

---

# OBJECT ORIENTED PROGRAMMING USING JAVA



# OUTLINE

- Data Type
- Variable
- Variable Name
- Operators
- Precedence of Operators
- Examples of Operators

---

## DATA TYPE

- There are mainly two types of data type:
  - ❖ **Primitive:** Byte, short, int, long, float, double, boolean and char.
  - ❖ **Non-primitive:** String, arrays and classes

## DATA TYPE (CONT.)

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

## VARIABLE

- ❑ A variables can be considered as a name given to the location in memory where values are stored.\
- ❖ **<datatype> <variable\_name>**
- ❑ Can the value of variable be changed?
- ❖ Yes

## VARIABLE NAME

- ❑ Use only the characters 'a' through 'z', 'A' through 'Z', '0' through '9', character '\_', and character '\$'.
- ❑ A name can't contain space character.
- ❑ Do not start with a digit. Variable name can be of any length.
- ❑ Case sensitive.
- ❑ A name can not be a reserved word (A reserved word is a word which has a predefined meaning in Java. Example: int, double, true, etc.).

---

## VARIABLE NAME (CONT...)

❑ Can you answer that whether the below lines are correct or not?

1. `int good-bye;`
2. `int shrift = 0;`
3. `char thisMustBeTooLong;`
4. `int bubble = 0, toil = 9, trouble = 8`
5. `int 8ball;`
6. `int double;`

---

## VARIABLE NAME (CONT...)

### ❑ Answers of the last slide:

1. `int good-bye; //bad variable name`
2. `int shrift = 0; //OK`
3. `char thisMustBeTooLong; //OK in syntax //but poor choice in variable name`
4. `int bubble = 0, toil = 9, trouble = 8 // “;” missing at the end`
5. `int 8ball; //can't start with a digit`
6. `int double; //double is a reserve word`



# OPERATORS

- ❑ There are many operators in Java::
  - ❖ Unary operators
  - ❖ Arithmetic operators
  - ❖ Relational operators
  - ❖ Bitwise operators
  - ❖ Logical operators
  - ❖ Assignment operators
  - ❖ Ternary operators

## OPERATORS (CONT....)

### □ Unary operators:

Operator	Description
+	Unary plus operator; indicates positive value (numbers are positive without this, however)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean



## OPERATORS (CONT....)

### □ Arithmetic operators:

Operator	Description
+ (Addition)	Adds values on either side of the operator.
- (Subtraction)	Subtracts right-hand operand from left-hand operand.
* (Multiplication)	Multiplies values on either side of the operator.
/ (Division)	Divides left-hand operand by right-hand operand.
% (Modulus)	Divides left-hand operand by right-hand operand and returns remainder.

## OPERATORS (CONT....)

### □ Relational operators:

Operator	Description
== (equal to)	Checks if the values of two operands are equal or not, if yes then condition becomes true.
!= (not equal to)	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
> (greater than)	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
< (less than)	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
>= (greater than or equal to)	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
<= (less than or equal to)	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

## OPERATORS (CONT....)

### □ Bitwise operators:

Operator	Description
& (bitwise and)	Binary AND Operator copies a bit to the result if it exists in both operands.
(bitwise or)	Binary OR Operator copies a bit if it exists in either operand.
^ (bitwise XOR)	Binary XOR Operator copies the bit if it is set in one operand but not both.
~ (bitwise compliment)	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.
<< (left shift)	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.
>> (right shift)	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.
>>> (zero fill right shift)	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.



## OPERATORS (CONT....)

### □ Logical operators:

Operator	Description
&& (logical and)	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.
(logical or)	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.
! (logical not)	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.



## OPERATORS (CONT....)

### □ Assignment operators:

Operator	Description
=	Simple assignment operator. Assigns values from right side operands to left side operand.
+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand.
-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand.
*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand.
/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand.
%=	Modulus AND assignment operator. It takes modulus using two operands and assign the result to left operand.
<<=	Left shift AND assignment operator.
>>=	Right shift AND assignment operator.
&=	Bitwise AND assignment operator.
^=	bitwise exclusive OR and assignment operator.
=	bitwise inclusive OR and assignment operator.

