Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

SAIRAM
DIGITAL RESOURCES

YEAR II

SEM III

CS8392

UNIT NO 1

INTRODUCTION TO OOP AND JAVA FUNDAMENTALS

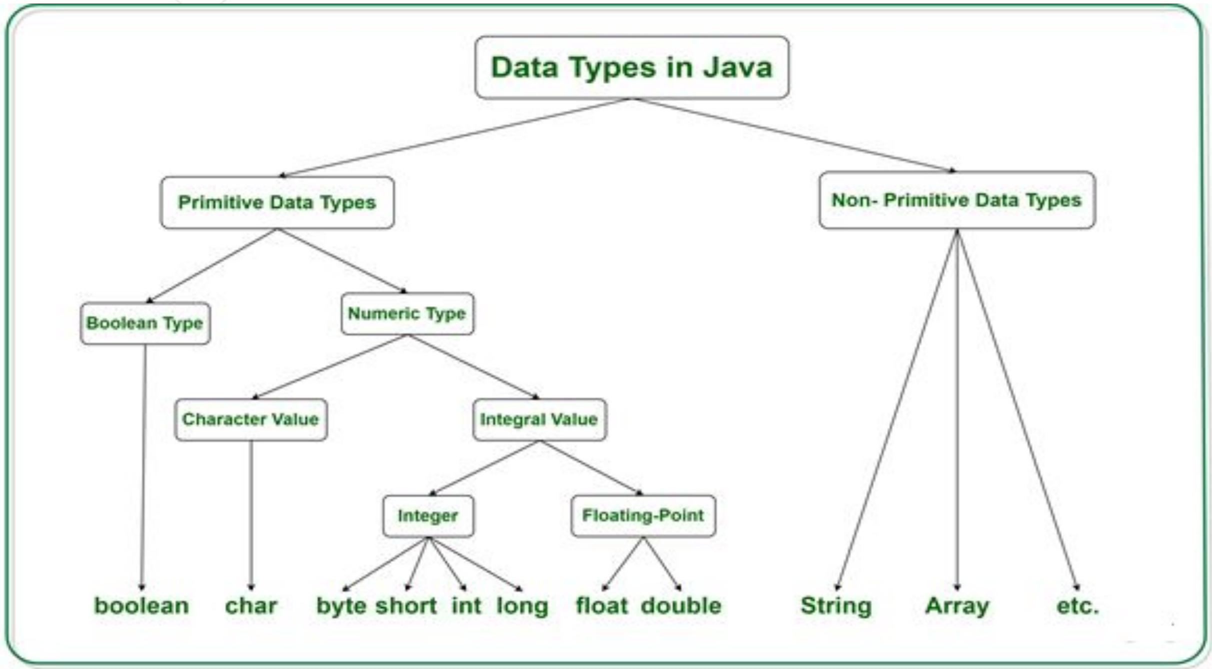
1.7 Data types, Variables, Operators

OBJECT ORIENTED PROGRAMMING
(Common to CSE, EEE, EIE, ICE, IT)

COMPUTER SCIENCE & ENGINEERING

Java Data Types

- Data types specify the different sizes and values that can be stored in the variable.
- Java is a statically-typed programming language .i.e all variables must be declared before its use.
- Two categories of data type:
  - Primitive Data Types
  - Non Primitive Data Types



Primitive Data Type

- Primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.
- Java has eight primitive data types:
  1. boolean,
  2. byte,
  3. char,
  4. short,
  5. int,
  6. long,
  7. float and
  8. double.
- These data types are included to maintain the portability of java as the size of these primitive data types do not change from one operating system to another.

1, boolean

- The Boolean data type is used to store only two possible values: true and false.
- Used for simple flags that track true/false conditions.
- Specifies one bit of information, but its "size" can't be defined precisely.

Syntax:

```
boolean booleanVar;
```

Example:

```
boolean bool=false;
```

2. byte

- The byte data type is an 8-bit signed two's complement integer.
- Useful for saving memory in large arrays.

Syntax:

```
byte var;
```

Example:

```
byte var = 126;
```

5. int

- The int data type is a 32-bit signed two's complement integer.
- Generally used as a default data type for integral values unless if there is no problem about memory.

Syntax:

```
int intVar;
```

Example:

```
int intVar = 900;
```

6. long

- The long data type is a 64-bit two's complement integer..
- Used when there is need a range of values more than those provided by int.

Syntax:

```
long longVar;
```

Example:

```
long longVar = -200000L;
```

7. float

- The float data type is a single-precision 32-bit IEEE 754 floating point.
- Generally used as the default data type for decimal values
- Used when there is a need to save memory in large arrays of floating point numbers.

