Operators (Chapter 3 of Schilit)

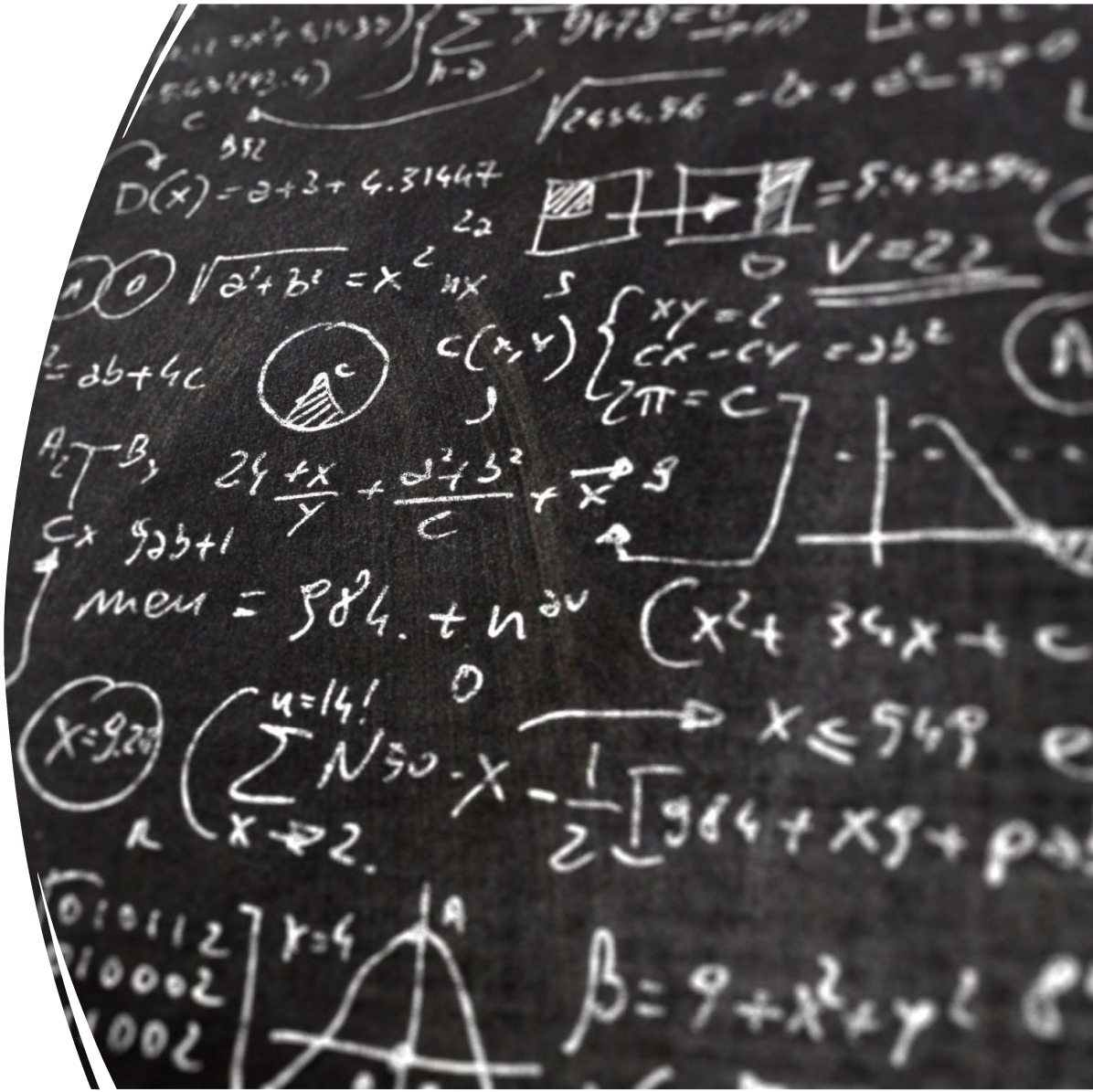
Object Oriented Programming BS (AI) II

Compiled By

Abdul Haseeb Shaikh

Operators

- Arithmetic
- Bitwise
- Relational
- Logical



sk1

Arithmetic operators

- Operands to these
- operators must be
- numeric

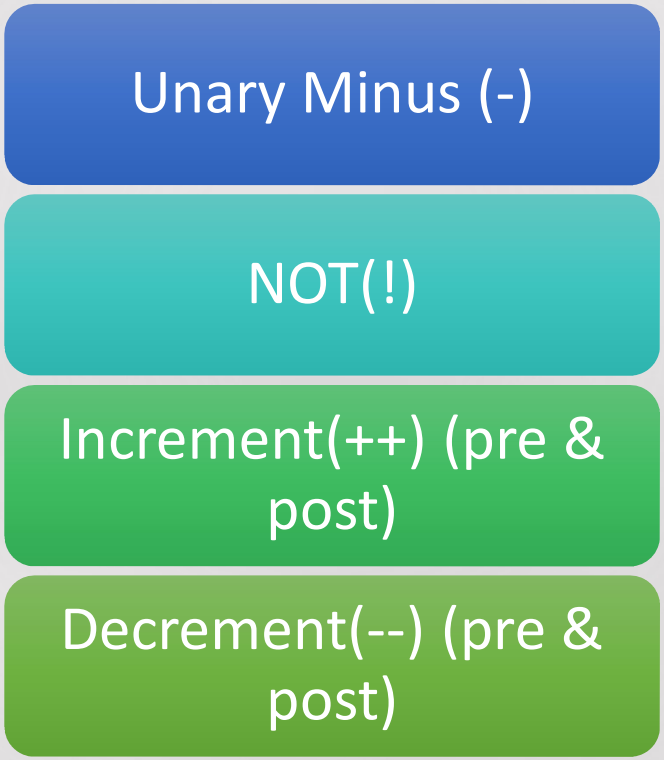
Operator	Result
+	Addition (also unary plus)
–	Subtraction (also unary minus)
*	Multiplication
/	Division
%	Modulus
++	Increment
+=	Addition assignment
– =	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment
– –	Decrement

Slide 3

sk1 sher khalil, 25/03/2021

Example (arithmetic with int and double)

