer value-

00000000 00000000 00000000 00000011

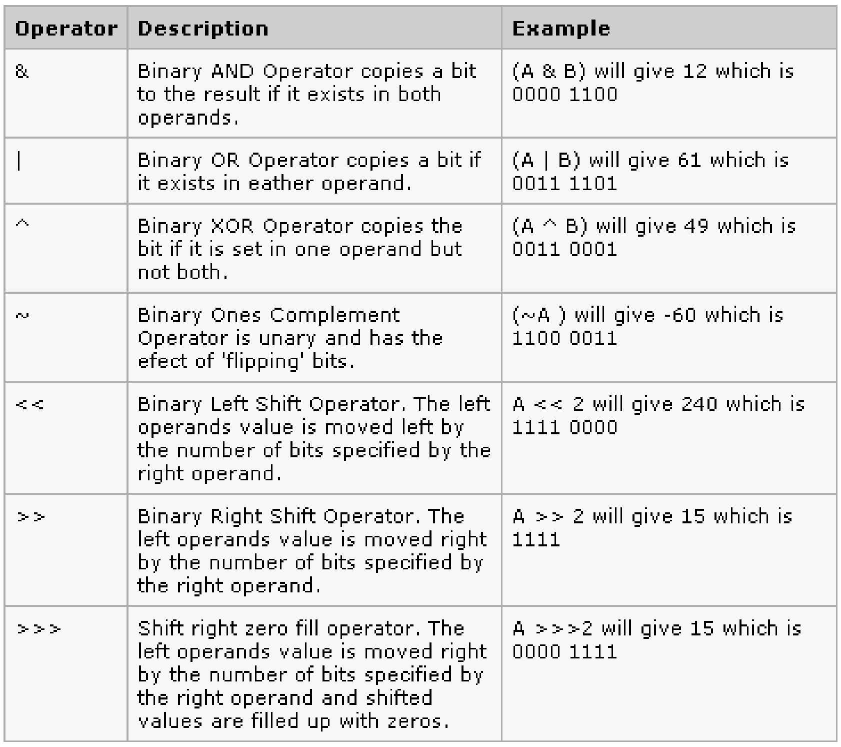
Inverting each bits would give

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 2.7: 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 eather 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 efect 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

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
```

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 8

1. Write a note on increment and decrement operators.

   ....

   ....

   ....

   ....

   ....

   ....

2.9 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 2.8: 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.

Check your progress 9

1. Write the hierarchy of Java operators with highest precedence.

   ....

   ....

   ....

   ....

   ....

   ....

   ....

   ....

   ....

2.1 Let Us Sum Up

This unit deals with several important basic aspects of Java Programming. One of them is the tokens of a language which are the basic building blocks and can be put together to construct programs. A token can be a reserved word (such as int or while), an identifier (such as b or sum), a constant (such as 25 or "Alice in Wonderland"), a delimiter (such as { or **;**) or an operator (such as + or =). These tokens are explained below:

Identifiers - Identifiers are those words, which help to identify an entity. It can be used for class names, method names and variable names. It can be represented using uppercase and lowercase characters, numbers, underscore and dollar sign characters as a) Java is a strongly typed language. Every variable, expression has a type and these types are strictly checked. Unlike C, type checking is strictly enforced at run time. 2) Impossible to typecast incompatible types. In Java, a floating point value can be assigned to an integer. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the memory. Therefore, by assigning different data types to variables, you can store int, float, char, boolean (any of the eight primitive data types in these variables. There are two data types available in Java: 1. Primitive Data Types 2.Reference/Object Data Types

In Java, variables are those quantities whose value may change during execution of program. These variables should be initialized before they are used. The syntax of variable declaration is given below:

Syntax - Type identifier [=value] [, identifier [=value]…];

Reducing the cost of software maintenance is the significant reason for following coding conventions. In their introduction to code conventions for the Java Programming Language, Sun Microsystems provides the rationale. Code conventions are important to programmers for a various reasons. Further it is understood that the conversion of one data type to another data type is called typecasting or type conversion. There are mainly two types of conversions; these are implicit or automatic and explicit conversions.

Java is a programming language used to create programs that can run on a variety of Operating Systems. A Java constant is a variable with a pre-defined value. Although Java does not have a constant type, you can effectively obtain the same effect with a final variable. This enables you to have a control of what is constant and what is not.

You can have a constant effect by declaring and initializing public, static and final variables. The static modifier makes the variable obtainable without loading an occurrence of the class where it is defined. Once you have to initialize the constant variables; their value cannot be changed anymore. After initializing, you can gain access to the constant value with the variable's name and the name of its class with a period. We learned that there is Standard Naming Convention. In declaring Java constant variables, you should declare the variable names in ALL CAPS (notice each letter of the variable name above). The words in Java constants are typically separated with underscores (as in the example above). This format indicates that these values are constants. It will be easier for an individual to read a code if this standard naming convention is followed.

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.

2.11 Suggested Answer for Check Your Progress

Check your progress 1

Answers: See Section 2.2

Check your progress 2

Answers: See Section 2.3

Check your progress 3

Answers: See Section 2.4

Check your progress 4

Answers: See Section 2.5

Check your progress 5

Answers: See Section 2.6

Check your progress 6

Answers: See Section 2.7

Check your progress 7

Answers: See Section 2.8

Check your progress 8

Answers: See Section 2.9

2.12 Glossary

1. **Tokens -** The tokens of a language are the basic building blocks which can be put together to construct programs.
2. **Identifiers -** Identifiers are those words, which help to identify an entity. It can be used for class names, method names and variable names.
3. **Literals/Constants -** Literals or constants are those quantities whose value may not change during execution of program.
4. **Typecasting -** conversion of one data type to another data type is called typecasting or type conversion.
5. **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.
6. **Class -** A class can be defined as a template/ blue print that describe the behaviors/states that object of its type support.
7. **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.

2.13 Assignment

Write a program to explain the use of operators.

2.14 Activities

1. Write the steps to save a file, compile and run program.
2. Explain class, object, methods and instant variables

2.15 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.

1.16 Further Reading

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