Syntax:

```
float floatVar;
```

Example:

```
float floatVar=98.4f;
```

8. double

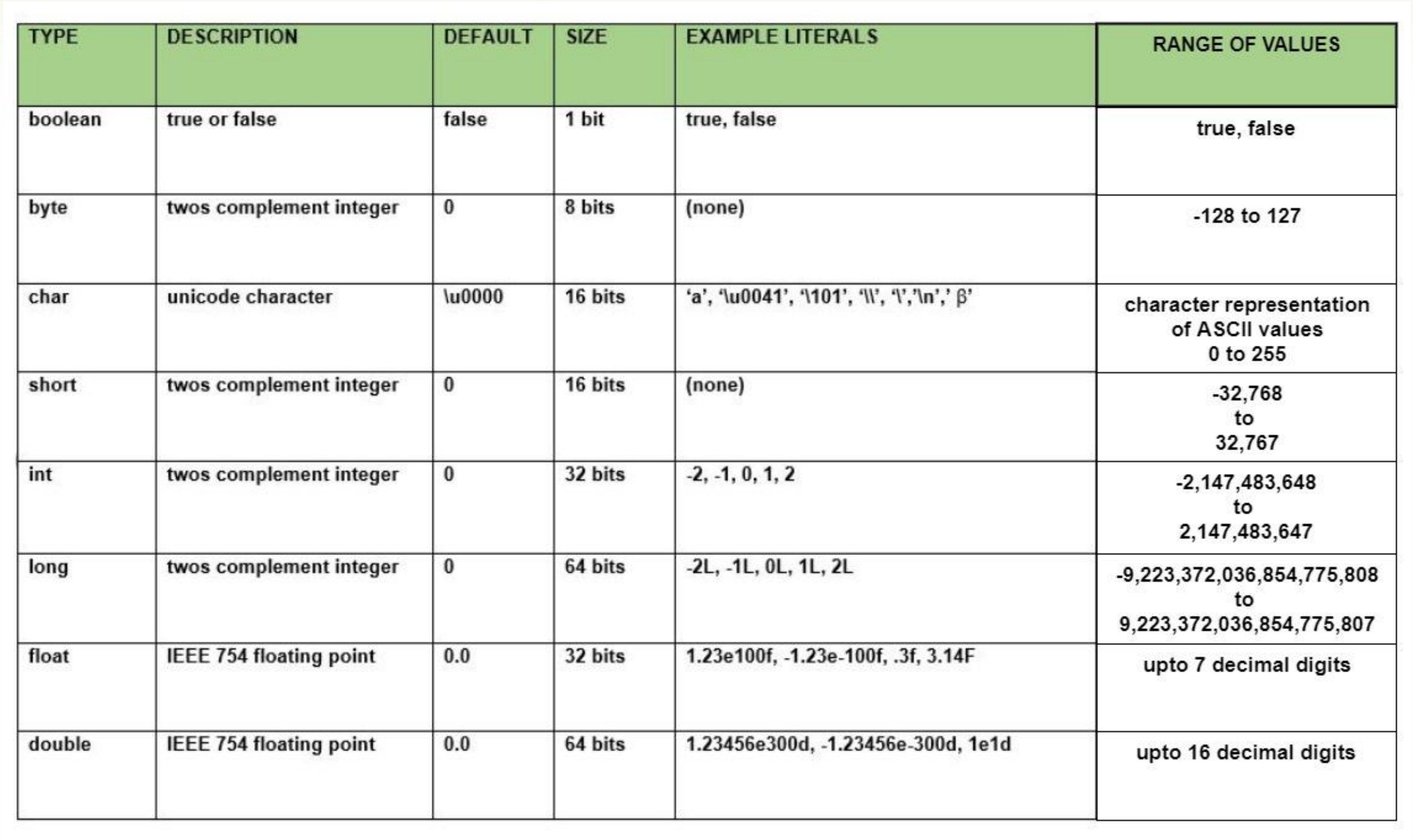
- The double data type is a double-precision 64-bit IEEE 754 floating point.
- Generally used as the default data type for decimal values

Syntax:

```
double doubleVar;
```

Example:

```
double doubleVar =196.5;
```



TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS	RANGE OF VALUES
boolean	true or false	false	1 bit	true, false	true, false
byte	twos complement integer	0	8 bits	(none)	-128 to 127
char	unicode character	\u0000	16 bits	‘a’, ‘\u0041’, ‘\101’, ‘\\’, ‘\’,’\n’,’ β’	character representation of ASCII values 0 to 255
short	twos complement integer	0	16 bits	(none)	-32,768 to 32,767
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2	-2,147,483,648 to 2,147,483,647
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F	upto 7 decimal digits
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d	upto 16 decimal digits

VARIABLES

VARIABLES

 A variable is a container which holds the value and that can be changed during the execution of the program.

A variable is assigned with a data type.

Variable is a name of memory location.

 All the variables must be declared before they can be used.

 There are three types of variables in java: **local variable, instance variable and static variable**

The basic form of a variable declaration is

datatype variable [= value][, variable [= value] ...] ;

Here data type is one of Java's data types and variable is the name of the variable. To declare more than one variable of the specified type, use a comma-separated list.

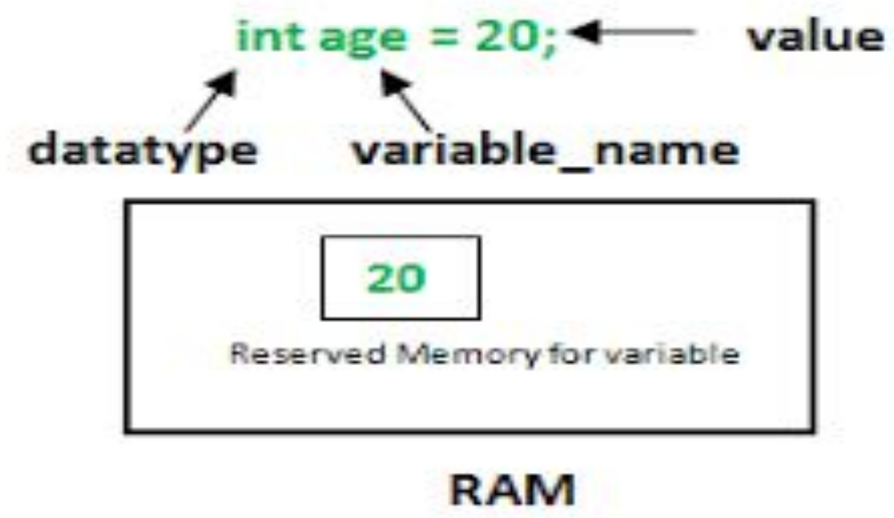
Example

int a, b, c; // Declaration of variables a, b, and c.

int a = 20, b = 30; // initialization

byte B = 22; // Declaration initializes a byte type variable B.

