# Operators (Chapter 3 of Schilit)

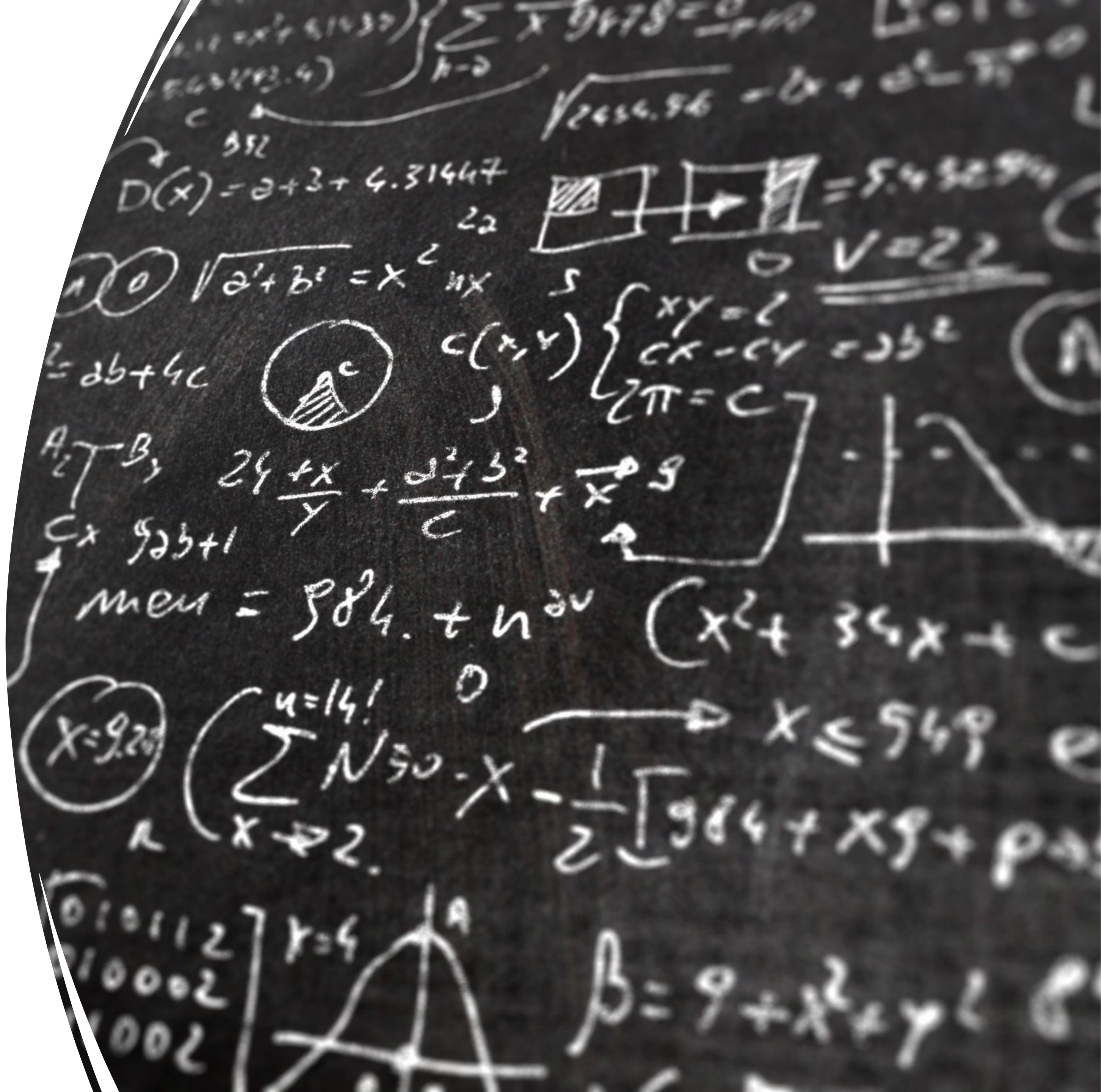
Object Oriented Programming BS (AI and MMG) II

