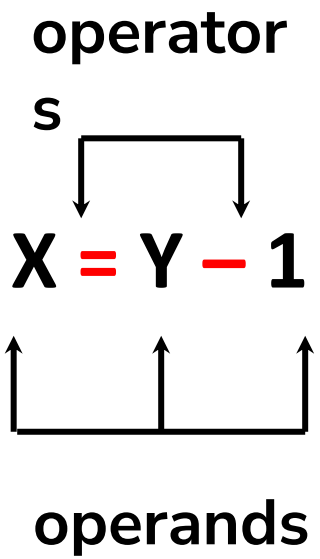

FACE

Operators

FACE

Operators are used to perform **operations** on variables and values.

- Unary operators
- Binary operators
- Ternary operators



FACE

Operators in Java

- Unary operators
- Arithmetic operators
- Relational operators
- Bitwise operators
- Logical operators
- Ternary operators
- Assignment operators
- instanceof operator

FACE

Unary operators

- Unary operator performs operation on only one operand.
- They are used to increment, decrement or negate a value.

FACE

Unary operators

Operator	Description	Example
++	Increases the value of a variable by 1	++x(prefix) x++(postfix)
--	Decreases the value of a variable by 1	--x(prefix) x--(postfix)
+	Used for giving positive values	+ x
-	Used for negating the values	- x
~	Negating an expression	~ x
!	Inverting the value of a boolean	! x

FACE

What is the difference between
++x and **x++** ?

FACE

```
1  class OperatorExample{
2        public static void main(String args[]){
3
4              int x=10;
5              System.out.println(x++);
6              System.out.println(++x);
7              System.out.println(x--);
10             System.out.println(--x);
11       }
12 }
```

output

```
10
12
12
10
```

```
1  class OperatorExample{
2     public static void main(String args[]){
3        int a=10;
4        int b=-10;
5        boolean c=true;
6        boolean d=false;
7        System.out.println(~a);
10       System.out.println(~b);
11       System.out.println(!c);
12       System.out.println(!d);
13    }
14 }
```

output

```
-11
9
False
true
```

FACE

Arithmetic operators

- Arithmetic operators are used to perform common mathematical operations like addition, subtraction etc..

FACE

Arithmetic operators

Operator	Description	Example
+	Adds two values	x + y
-	Subtracts one value from another	x - y
*	Multiplies two values	x * y
/	Returns the division quotient	x / y
%	Returns the division remainder	x % y

FACE

What is the difference between
/ and **%** **?**

FACE

```
1  class OperatorExample{
2      public static void main(String args[]){
3        int a=10;
4        int b=5;
5        System.out.println(a+b);
6        System.out.println(a-b);
7        System.out.println(a*b);
10       System.out.println(a/b);
11       System.out.println(a%b);
12     }
13 }
```

output

```
15
5
50
2
0
```

Relational operators

- Relational/Comparison operators are used to compare two values.

- They return **boolean** result after the comparison.

FACE

Relational operators

Operator	Description	Example
==	Returns true if left hand side is equal to right hand side	x == y
!=	Returns true if left hand side is not equal to right hand side	x != y
<	Returns true if left hand side is less than right hand side	x < y
<=	Returns true if left hand side is less than or equal to right hand side	x < =y
>	Returns true if left hand side is greater than right hand side	x >=y
>=	Returns true if left hand side is greater than or equal to	x >= y

```
1  public class Test {
2     public static void main(String args[]){
3        int a = 10;
4        int b = 20;
5        System.out.println(a>b);
6        System.out.println(a<b);
7        System.out.println(a>=b);
10       System.out.println(a<=b);
11       System.out.println(a==b);
12       System.out.println(a!=b);
13    }
14 }
```

output

```
false
true
false
true
false
true
```

Bitwise operators

- Bitwise operator works on bits and performs bit-by-bit operation.

- Can be applied to the integer types, long, int, short, char, and byte.

FACE

Bitwise operators

Operator	Description	Example
&	Returns bit by bit AND of input values	x & y
|	Returns bit by bit OR of input values	x | y
^	Returns bit by bit XOR of input values	x ^ y
~	Returns the one's compliment representation of the input value	~ x
<<	shifts the bits of the number to the left and fills 0 on voids left as a result	x << 2
>>	shifts the bits of the number to the right	x >> 2
>>>	shifts the bits of the number to the right and fills 0 on voids left as a result	x >>>2

FACE

```java
public class Main {
    public static void main(String args[]){
        int a = 10;
        int b = 20;
        System.out.println(a&b);
        System.out.println(a|b);
        System.out.println(~a);
        System.out.println(a<<2);
        System.out.println(a>>2);
        System.out.println(a>>>2);
    }
}
```

output

```
0
30
-11
40
2
2
```

Logical operators

- The logical operators || (conditional-OR) and && (conditional-AND) operates on boolean expressions.

- The second condition is not evaluated if the first one is false, i.e. it has a **short-circuiting effect**.

FACE

Logical operators

Operator	Description	Example
&&	Returns true if both statements are true	x < 5 && x < 10
||	Returns true if one of the statements is true	x < 5 || x < 4
!	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

```
1  public class Test {
2     public static void main(String args[]){
3        boolean a = true;
4        boolean b = false;
5        System.out.println(a&&b);
6        System.out.println(a||b);
7        System.out.println(!(a && b));
10    }
11 }
```

output

```
false
true
true
```

FACE

Ternary operator

- Ternary/Conditional operator consists of three operands and is used to evaluate Boolean expressions.
- Ternary operator is a **shorthand version of if-else statement**.
- It has three operands and hence the name ternary.