VARIABLES


int age = 20;
value
datatype
variable_name
20
Reserved Memory for variable
RAM

Types of Variable

There are three types of variables in java:

- **Local variable**
  Local Variables are a variable that are declared inside the body of a method.

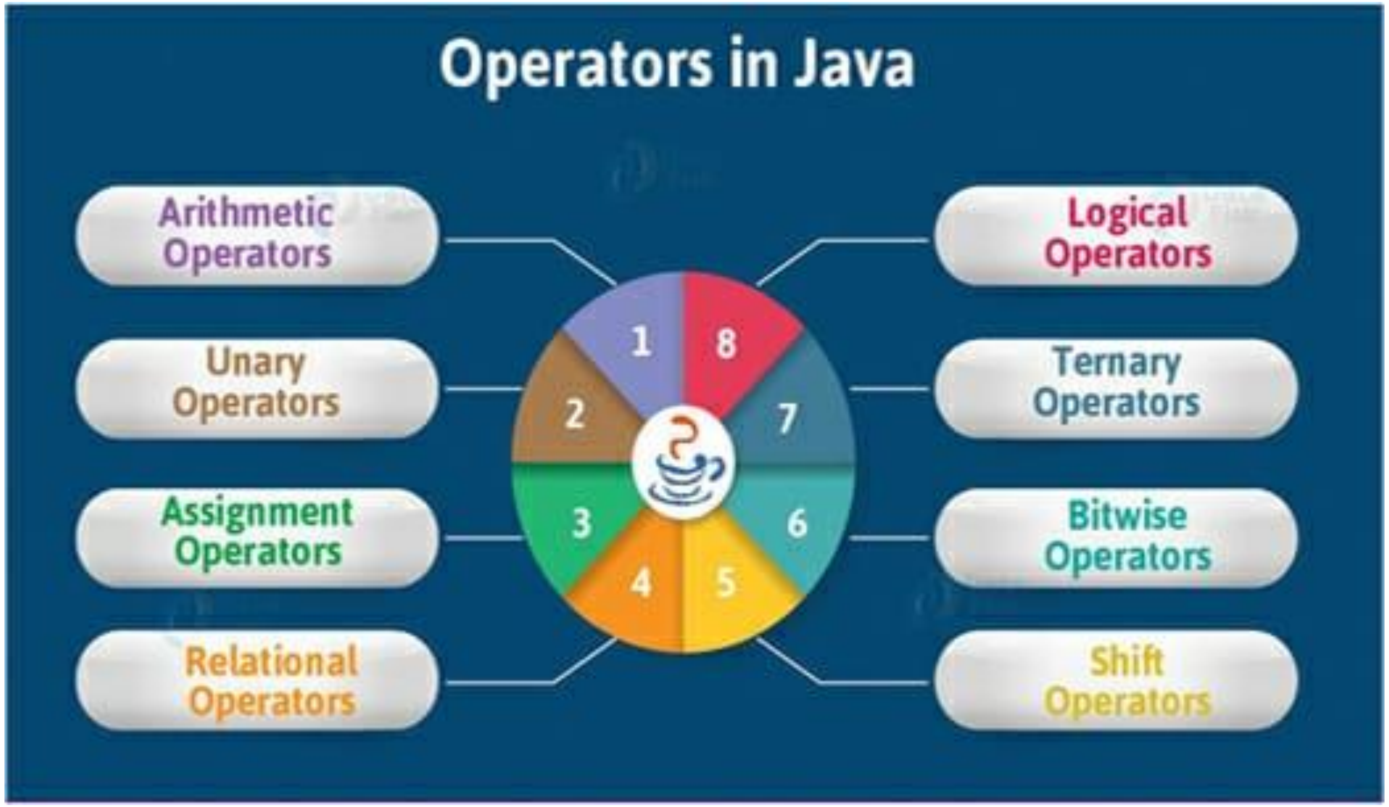
- **Instance variable**
  Instance variables are defined without the STATIC keyword .They are defined Outside a method declaration. They are Object specific and are known as instance variables.