Compiled By

Abdul Ghafoor

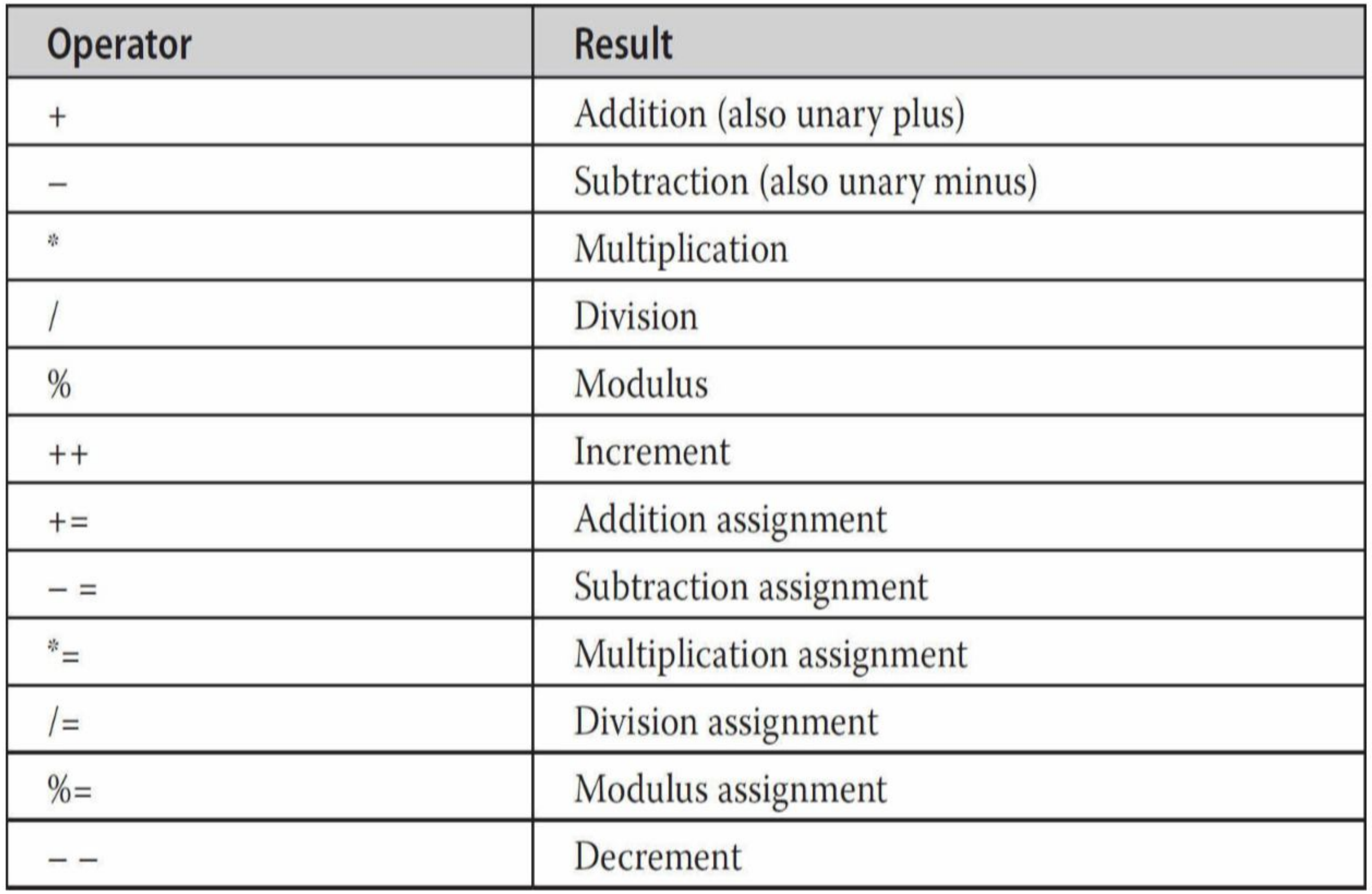
# Operators

- Arithmetic
- Bitwise
- Relational
- Logical



# Arithmetic operators

- Operands to these
- operators must be
- numeric



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

## Example (arithmetic with int and double)

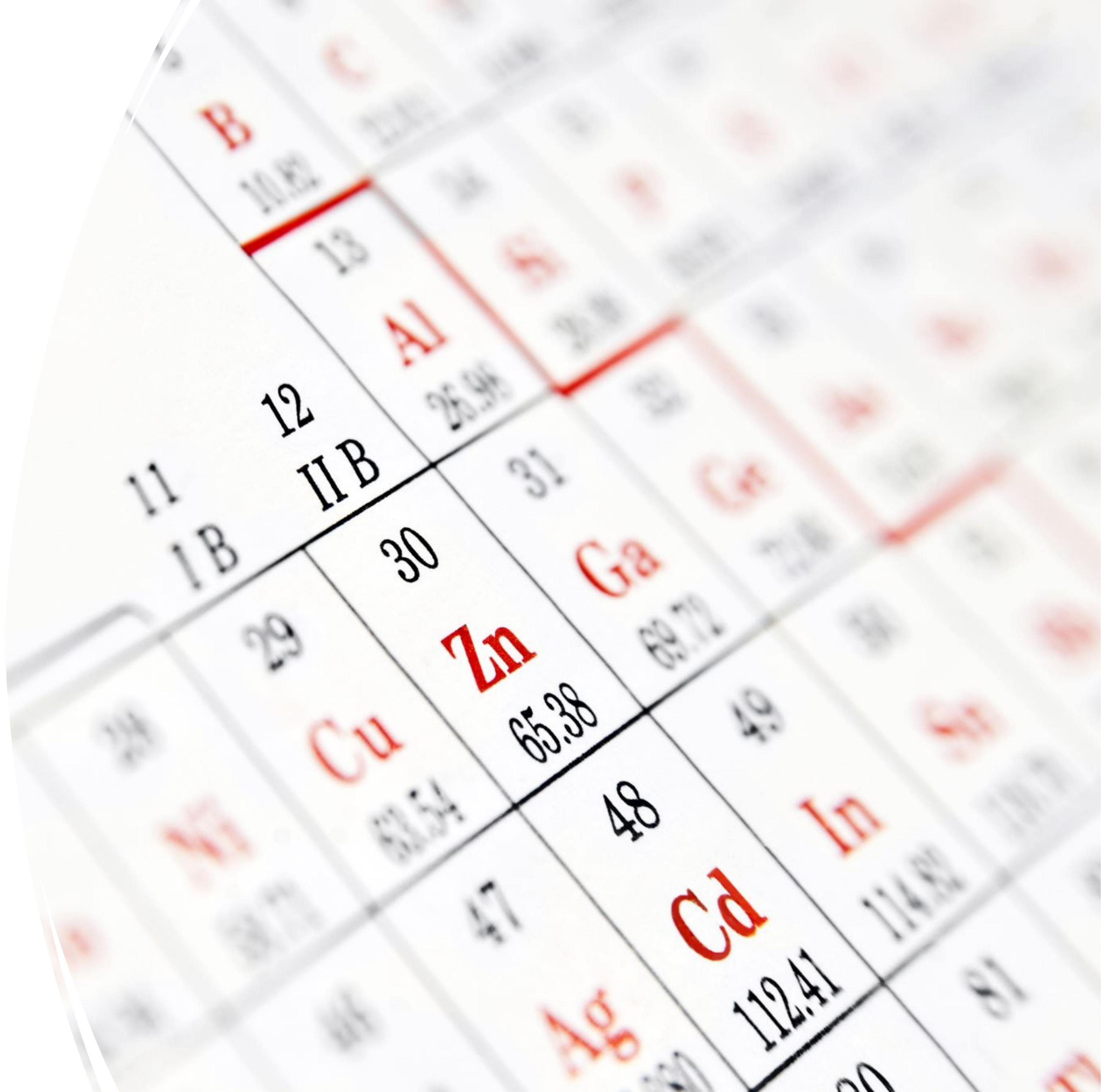
## Unary Operator

- Unary Minus (-)
- NOT(!)
- Increment(++) (pre & post)
- Decrement(--) (pre & post)

## Modulus Operator(%)

- Floating
- Integer
- What happens when left side is smaller than right side?

