String type

Java also provides support for character strings via java.lang.String class. Strings in Java are not primitive types. Instead, they are objects. For example,

```
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.

Operators in Java can be classified into below types:

1. Arithmetic Operators
2. Assignment Operators
3. Relational Operators
4. Logical Operators
5. Unary Operators
6. Bitwise Operators

1. Java Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on variables and data. For example,

a + b;

Here, the + operator is used to add two variables a and b. Similarly, there are various other arithmetic operators in Java.

Operator	**Operation**
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Operation (Remainder after division)

2. Java Assignment Operators

Assignment operators are used in Java to assign values to variables. For example,

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

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

Let's see some more assignment operators available in Java.

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;

3. Java Relational Operators

Relational operators are used to check the relationship between two operands. For example,

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

Here, > operator is the relational operator. It checks if a is less than b or not.

It returns either true or false.

Operator	**Description**	**Example**
==	Is Equal To	3 == 5 returns false
!=	Not Equal To	3 != 5 returns true
>	Greater Than	3 > 5 returns false
<	Less Than	3 < 5 returns true
>=	Greater Than or Equal To	3 >= 5 returns false
<=	Less Than or Equal To	3 <= 5 returns true

4. Java Logical Operators

Logical operators are used to check whether an expression is true or false. They are used in decision making.

Operator	Example	Meaning
&& (Logical AND)	expression1 **&&** expression2	true only if both expression1 and expression2 are true
|| (Logical OR)	expression1 **||** expression2	true if either expression1 or expression2 is true
! (Logical NOT)	**!**expression	true if expression is false and vice versa

5. Java Unary Operators

Unary operators are used with only one operand. For example, ++ is a unary operator that increases the value of a variable by 1. That is, ++5 will return 6.

Different types of unary operators are:

Operator	**Meaning**
+	Unary plus: not necessary to use since numbers are positive without using it
-	Unary minus: inverts the sign of an expression
++	Increment operator: increments value by 1
--	Decrement operator: decrements value by 1
!	Logical complement operator: inverts the value of a boolean

6. Java Bitwise Operators

Bitwise operators in Java are used to perform operations on individual bits. For example,

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).

Operator	**Description**
~	Bitwise Complement
<<	Left Shift
>>	Right Shift
>>>	Unsigned Right Shift
&	Bitwise AND
^	Bitwise exclusive OR

These operators are not generally used in Java.

Java instanceof Operator

The **instanceof** operator checks whether an object is an instanceof a particular class. For example,

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

    String str = "Programming";
    boolean result;

    // checks if str is an instance of
    // the String class
    result = str instanceof String;
    System.out.println("Is str an object of String? " + result);
  }
}
```

Java Ternary Operator

The ternary operator (conditional operator) is shorthand for the if-then-else statement.
For example,
variable = Expression ? expression1 : expression2

If the Expression is true, expression1 is assigned to the variable.
If the Expression is false, expression2 is assigned to the variable.

```
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);
  }
}
```

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.

Difference between println(), print() and printf()

print() - It prints string inside the quotes.

println() - It prints string inside the quotes similar like print() method. Then the cursor moves to the beginning of the next line.

printf() - It provides string formatting (similar to printf in C/C++ programming).

Example: Printing Variables and Literals

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

        Double number = -10.6;

        System.out.println(5);
        System.out.println(number);
    }
}
```

When you run the program, the output will be:

```
5
-10.6
```

Here, you can see that we have not used the quotation marks. It is because to display integers, variables and so on, we don't use quotation marks.

Example: Print Concatenated Strings

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

        Double number = -10.6;

        System.out.println("I am " + "awesome.");
        System.out.println("Number = " + number);
    }
}
```

Output:

```
I am awesome.
Number = -10.6
```

In the above example, notice the line,

```
System.out.println("I am " + "awesome.");
```