- **Static variable**
  Static variables are initialized only once, at the start of the program execution. These variables should be initialized first, before the initialization of any instance variables.

```
class exvariable
 {
     int data = 99; //instance variable
     static int a = 1; //static variable
         void method()
         {
         int b = 90; //local variable
          }
}
```

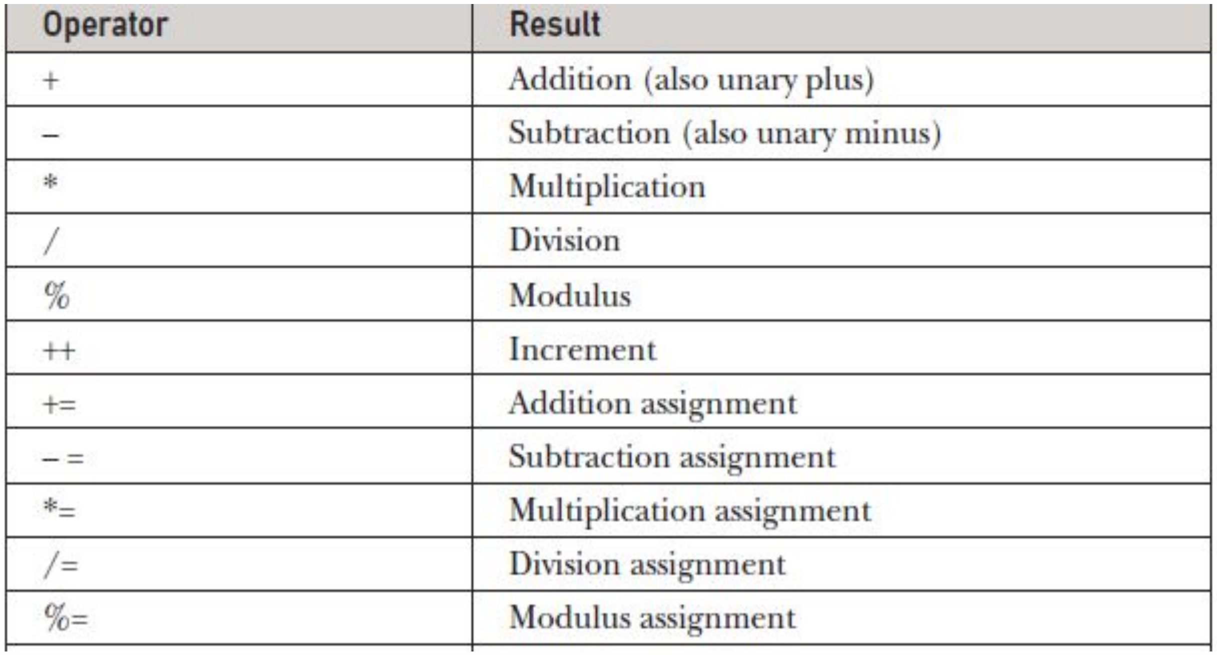
OPERATORS

Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups



ARITHMETIC OPERATORS

Arithmetic operators are used to manipulate mathematical expressions



Operator	Result
+	Addition (also unary plus)
–	Subtraction (also unary minus)
*	Multiplication
/	Division
%	Modulus
++	Increment
+=	Addition assignment
– =	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment

ARITHMETIC OPERATORS EXAMPLE

Arithmetic Operator Example

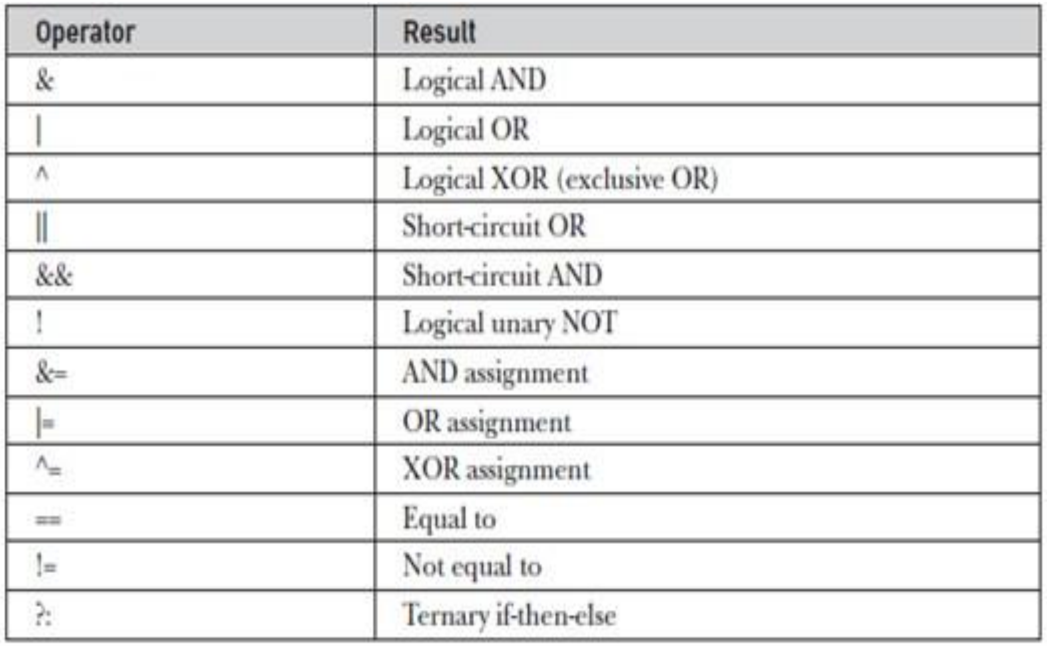
```
class OperatorExample{

 public static void main(String args[]) {
        int a=10;
        int b=5;
        System.out.println(a+b);//15
        System.out.println(a-b);//5
        System.out.println(a*b);//50
        System.out.println(a/b);//2
        System.out.println(a%b);//0 9.
        }
 }
```

Output:

```
15
5
50
2
0
```

BITWISE OPERATORS

Bitwise Operators



Operator	Result
&	Logical AND
|	Logical OR
^	Logical XOR (exclusive OR)
||	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to
?:	Ternary if-then-else