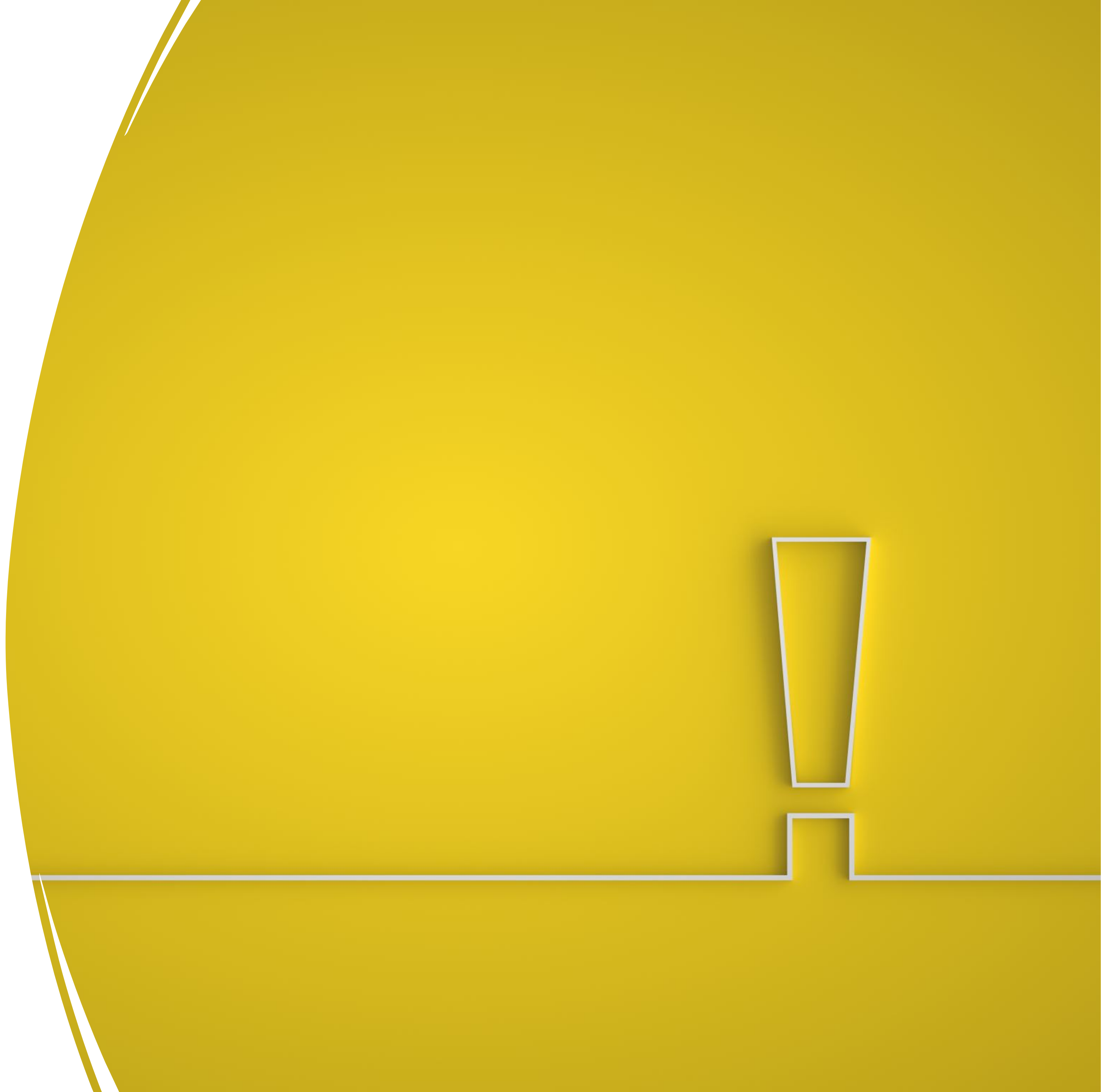
## Take a floating point number as input, find its remainder when divided with 5



## Compound Assignment Operators

var = <var> op <expression> **Equal to** var op= <expression>;

In Java, **compound assignment operators** are shorthand notations for performing an operation on a variable and assigning the result back to the same variable. They simplify expressions and make the code more concise.



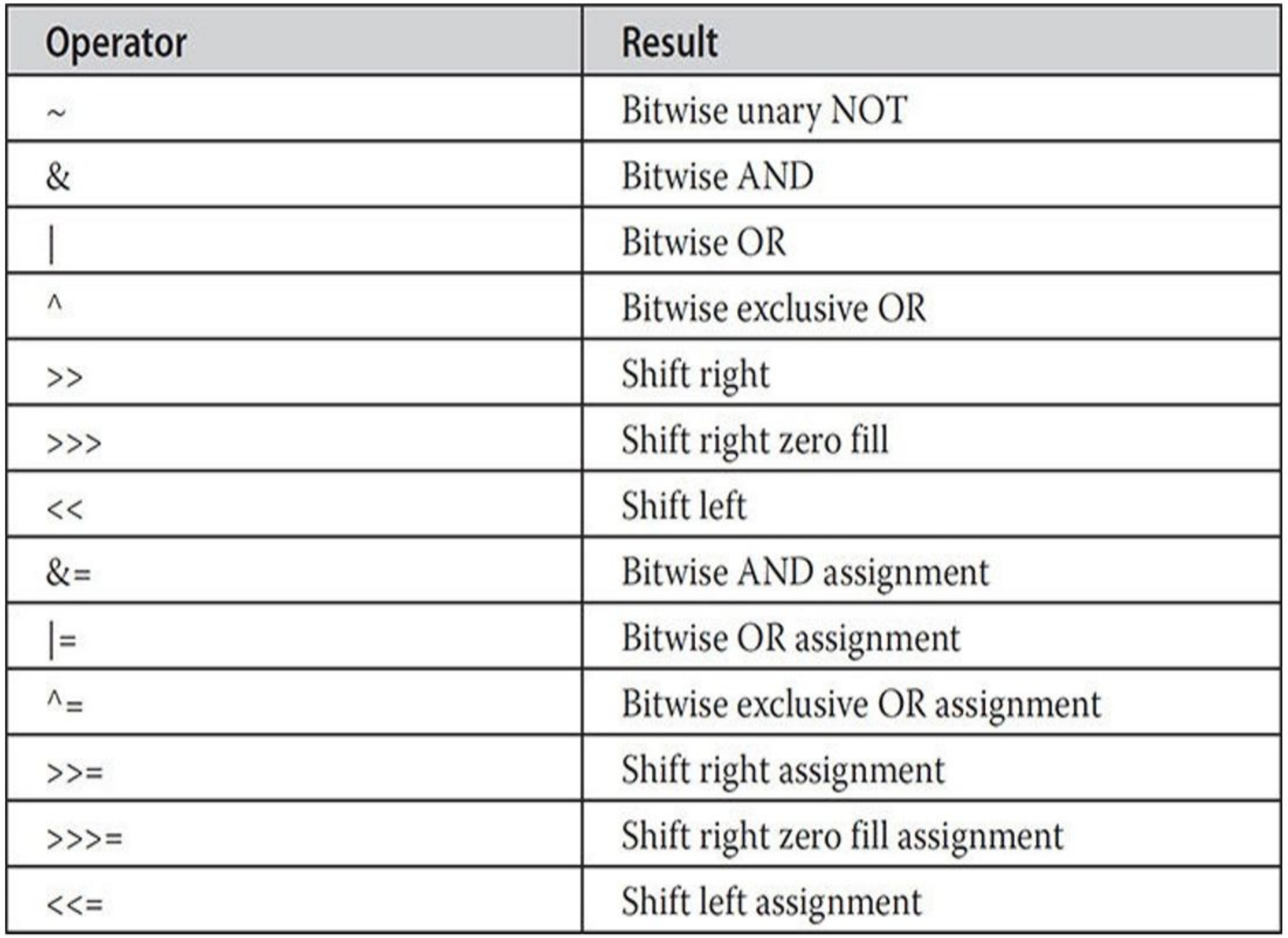
## Example (compound operator)

