String data type :-

Java also provides support for character string via java.lang.String class. String in Java are not Primitive types. Instead, they are objects.

Eg.

```
String myString = "Java Programming";
```

Here, myString is an object of the string class.

JAVA Operators :-

operators are symbols that perform operations on variables and values. for example, + is an operator used for Addition, while * is also an operator used for multiplication.

operator in java can be classified into 5 types.

1. Arithmatic Operators
2. Assignment Operators
3. Relational Operators
4. Logical operators
5. Bitwise operators

1. Java Arithmatic Operators :- Arithmatic operators are used to perform arithmetic operations on variable and data.

E.g. a + b ;

Here, the + operator is used to add two variables a and b.

operator	operation
+	Addition
-	Subtraction
*	multiplication
/	Division
%	module operation (Remainder after division)

E.g.
```
class main {
Public static void main (String [] args) {

    // declare variables
    int a = 12; b = 5;

    // addition operator
system. out. println (" a + b = " + (a + b));

    // Subtraction operator
system. out. println (" a - b = " + (a - b));

    // multiplication operator
system. out. println (" a * b = " + (a * b));

    // division operator
system. out. println (" a / b = " + (a / b));
```

Sub: Lesson No. Date

```
// Module operator
System.out.println("a %. b = " + (a %. b));
        }
    }
```

Output -

```
a + b = 17
a - b = 7
a * b = 60
a / b = 2
a %. b = 2
```

2. Java Assignment Operators :-

Assignment operators are used in Java to assign values to variables.

E.g.

```
int age;
age = 5;
```

Here, = is the assignment operator. it assigns the value on the right to the variable on its left. That is, 5 is assigned to the variable age.

Operator	Example	Equivalent to
=	a = b;	a = b;
+=	a += b;	a = a + b;
-=	a -= b;	a = a - b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a %. b;

Ex-

```
class main {
Public static void main (String [] args) {

    // Create variables
    int a = 4;
    int var;

// assign value using =
    var = a;
system.out.println ("var using =: " + var);

// assign value using =+
    var += a;
system.out.println ("var using +=: " + var);

// assign value using =*
    var *= a;
System.out.println ("var using *=: " + var);
    }
}
```

Output -

```
var using =: 4
var using +=: 8
var using *=: 32
```

3. Java Relational Operators :-

Relational operators are used to check the Relationship between two operands.

Ex-

```
// check if a is less than b
a<b ;
```

Here, < operator is the Relational operators. it checks if a is less than b or not. it Return Either true or false.

Operator	Description	Example
==	is Equal to	3==5 (Return false)
!=	Not Equal to	3!=5 (Return true)
>	Greater than	3>5 (Return false)
<	Less than	3<5 (Return true)
>=	Greater than or Equal to	3>=5 (Return false)
<=	Less than or Equal to	3<=5 (Return true)

E.g.

```
Class main {
Public static void main ( string [] args)
{
```

```
// Create Variables
int a = 7, b = 11;

// value of a and b
System.out.println("a is " + a + " and b is
                    " + b);

// == operator
System.out.println(a == b); // false

// != operator
System.out.println(a != b); // true

// > operator
System.out.println(a > b); // false

// < operator
System.out.println(a < b); // true

// >= operator
System.out.println(a >= b); // false

// <= operator
System.out.println(a <= b); // true
    }
}
```

O/P - a is 7 and b is 11
false
true
false
true
false → true

4. Java logical Operators :-

Logical operator are used to check weather an Expression is true or false. They are used in decision making.

operator	Example	Meaning
&& (Logical AND)	Expression 1 && Expression 2	true only if both Expression 1 and Expression 2 are true
|| (Logical OR)	Expression 1 || Expression 2	true if either Expression 1 or Expression 2 is true.
! (Logical NOT)	! Expression	true if Expression is false and vice versa.

E.g.

```
Class Main
{
Public static void main (String [] args)
{

// && operator
System.out.println((5>3) && (8>5)); // true
System.out.println((5>3) && (8<5)); // false

// OR (||) operator
```

```
System.out.println((5<3) || (8>5)); // true
System.out.println((5>3) || (8<5)); // true
System.out.println((5<3) || (8<5)); // false

        // NOT (!) operator
System.out.println (!(5==3)); // true
System.out.println (!(5>3)); // false
    }
}
```

O/P - true
false
true
true
false
true
false

Working of Program:-

- (5>3) && (8>5) Returns true because both (5>3) and (8>5) are true.
- (5>3) && (8<5) Returns false because the expression (8<5) is false.
- (5<3) || (8<5) Returns false because both (5<3) and (8<5) is false.
- !(5==3) Returns true because (5==3) is false.
- !(5>3) Returns false because (5>3) is true.

5. Java bitwise Operators :-

Bitwise operators in java are used to perform operations on individual bits.

E.g.

Bitwise Complement Operation of 35

35 = 00100011 (in Binary)

~ 00100011

11011100 = 220 (in decimal)

Here, ~ is a bitwise operator. it inverts the value of each bit (0 to 1 and 1 to 0).

The various bitwise operators present in java are -

Operator	Description
~	Bitwise Complement
<<	Left shift
>>	Right shift
>>>	Unsigned Right shift
&	Bitwise AND
^	Bitwise Exclusive OR

These operators are not generally used in java.

Increment and Decrement operators :-

```
class main {
Public static void main ( String [ ] args)
    {
        // declare Variables
    int a = 12 , b = 12 ;
    int Result1 , Result2 ;

        // original Value
system. out . println (" Value of a :" + a) ;

        // increment operators
    Result1 = ++a ;
system. out . println (" After increment :" + Result1) ;

    system. out . println (" Value of b :" + b) ;

        // decrement operator
    Result2 = --b ;
system. out . println (" After decrement :" + Result2) ;
        }
    }
```

Output -

```
Value of a : 12
After increment : 13
Value of b : 12
After decrement : 11
```

Java Ternary operator :-

The ternary operator (conditional operator) is shorthand for the if - then - else statement.

E.g.

```
variable = Expression ? expression1 : expression2
```

Let's see an example of a ternary operator -

E.g.

```
class java {
Public static void main ( String [ ] args)
        {
            int februaryDays = 29;
            String Result;

        // ternary operator
        Result = (februaryDays == 28) ? "Not a Leap year" :
                                        "Leap year";

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

O/P - Leap year.

In the above example, we have use the ternary operator to check if the year is a leap year or not.

Java Basic Input and Output :-

Java output :-

in Java, you can simply use

```
system.out.println();   or
system.out.print();     or
system.out.printf();
```

to send output to standard output (screen).
Here,

System is a class.
out is a public static field: it accepts output data.

Eg.

```
class AssignmentOperator {
Public static void main (String [] args)
        {
System.out.println ("Java Programming is interesting.");
        }
}
```

Output - Java Programming is interesting.

Here, we have used to println() method to display the string.