---

## OPERATORS (CONT....)

- ❑ Ternary operators:

- ❖ Conditional Operator ( ? : )

# PRECEDENCE OF THE OPERATORS

Category	Operator	Associativity
Postfix	expression++ expression--	Left to right
Unary	++expression --expression +expression -expression ~ !	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >> >>>	Left to right
Relational	< > <= >= instanceof	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= ^=  = <<= >>= >>>=	Right to left

# EXAMPLES OF OPERATORS

```
class Operatorexample
{
    public static void main(String args[])
    {
        int x=10;
        System.out.println(x++);
        System.out.println(++x);
        System.out.println(x--);
        System.out.println(--x);
    }
}
```

# EXAMPLES OF OPERATORS

```
class Operatorexample
{
    public static void main(String args[])
    {
        int x=10;
        System.out.println(x++);
        System.out.println(++x);
        System.out.println(x--);
        System.out.println(--x);
    }
}
```

Output:

❖ 10

❖ 12

❖ 12

❖ 10

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample1
{
    public static void main(String args[])
    {
        System.out.println(10*10/5+3-1*4/2);
    }
}
```

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample1
{
    public static void main(String args[])
    {
        System.out.println(10*10/5+3-1*4/2);
    }
}
```

Output:  
❖21

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample2
{
    public static void main(String args[])
    {
        System.out.println(10<<2);
        System.out.println(20>>2);
    }
}
```



## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample2
{
    public static void main(String args[])
    {
        System.out.println(10<<2);
        System.out.println(20>>2);
    }
}
```

Output:

❖40

❖5

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample3
{
    public static void main(String args[])
    {
        int a=10;
        int b=5;
        int c=20;
        System.out.println(a<b&&a<c);|
        System.out.println(a<b&a<c);
    }
}
```

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample3
{
    public static void main(String args[])
    {
        int a=10;
        int b=5;
        int c=20;
        System.out.println(a<b&&a<c);|
        System.out.println(a<b&a<c);
    }
}
```

Output:

❖false

❖false

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample4
{
    public static void main(String args[])
    {
        int a=3;
        int b=6;
        int c=(a<b)?a:b;
        System.out.println(c);
    }
}
```

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample4
{
    public static void main(String args[])
    {
        int a=3;
        int b=6;
        int c=(a<b)?a:b;
        System.out.println(c);
    }
}
```

Output:  
❖3

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample5
{
    public static void main(String args[])
    {

        int a=12;
        a+=4;
        System.out.println(a);
        a-=3;
        System.out.println(a);
        a*=4;
        System.out.println(a);
        a/=5;
        System.out.println(a);
    }
}
```

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample5
{
    public static void main(String args[])
    {
        int a=12;
        a+=4;
        System.out.println(a);
        a-=3;
        System.out.println(a);
        a*=4;
        System.out.println(a);
        a/=5;
        System.out.println(a);
    }
}
```

Output:

❖ 16

❖ 13

❖ 52

❖ 10

## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample6
{
    public static void main(String args[])
    {
        int a=10;
        int b=20;
        System.out.println("a == b =" + (a == b));
        System.out.println("a != b =" + (a != b));
        System.out.println("a > b =" + (a > b));
        System.out.println("a < b =" + (a < b));
        System.out.println("a >= b =" + (a >= b));
        System.out.println("a <= b =" + (a <= b));
    }
}
```



## EXAMPLES OF OPERATORS (CONT....)

```
class Operatorexample6
{
    public static void main(String args[])
    {
        int a=10;
        int b=20;
        System.out.println("a == b =" + (a == b));
        System.out.println("a != b =" + (a != b));
        System.out.println("a > b =" + (a > b));
        System.out.println("a < b =" + (a < b));
        System.out.println("a >= b =" + (a >= b));
        System.out.println("a <= b =" + (a <= b));
    }
}
```

Output:

- ❖ a == b = false
- ❖ a != b = true
- ❖ a > b = false
- ❖ a < b = true
- ❖ a >= b = false
- ❖ a <= b = true



THANK YOU  
?