## How integers are stored in memory by Java and representation of sign

- In java integers are signed:
    - Store negative as well as positive values

- To store negative numbers, use the concept of Two’s complement:
    - Invert all the bits and add 1 to the result from LSB
    - Example 8 is represented in binary as 00001000
    - Invert all bits= 11110111
    - +1
    - 10000000

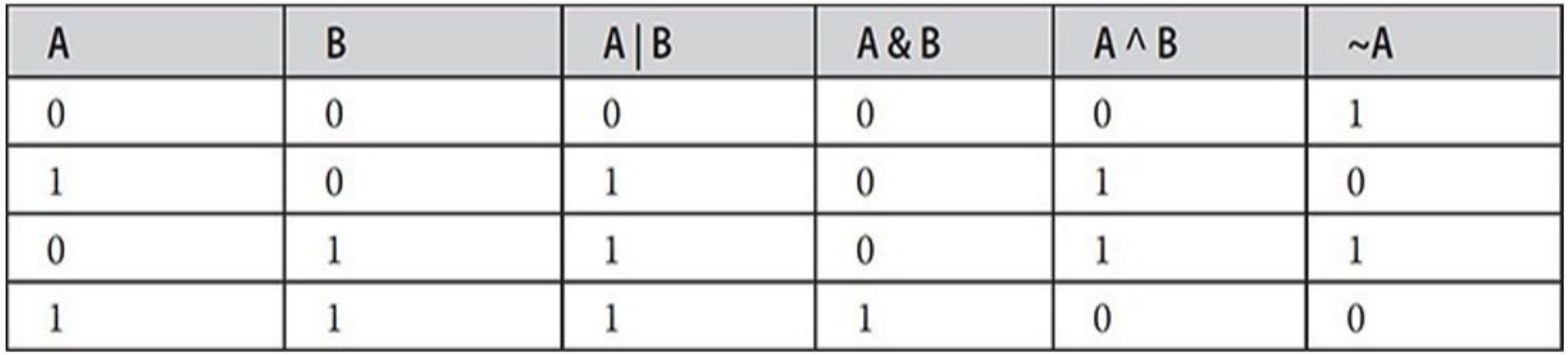
## Bitwise Operators



| 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 |

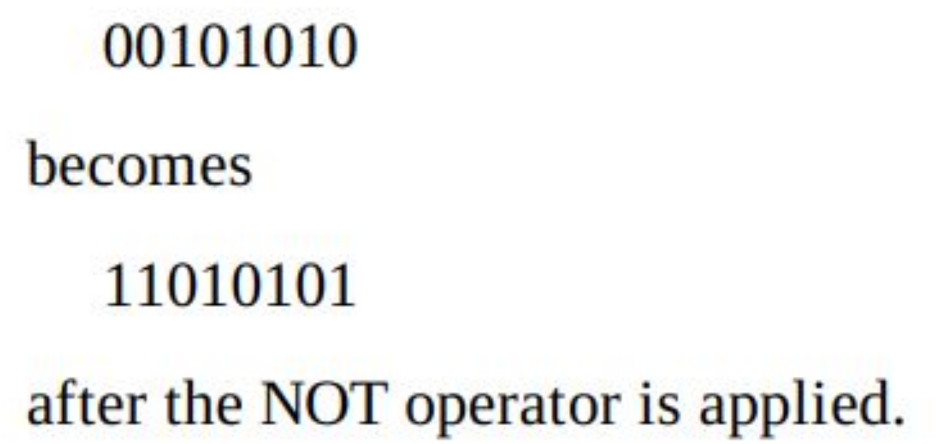
# Bitwise Logical Operators

&, |, ^, and ~



| A | B | A\|B | A&B | A^B | ~A |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

