Operators: An operator is a symbol that tells the compiler to perform specific mathematical or logic function.

1. Arithmetic Operator:
2. Relational Operator
3. Logical Operator
4. Conditional Operator
5. Assignment Operator
6. Bitwise Operator
7. Shift Operator

Arithmetic Operators:

+	Addition
-	Substraction
*	Multiplication
/	Division
%	Modulus
Unary Operator:	
In++	Post Increment
De--	Post Decrement

++In	Pre Increment
--De	Pre Decrement
+=	Addition assignment
-=	Substraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment

Program:

```
public class ArithmeticOperator {
     public static void main(String[] args)
     {
       int a = 9;
       int b = 2;
       int add = a+b;
       int subs = a-b;
       int multi = a*b;
       int divi = a/b;
       int modulo = a%b;
       System.out.println("Addition of 9 + 2 = " + add);
       System.out.println("Substraction of 9 - 2 = " + subs);
       System.out.println("Multiplication of 9 * 2 = " + multi);
       System.out.println("Division of 9 / 2 = " + divi);
       System.out.println("Modulus of 9 % 2 = " + modulo);
    }
}
```

Increment(++) & Decrement Operator(--) :

The increment operator (++) increases its operand by one AND The decrement operator (- -) decreases its operand by one.

Example: x = x + 1 → x++; x = x - 1 → x—;

a++: Post increment -> Next Statement
++a; Pre increment -> Same statement
a--; Post decrement -> Next Statement
--a; Pre decrement -? Same statement

Program:

```java
public class IncOrDec2 {
  public static void main(String[] args) {
     int x = 21;
     x++;
    System.out.println(x++); //Print = 22, Store = 23
    System.out.println(++x); //Print = 24, Store = 25
    System.out.println(x--); //Print = 24, Store = 23
    System.out.prinltn(--x); //Print = 22, Store = 20
     }
}
```

Relational Operator: The relational operators determines the relationship that one operand has to the other. Specifically, they determine equality and ordering

Operator	Result
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

→The outcome of these operation is a Boolean value. The relationship operators are most frequently used in the expression that control the if statement and the various loop statements. Any type in Java, including Integers, Floating-point numbers, characters, and Boolean can be compared using the equality test, ==, and the inequality test, !=.

Remember: A single equal sign is the assignment operator, and two equal signs is used to denoted equality.

```
if(!equal)... // Valid in C/C++, but not valid in Java.
    This statement must be written like:
if (done == 0) // This is Java-Style
if (done != 0)
```

Program:

```java
// Writing a program on Relational Operator
public class RelationalOperator {
    public static void main(String[] args)
    {
      int dhruv = 21;
      int lenka = 19;
      System.out.println(dhruv == lenka);
      System.out.println(dhruv != lenka);
      System.out.println(dhruv > lenka);
      System.out.println(dhruv < lenka);
      System.out.println(dhruv >= lenka);
      System.out.println(dhruv <= lenka);
    }
}
```

Logical Operator: The logical operators combine two boolean values to form a resultant boolean value. The logical (&&) operator doesn't
check the second condition if the first condition is false. It checks second condition only when first condition is true. The logical (||) operator doesn't check second condition if the first condition is true. It checks second condition only when first condition is false.

Operator	Result
&	Logical AND
&&	Short-circuit AND
|	Logical OR
||	Short-circuit OR
!	Logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment

A	B	A ^ B	!A
FALSE	FALSE	FALSE	TRUE
TRUE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	TRUE
TRUE	TRUE	FALSE	FALSE

A	B	A | B	A & B
FALSE	FALSE	FALSE	FALSE
TRUE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE

```java
public class LogicalOperator2 {
    public static void main(String[] args) {
        int a = 21;
        int b = 18;
        int c = 19;
        System.out.println(a<b && a<c); //false
        System.out.println(a<b || b<c); //true
    }
}
```

ANDOperator: The bitwise (&) operator always checks both condition whether first condition true or false.

Program:

```
public class LogicalOperator2 {
    public static void main(String[] args) {
        int a = 21;
        int b = 18;
        int c = 19;
        System.out.println(a<b && a<c); //false
        System.out.println(a<b & a<c); //false
    }
}
```

OR Operator: The bitwise (|) operator always checks both condition whether first condition true or false.

```java
public class OROperator {
    public static void main(String[] args) {
        int x = 40;
        int y = 10;
        int z = 20;
        System.out.println(x>y||x<z);  //true || true = true
        System.out.println(x>y|x<z);  //true | true = true
        System.out.println(x>y||x++<z);  //true || true = true
        System.out.println(x);  //40 because second condition is not checked
        System.out.println(x>y|x++<z);  //true | true = true
        System.out.println(x);  //41 because second condition is checked
    }
}
```

Bit-wise Operator:

The logical ! Operator inverts the Boolean state: !true == false and !false == true.