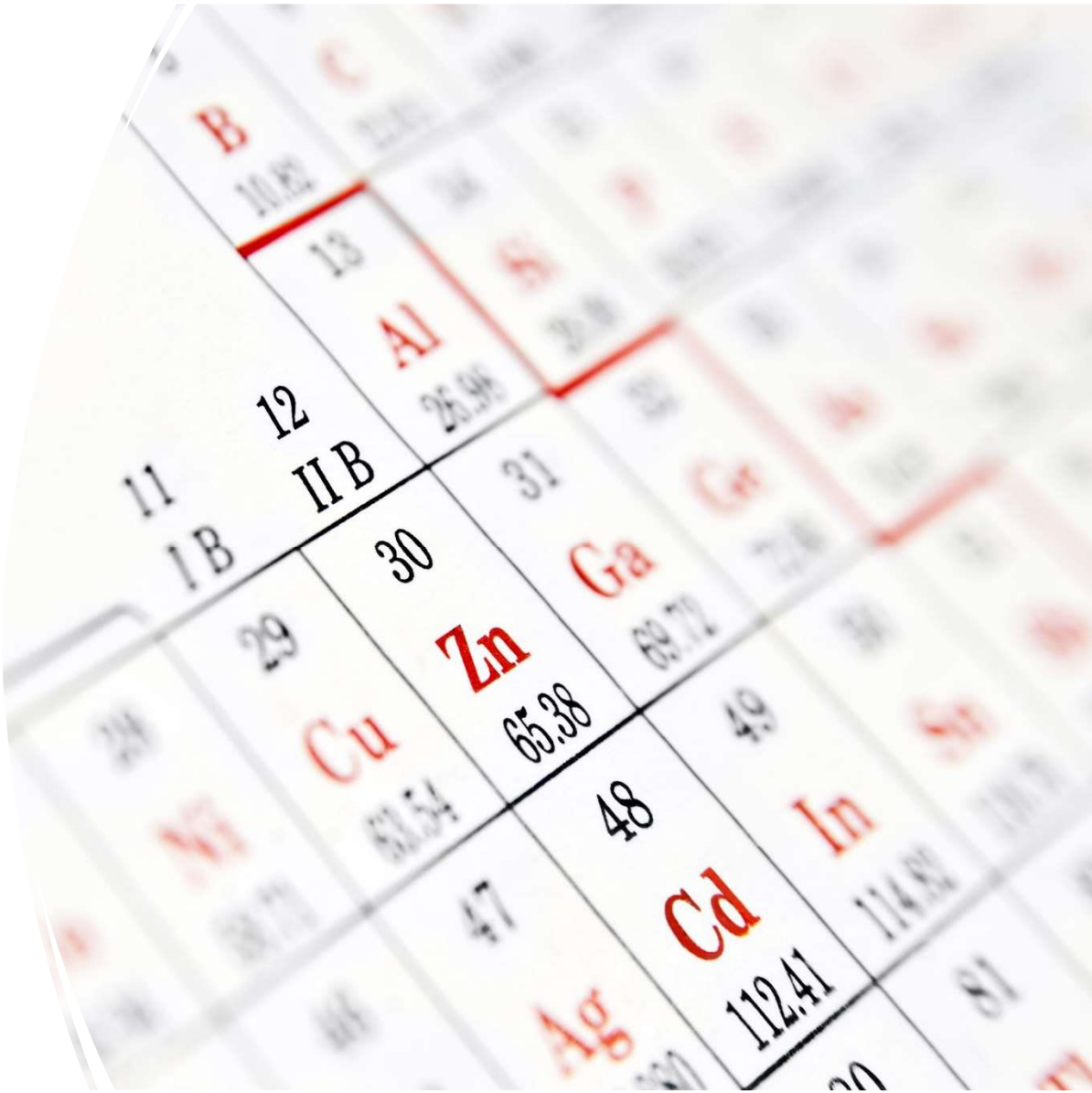
Unary Operator



Modulus Operator(%)