## Bitwise NOT(Complement) ~



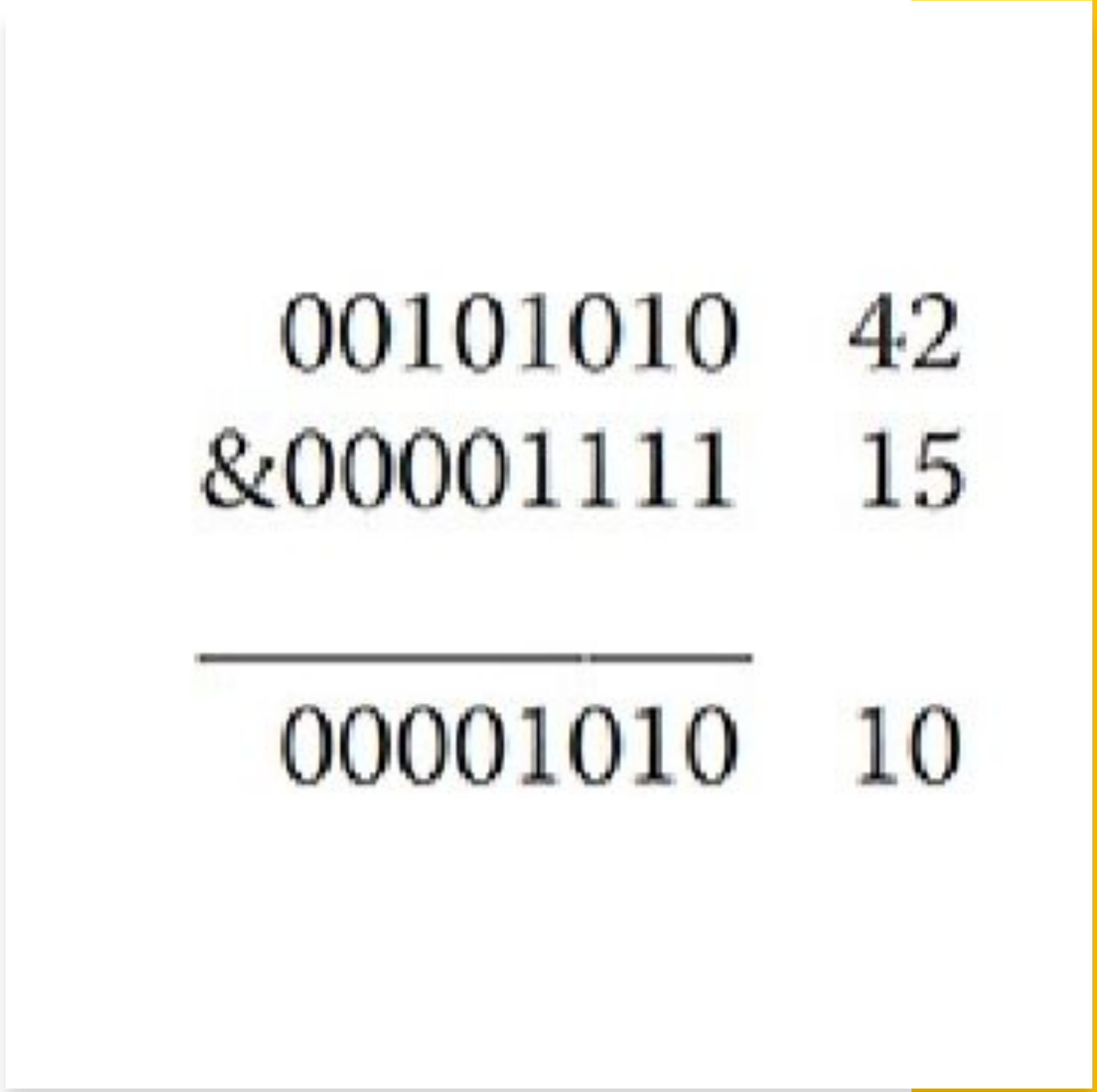
00101010

becomes

11010101

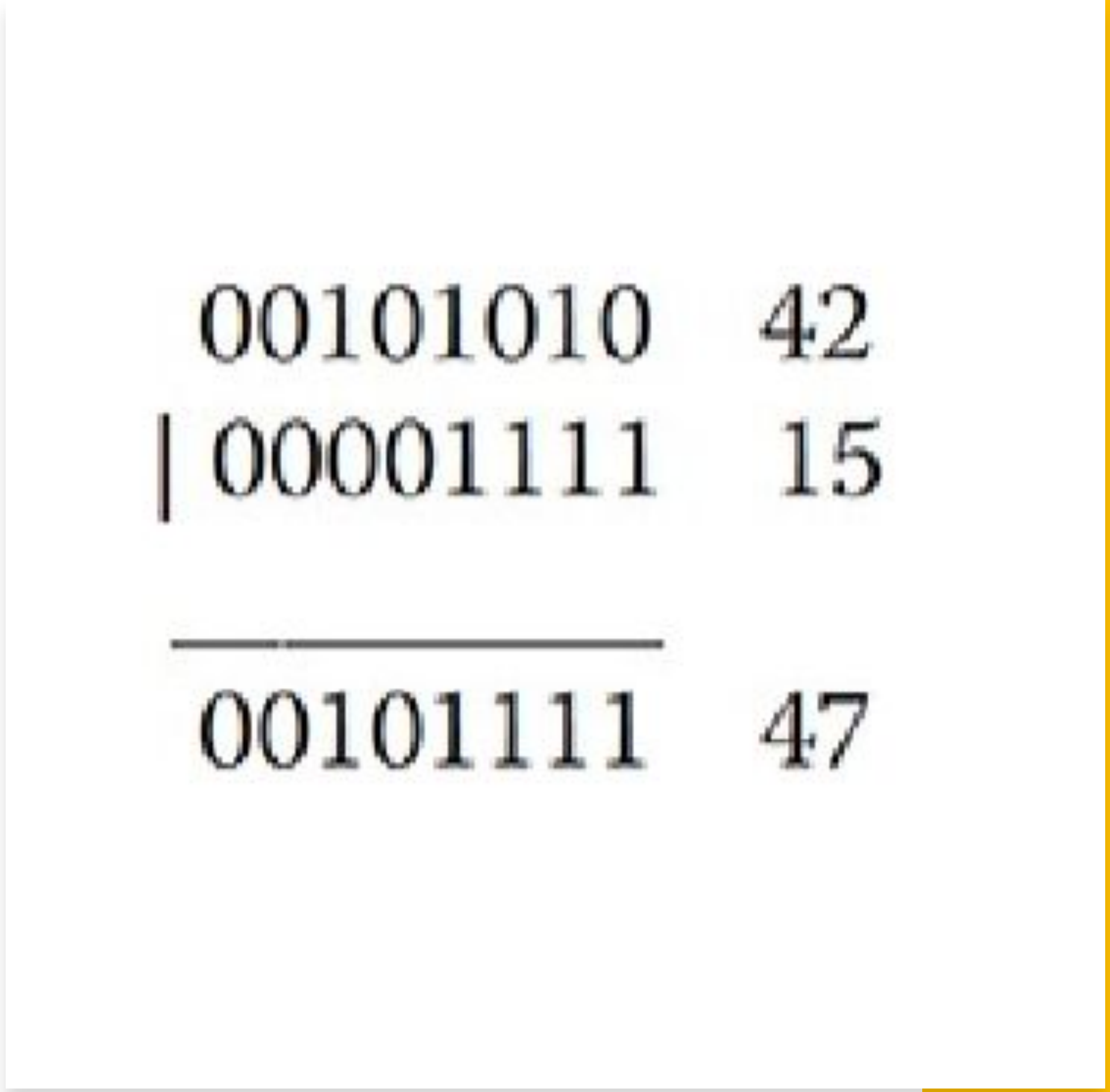
after the NOT operator is applied.