Operator	Result
~	Bitwise unary NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
>>	Shift right
>>>	Shift right zero fill
<<	Shift Left
&=	Bitwise AND assignment
|=	Bitwise OR assignment
^=	Bitwise exclusive OR assignment
>>=	Shift right assignment
>>>=	Shift right zero fill assignment
<<=	Shift left assignment

Left Shift Operator: The Left Shift Operator is a type of Bitwise Shift Operator, which performs operations on the binary bits. It is a binary operator that requires two operands to shift or move the position of the bits to the left side and add zeroes to the empty space created at the right side shifting the bits.

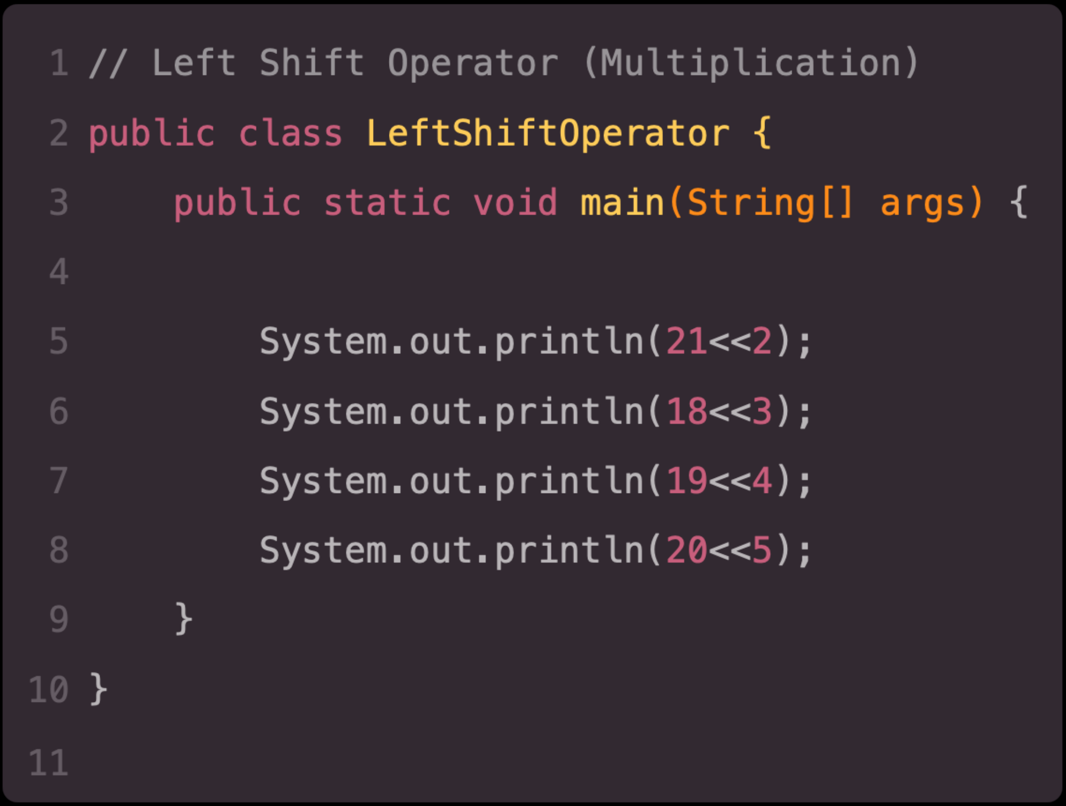
Method:

1. Suppose, 10<<2, We convert 10 into binary and remove 2 bits from left and fill empty spaces as zero(0) in right bits.
2. Then, We convert Binary to Decimal

Shortcut Method:

Left Shift Operator (Value * 2^n Multiplication)

- Suppose 10<<2 = 10*2*2 = 40



```java
// Left Shift Operator (Multiplication)
public class LeftShiftOperator {
    public static void main(String[] args) {

        System.out.println(21<<2);
        System.out.println(18<<3);
        System.out.println(19<<4);
        System.out.println(20<<5);
    }
}

```

Right Shift Operator: The Right Shift Operator is a type of bitwise shift operator used to move the bits at the right side. Like the left side operator, the right shift operator also requires two operands to shift the bits at the right side and then insert the zeroes at the empty spaces created at the left side after shifting the bits.