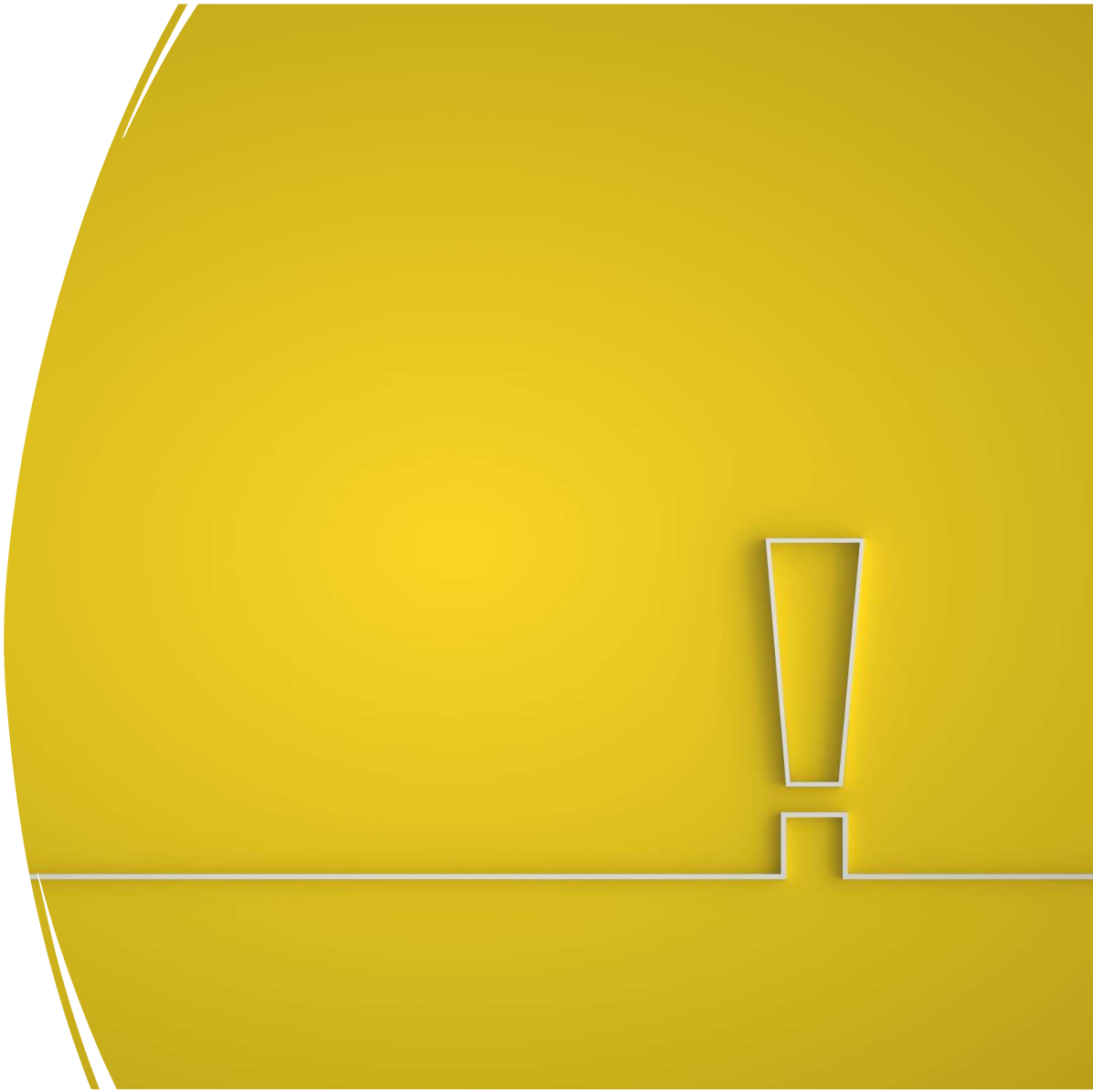
- Floating
- Integer
- What happens when left side is smaller than right side?

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>;