## Bitwise AND &



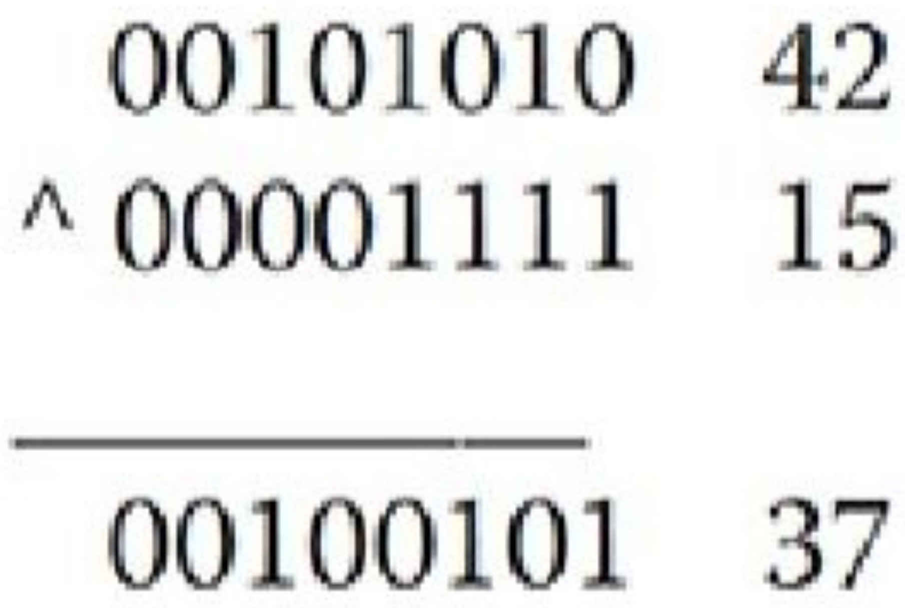
```
 00101010   42
&00001111   15
_________
 00001010   10
```

## Bitwise OR |



```
  00101010   42
| 00001111   15
  ________
  00101111   47
```

# Bitwise XOR ^

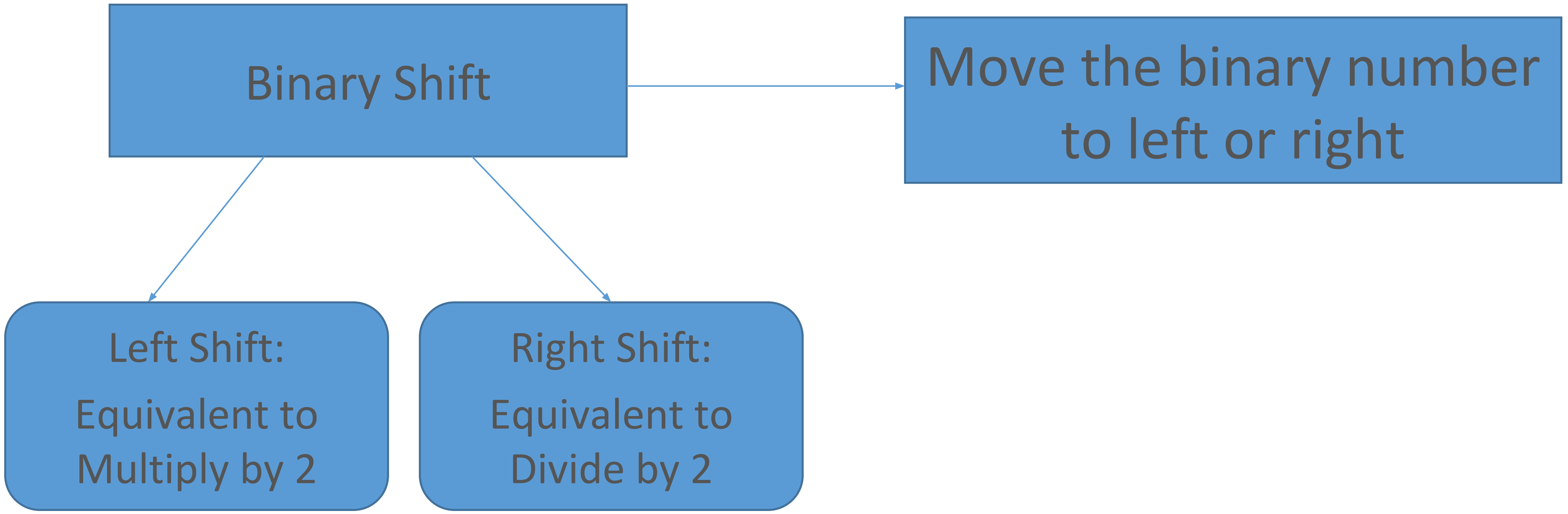


```
  00101010   42
^ 00001111   15
__________
  00100101   37
```