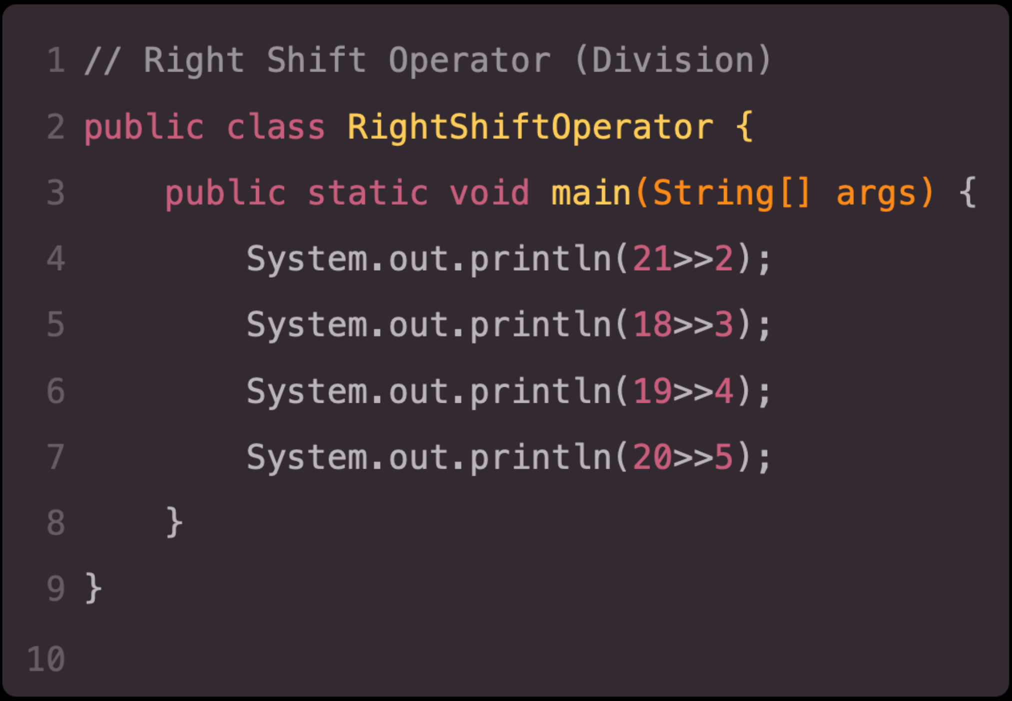
Method:

1. Suppose 10>>2, We convert left side operand to binary and remove two bits from the right and fill empty spaces as zero(0) in left bits.
2. Then, We convert Binary to Decimal

Shortcut Method:

Right Shift Operator(Value / 2^n Division)

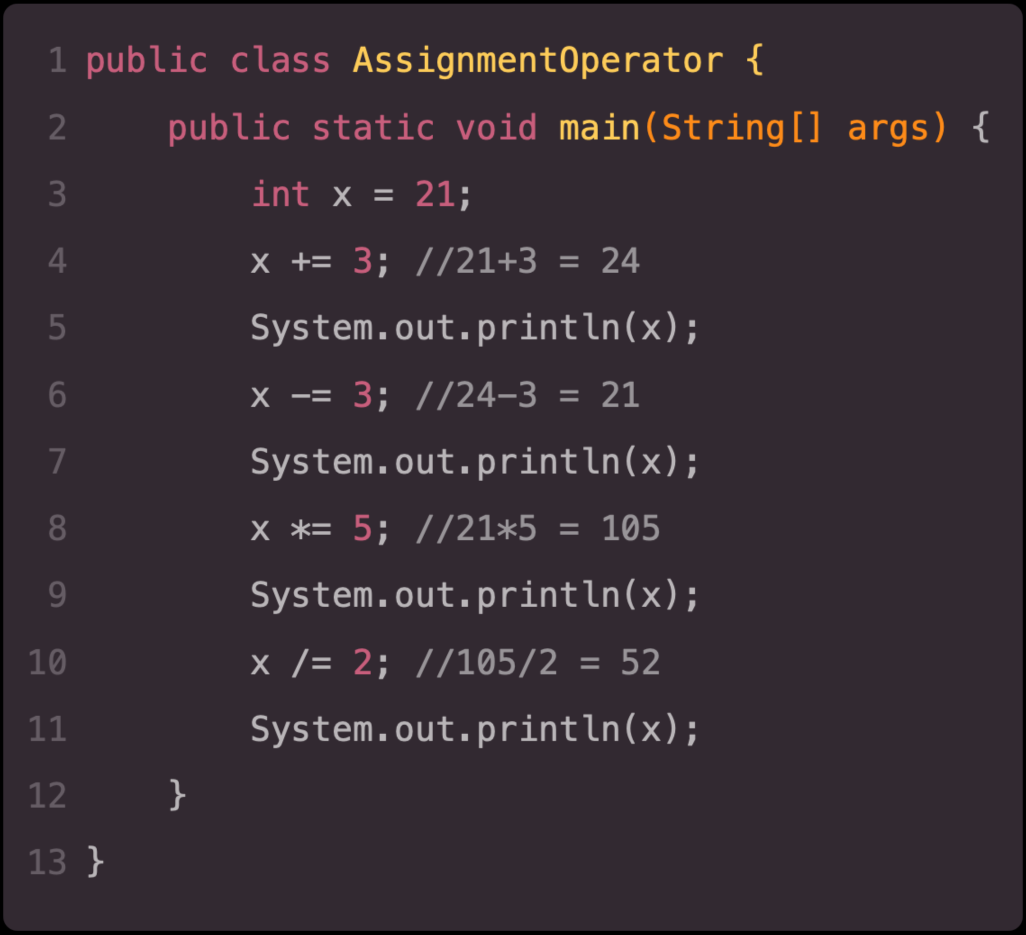
- Suppose 10>>2 = 10/2*2 = 10/4 = 2



```java
// Right Shift Operator (Division)
public class RightShiftOperator {
    public static void main(String[] args) {
        System.out.println(21>>2);
        System.out.println(18>>3);
        System.out.println(19>>4);
        System.out.println(20>>5);
    }
}
```