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

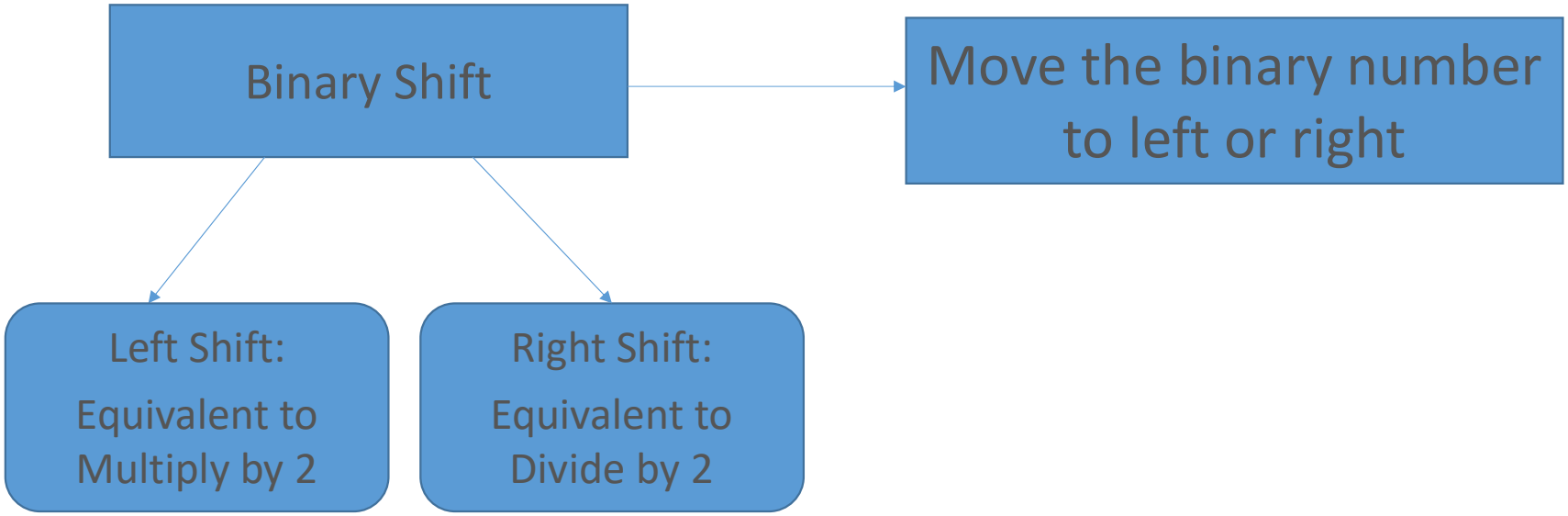
a=0011

b=0110

---

a&b=0010

LOGICAL BINARY SHIFTS



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>
```

File Edit View

```
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

Operator	Result
&	Logical AND
|	Logical OR
^	Logical XOR (exclusive OR)
||	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to
?:	Ternary if-then-else

Boolean Logical Operators

```
// Demonstrate the boolean logical operators.
class BoolLogic {
  public static void main(String args[]) {
    boolean a = true;
    boolean b = false;
    boolean c = a | b;
    boolean d = a & b;
    boolean e = a ^ b;
    boolean f = (!a & b) | (a & !b);
    boolean g = !a;
    System.out.println("        a = " + a);
    System.out.println("        b = " + b);
    System.out.println("      a|b = " + c);
    System.out.println("      a&b = " + d);
    System.out.println("      a^b = " + e);
    System.out.println("!a&b|a&!b = " + f);
    System.out.println("       !a = " + g);
  }
}
```

```
        a = true
        b = false
      a|b = true
      a&b = false
      a^b = true
!a&b|a&!b = true
       !a = false
```

Short Circuit Logical Operator

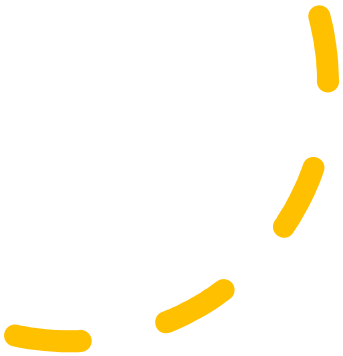
```
if( denom !=0 && num / denom > 10 )
```

Assignment Operator

```
int x, y, z;

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

- = Operator



The ? Operator

- Also known as ternary(three way) operator
- expression1?expression2:expression3

```
// Demonstrate ?.
class Ternary {
  public static void main(String args[]) {
    int i, k;

    i = 10;
    k = i < 0 ? -i : i; // get absolute value of i
    System.out.print("Absolute value of ");
    System.out.println(i + " is " + k);

    i = -10;
    k = i < 0 ? -i : i; // get absolute value of i
    System.out.print("Absolute value of ");
    System.out.println(i + " is " + k);
  }
}
```

The ? 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

Highest						
++ (postfix)	– – (postfix)					
++ (prefix)	– – (prefix)	~	!	+ (unary)	– (unary)	(*type-cast*)
*	/	%				
+	–					
>>	>>>	<<				
>	>=	<	<=	instanceof		
==	!=					
&						
^						
|						
&&						
||						
?:						
–>						
=	op=					
Lowest						

Operator Precedence

Use of parentheses

```
a >> b + 3
```



```
a >> (b + 3)
```

```
(a >> b) + 3
```