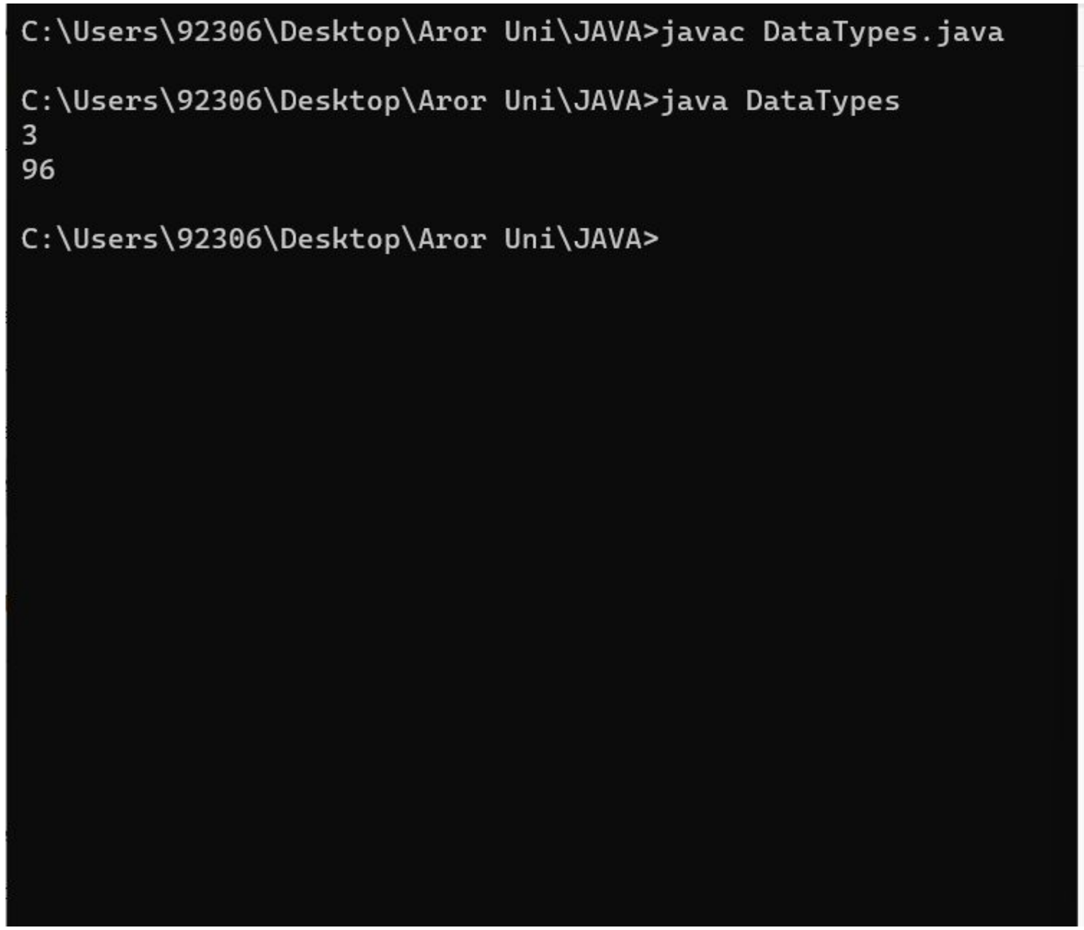
```
    a=0011
    b=0110
  ----------
a|b=0111

    a=0011
    b=0110
  ----------
a&b=0010
```

```
    a=0011
    b=0110
  ----------
a^b=0101

    a=0011
    b=0110
  ----------
a&b=0010
```

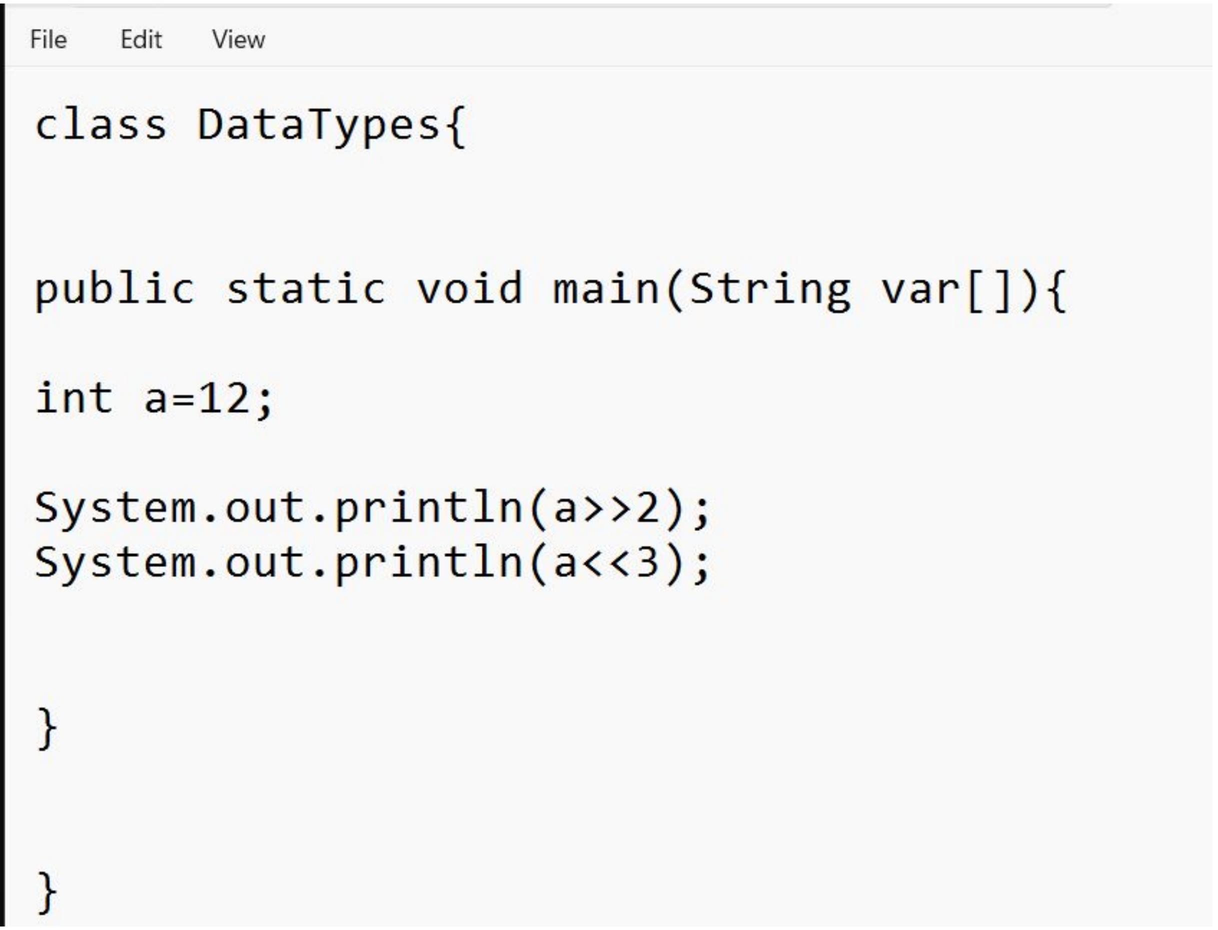
# LOGICAL BINARY SHIFTS


Binary Shift
Move the binary number to left or right
Left Shift: Equivalent to Multiply by 2
Right Shift: Equivalent to Divide by 2