BITWISE OPERATORS EXAMPLE

Bitwise Operators Example

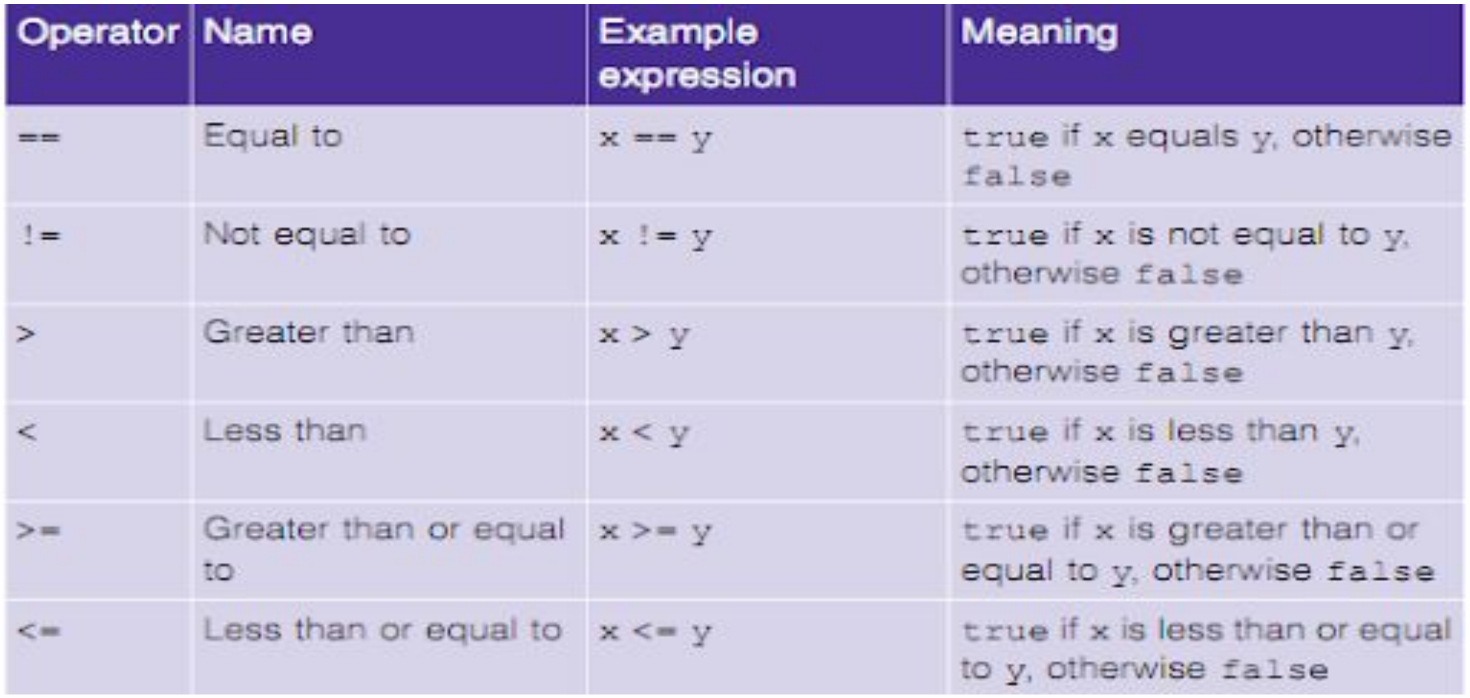
```
public class operators {
    public static void main(String[] args)
    {
        int a = 5;
        int b = 7;
        System.out.println("a&b = " + (a & b));
        System.out.println("a|b = " + (a | b));
        System.out.println("a^b = " + (a ^ b));
        System.out.println("~a = " + ~a);
        a &= b;
        System.out.println("a= " + a);
    }
}
```

Output :

```
a&b = 5
a|b = 7
a^b = 2
~a = -6
a= 5
```

RELATIONAL OPERATORS

The Relational Operators



Operator	Name	Example expression	Meaning
`==`	Equal to	`x == y`	`true` if x equals y, otherwise `false`
`!=`	Not equal to	`x != y`	`true` if x is not equal to y, otherwise `false`
`>`	Greater than	`x > y`	`true` if x is greater than y, otherwise `false`
`<`	Less than	`x < y`	`true` if x is less than y, otherwise `false`
`>=`	Greater than or equal to	`x >= y`	`true` if x is greater than or equal to y, otherwise `false`
`<=`	Less than or equal to	`x <= y`	`true` if x is less than or equal to y, otherwise `false`