Assignment Operator: Assignment operators are used in Java to assign values to variables.
It assigns the value on its right to the variable on its left.
Ex:
int age = 21;

+=	Addition assignment
-=	Substraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment



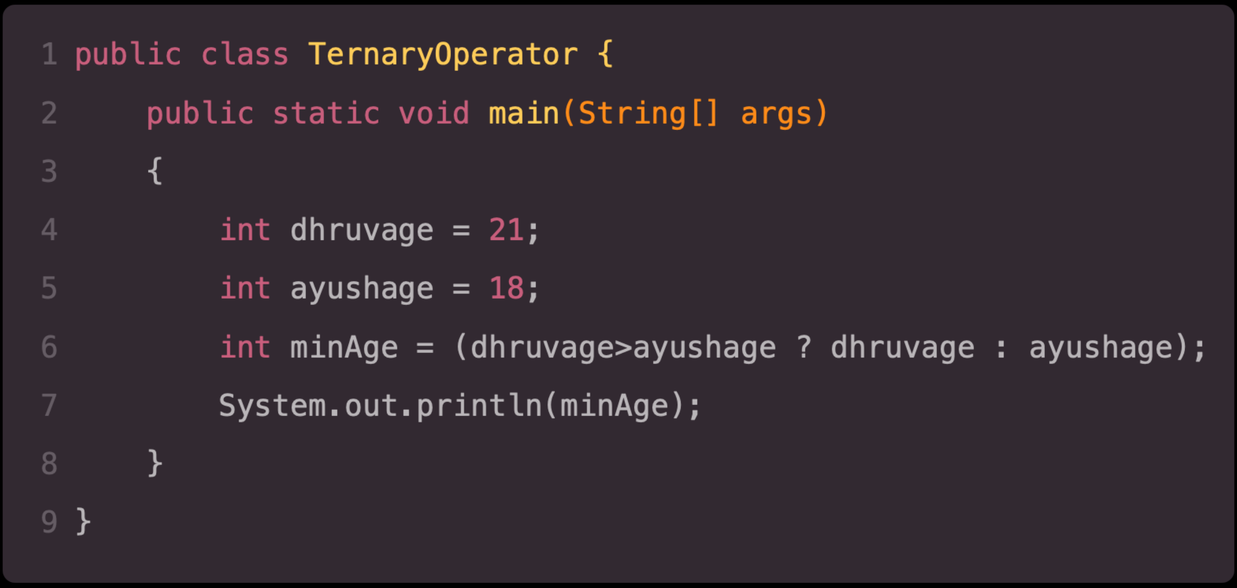
```java
public class AssignmentOperator {
    public static void main(String[] args) {
        int x = 21;
        x += 3; //21+3 = 24
        System.out.println(x);
        x -= 3; //24-3 = 21
        System.out.println(x);
        x *= 5; //21*5 = 105
        System.out.println(x);
        x /= 2; //105/2 = 52
        System.out.println(x);
    }
}
```

Ternary Operator(?) : Java Ternary Operator is used as one line replacement for if-then else statement. It is only conditional operator which takes three operands.

Ex:

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.



```java
public class TernaryOperator {
    public static void main(String[] args)
    {
        int dhruvage = 21;
        int ayushage = 18;
        int minAge = (dhruvage>ayushage ? dhruvage : ayushage);
        System.out.println(minAge);
    }
}
```