## Left Shift and Right Shift Demo



```
C:\Users\92306\Desktop\Aror Uni\JAVA>javac DataTypes.java

C:\Users\92306\Desktop\Aror Uni\JAVA>java DataTypes
3
96

C:\Users\92306\Desktop\Aror Uni\JAVA>
```



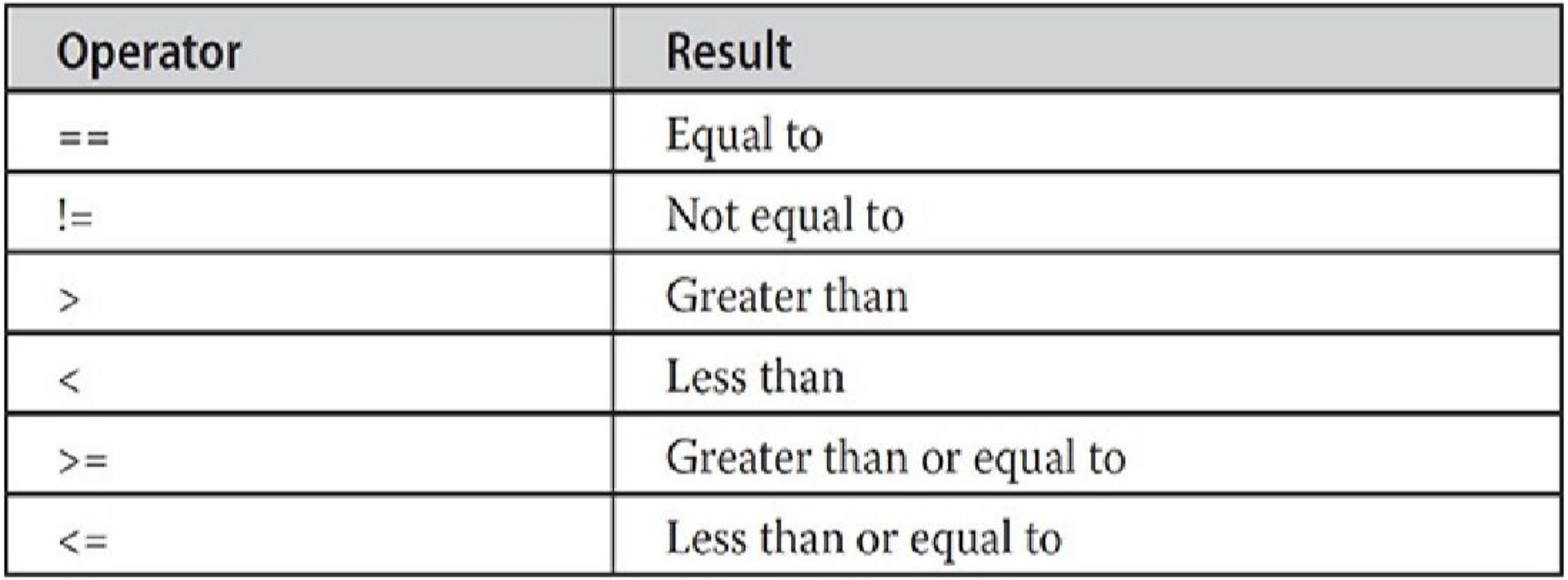
```
class DataTypes{


public static void main(String var[]){

int a=12;

System.out.println(a>>2);
System.out.println(a<<3);


}


}
```

## Bitwise Operator Compound Operator

```
a = a >> 4;
a >>= 4;
```

```
a = a | b;
a |= b;
```

## Relational Operators (Boolean Outcome)



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

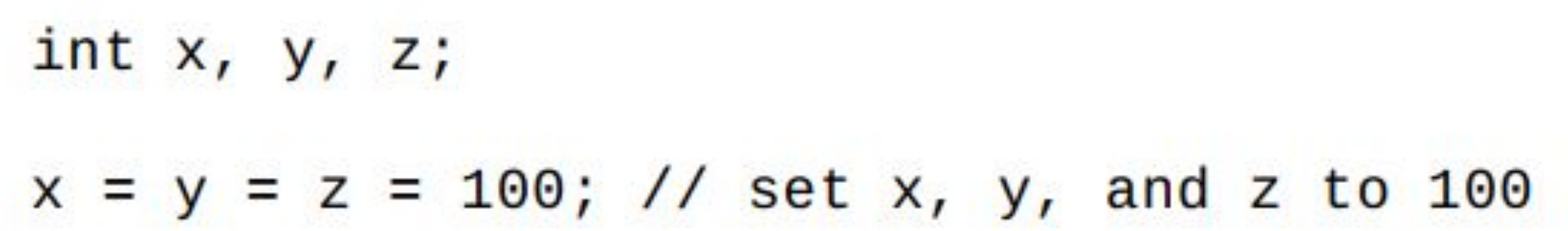
## Boolean Logical Operators

1. AND  (&&)
2. OR (||)
3. NOT (!)
4. Equal to (==)
5. Not Equal to (!=)
6. Ternary if-then-else (?:)

## Boolean Logical Operators

- Ternary Operator (?:)

```
int number = 10;
String result = (number % 2 == 0) ? "Even" : "Odd";
System.out.println("Number is: " + result);
```



```
int x, y, z;

x = y = z = 100; // set x, y, and z to 100
```

- = Operator

## Assignment Operator

## Task

- Input salary
- Use Ternary Operator to check if the salary is above 70000 output managerial level, otherwise output staff level
- You will only use conditional ternary operator