RELATIONAL OPERATORS

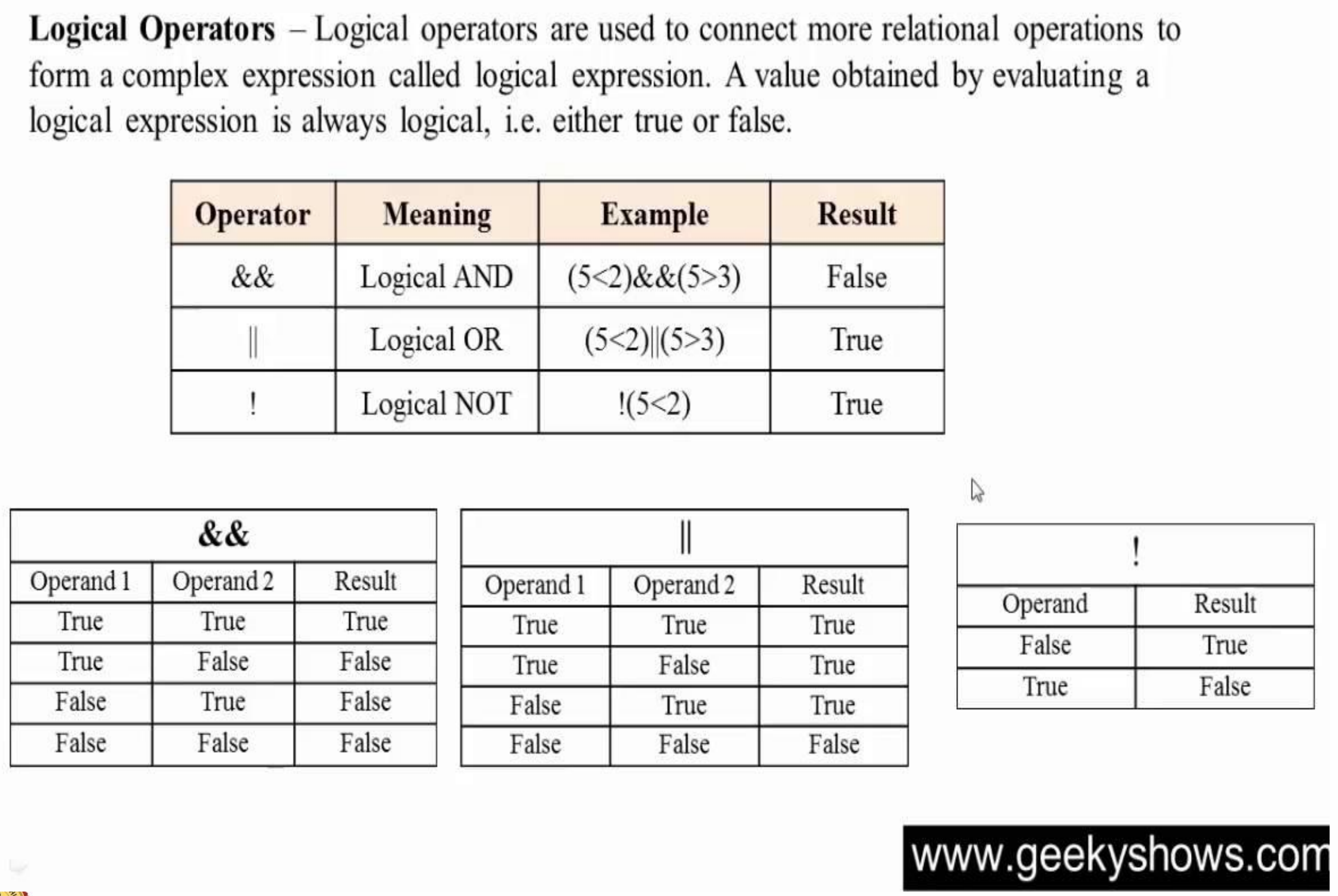
The Relational Operators Example

```
public class Test {
   public static void main(String args[]) {
      int a = 10;
      int b = 20;
      System.out.println("a == b = " + (a == b) );
      System.out.println("a != b = " + (a != b) );
      System.out.println("a > b = " + (a > b) );
      System.out.println("a < b = " + (a < b) );
      System.out.println("b >= a = " + (b >= a) );
      System.out.println("b <= a = " + (b <= a) );
   }
}
```

Output:

```
a == b = false
a != b = true
a > b = false
a < b = true
b >= a = true
b <= a = false
```

LOGICAL OPERATORS



Logical Operators – Logical operators are used to connect more relational operations to form a complex expression called logical expression. A value obtained by evaluating a logical expression is always logical, i.e. either true or false.

Operator	Meaning	Example	Result
&&	Logical AND	(5<2)&&(5>3)	False
||	Logical OR	(5<2)||(5>3)	True
!	Logical NOT	!(5<2)	True

&&		
Operand 1	Operand 2	Result
True	True	True
True	False	False
False	True	False
False	False	False

||		
Operand 1	Operand 2	Result
True	True	True
True	False	True
False	True	True
False	False	False