FACE

```java
class OperatorExample{
    public static void main(String args[]){
        int a=2;
        int b=5;
        int min=(a<b)?a:b;
        System.out.println(min);
    }
}
```

output

```
2
```

Assignment operators

- Assignment operator is used to assign a value to any variable.

- In many cases assignment operator can be combined with other operators to build a shorter version of statement called **Compound Statement**.

FACE

Assignment operators

Operator	Description	Example
=	Assigns values from right side operands to left side operand.	C = A + B
+=	Adds right operand to the left operand and assign the result to left operand.	C += A
-=	Subtracts right operand from the left operand and assign the result to left operand.	C -= A
*=	Multiplies right operand with the left operand and assign the result to left operand.	C *= A
/=	Divides left operand with the right operand and assign the result to left operand.	C /= A
%=	Takes modulus using two operands and assign the result to left operand.	C %= A

FACE

Assignment operators

Operator	Description	Example
<<=	Left shift AND assignment operator	C <<= 2
>>=	Right shift AND assignment operator	C >>= 2
&=	Bitwise AND assignment operator	C &= 2
^=	Bitwise exclusive OR and assignment operator	C ^= 2
|=	Bitwise inclusive OR and assignment operator	C |= 2

FACE

What is the difference between
= and == ?

FACE

```
1  class OperatorExample{
2    public static void main(String[] args){
3        int a=10;
4        a+=3;
5        System.out.println(a);
6        a-=4;
7        System.out.println(a);
10       a*=2;
11       System.out.println(a);
12       a/=2;
13       System.out.println(a);
14   }
15 }
```

output

```
13
9
18
9
```

instanceof operators

- Instanceof operator is used only for object reference variables.

- The operator checks whether the object is of a particular type (class type or interface type).

FACE

```
1  public class Test {
2   public static void main(String args[]) {
3      String name = "James";
4      boolean result = name instanceof String;
5      System.out.println( result );
6   }
7  }
10
11
```

output

```
true
```

Precedence and associativity

- **Operator precedence** determines which operator is evaluated first when an expression has more than one operators.

- **Associativity** is used when there are two or more operators of same precedence is present in an expression.

FACE

Operator	Description	Associativity

`()`
`[]`
`.`
	method invocation
	array subscript
	member access/selection
		left-to-right

`++` `--`	unary postfix increment unary postfix decrement	right-to-left

`++`
`--`
`+`
`-`
`!`
`~`
`( type )`
`new`
	unary prefix increment
	unary prefix decrement
	unary plus
	unary minus
	unary logical negation
	unary bitwise complement
	unary cast
	object creation
		right-to-left

`*`
`/`
`%`
	multiplication
	division
	modulus (remainder)
		left-to-right

`+` `-`	addition or string concatenation subtraction	left-to-right

`<<`
`>>`
`>>>`
	left shift
	arithmetic/signed right shift (sign bit duplicated)
	logical/unsigned right shift (zero shifted in)
		left-to-right

`<`
`<=`
`>`
	less than
	less than or equal to
	greater than
		left-to-right

FACE

`>`
`>=`
`instanceof`
	greater than
	greater than or equal to
	type comparison	

`==` `!=`	is equal to (equality) is not equal to (inequality)	left-to-right
`&`	bitwise AND boolean logical AND (no short-circuiting)	left-to-right
`^`	bitwise exclusive OR boolean logical exclusive OR	left-to-right
`|`	bitwise inclusive OR boolean logical inclusive OR (no short-circuiting)	left-to-right
`&&`	logical/conditional AND (short-circuiting)	left-to-right
`||`	logical/conditional OR (short-circuiting)	left-to-right
`?:`	conditional/ternary (if-then-else)	right-to-left

`=`
`+=`
`-=`
`*=`
`/=`
`%=`
`&=`
`^=`
`|=`
`<<=`
`>>=`
`>>>=`
	assignment
	addition assignment
	subtraction assignment
	multiplication assignment
	division assignment
	modulus/remainder assignment
	bitwise AND assignment
	bitwise exclusive OR assignment
	bitwise inclusive OR assignment
	bitwise left shift assignment
	bitwise arithmetic/signed right shift assignment
	bitwise logical/unsigned right shift assignment
		right-to-left

```
1  // Predict the output
2  public class A {
3      public static void main(String[] args)
4      {
5          int $_ = 5;
6      }
7  }
```

OUTPUT

1. Nothing
2. Error

```
1  // Predict the output
2  class Test {
3      public static void main(String args[])  {
4          System.out.println(10  +  20 + “Face");
5          System.out.println(“Face" + 10 + 20);
6      }
7  }
```

OUTPUT

1. 30Face Face30
2. 1020Face Face1020
3. 30Face Face1020
4. 1020Face Face30

```
1   // Predict the output
2   class Test
3   {
4       public static void main(String args[])
5       {
6           String s1 = “FACE";
7           String s2 = “FACE";
8               //System.out.println(s1==s2);
9           System.out.println("s1 == s2 is:" + s1 == s2);
10      }
    }
```

OUTPUT

1. true
2. false
3. compiler error
4. throws an exception

THANK YOU

FACE