!	
Operand	Result
False	True
True	False

LOGICAL OPERATORS

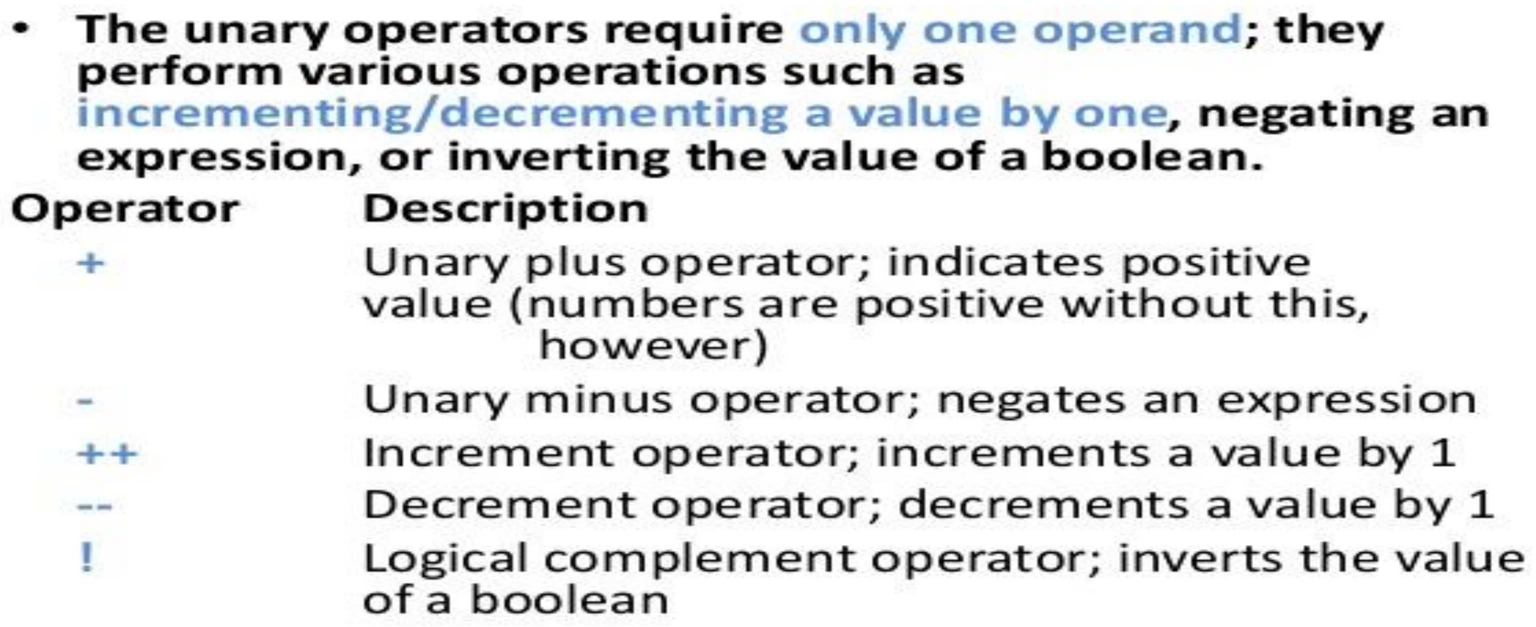
Logical Operators Example

```
class OperatorExample  {
public static void main(String args[])  {
        int a=10;
        int b=5;
        int c=20;
        System.out.println(a>b||a<c);//true || true = true
        System.out.println(a>b|a<c);//true | true = true
        System.out.println(a>b||a++<c);//true || true = true
        System.out.println(a);//10 because second condition is not checked
        System.out.println(a>b|a++<c);//true | true = true
        System.out.println(a);//11 because second condition is checked
         }
}
```

Output:

```
true
true
true
10
true
11
```

UNARY OPERATORS



- **The unary operators require only one operand; they perform various operations such as incrementing/decrementing a value by one, negating an expression, or inverting the value of a boolean.**

Operator	Description
+	Unary plus operator; indicates positive value (numbers are positive without this, however)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean

UNARY OPERATORS

Java Unary Operator Example: ++ and --

```
class OperatorExample{
        public static void main(String args[]) {
                int x=10;
                System.out.println(x++);//10 (11)
                System.out.println(++x);//12
                System.out.println(x--);//12 (11)
                System.out.println(--x);//10
            }
        }
```

Output:

```
10
12
12
10
```

OPERATORS

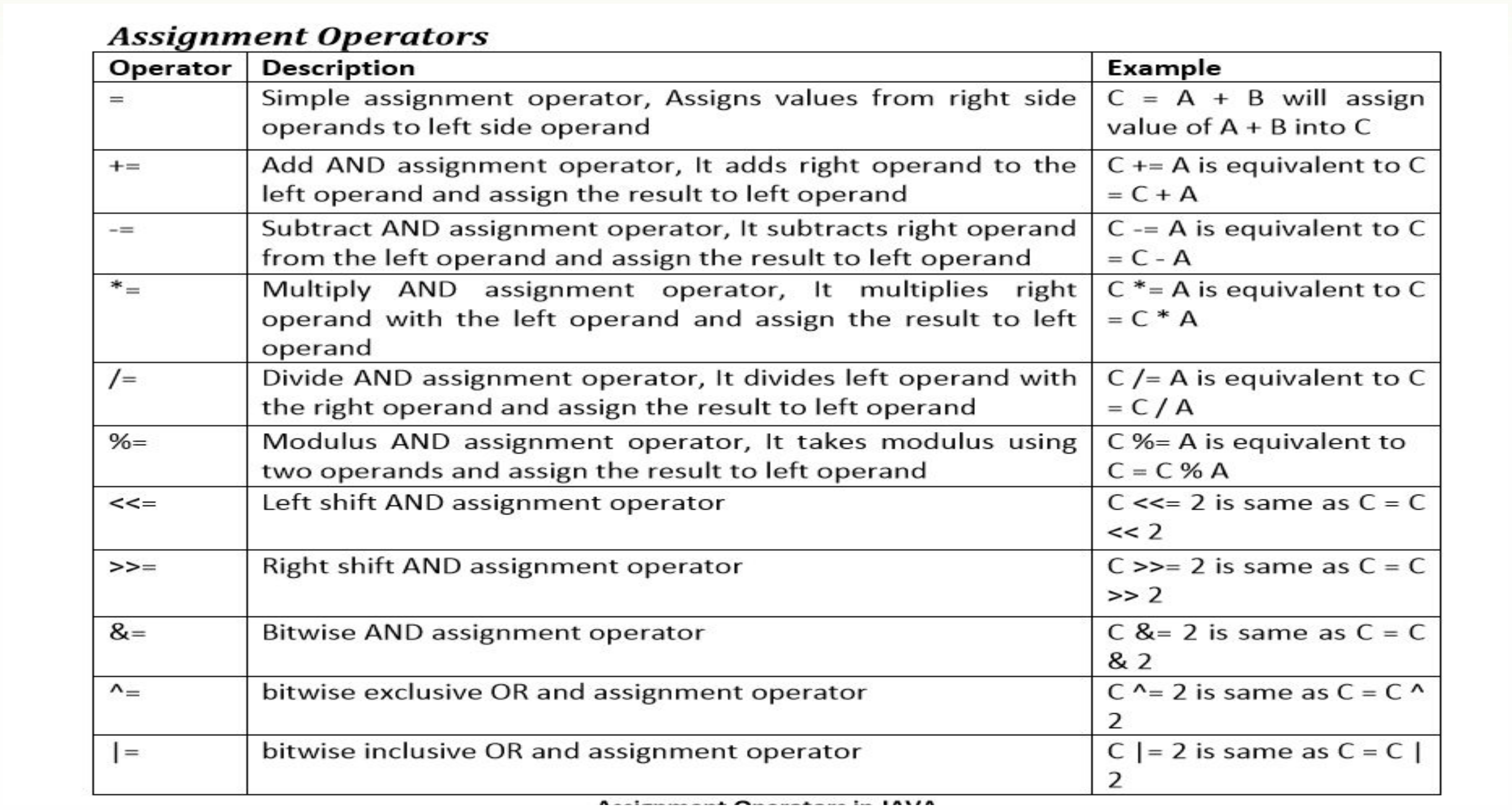
<u>Java Unary Operator Example</u>: ~ and !

```
class OperatorExample  {
     public static void main(String args[]) {
         int a=10; 4. int b=-10;
         boolean c=true;
         boolean d=false;
         System.out.println(~a);//-11 (minus of total positive value which starts from 0)
         System.out.println(~b);//9 (positive of total minus, positive starts from 0)
         System.out.println(!c);//false (opposite of boolean value)
         System.out.println(!d);//true
      }
}
```

Output:

```
-11
9
false
true
```

ASSIGNMENT OPERATORS



Assignment Operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign 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

ASSIGNMENT OPERATORS

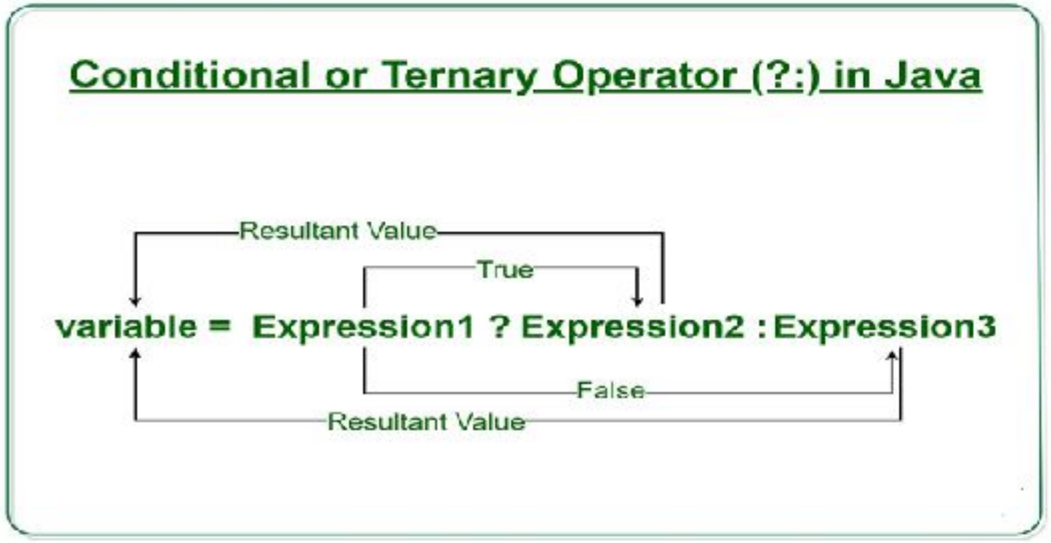
Java Assignment Operator Example

```
class OperatorExample{
      public static void main(String args[]){
         int a=10;
         int b=20;
         a+=4;//a=a+4 (a=10+4)
         b-=4;//b=b-4 (b=20-4)
         System.out.println(a);
         System.out.println(b);
       }
  }
```

Output:

```
14
16
```

TERNARY OPERATORS


Conditional or Ternary Operator (?:) in Java
Resultant Value
True
variable = Expression1 ? Expression2 : Expression3
False
Resultant Value

TERNARY OPERATORS

Java Ternary Operator Example:

```
import java.io.*;
class Ternary {
    public static void main(String[] args)
    {
        // variable declaration
        int n1 = 5, n2 = 10, max;
        System.out.println("First num: " + n1);
        System.out.println("Second num: " + n2);
        // Largest among n1 and n2
        max = (n1 > n2) ? n1 : n2;
        // Print the largest number
        System.out.println("Maximum is = " + max);
    }
}
```

Output:

```
First num: 5
Second num: 10
Maximum is = 10
```

VIDEO LINK

https://www.youtube.com/watch?v=8CX4Tdttbqk