WEEK 3 : OPERATORS, EXPRESSIONS, STATEMENTS

data types :

Widening
narrowing

- byte
- short
- char
- int
- long
- float
- double
- boolean

STATEMENTS

* JAVA execute one statement after another in the order they are written

- A statement, is a complete instruction that ends with a semi-colon (;)

```
String sauces = "List of available sauces for selection: "; // assignment statement
```

- A block, is a group of statements enclosed in braces (curly brackets)

```
{
        String sauces = "List of available sauces for selection:";
        sauces += "\n" + "Chilli";
        sauces += "\n" + "Tomato";
        sauces += "\n" + "Mayo";
        sauces += "\n" + "Mustard";
        System.out.println(sauces);
}
```

- Type of statements in JAVA :

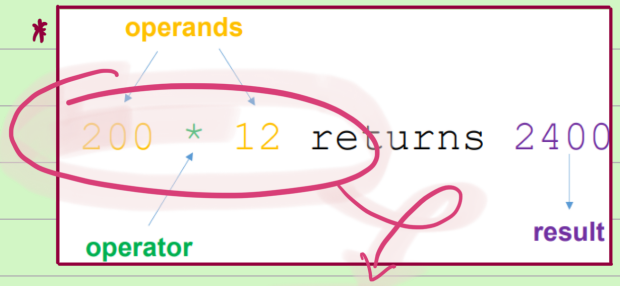
```
;                          } empty statement
double price ;             } declaration statement
price = 51.20 ;            } expression statement
if ( price > 50) {
    system.out.println("Exceeds budget") ;
}
else {
    system.out.println("Within budget") ;
}
```

} control flow statement

OPERATORS

- special symbols
- perform specific operations on operands
- then return a result
- Types of operators in JAVA :
  - Arithmetic operators
  - Comparison operators
  - Logical operators
  - Assignment operators
  - Bitwise operators

*



EXPRESSION

- legal combination of variables, constants, literals, method calls & operators
- evaluate to a single value
- used to compute & assign values to variables

Arithmetic operators :

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

* RHS MUST BE FULLY EVALUATED FIRST

BEFORE THE VALUE CAN BE STORED AT LHS TARGET MEMORY LOCATION

* java.lang.Math class contains methods for performing basic numeric operations
  ↳ eg. exponential, logarithm, square root, trigonometric functions

Assignment operators :

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

* Rational operators & String :
  - compare content of String in JAVA, use equals()
  - == (identity address), .equals (equality)

```
int num1 = 11;
int num2 = 2;

boolean result = (num1>10 && (num1++ > num2));
System.out.println("result = " + result);
System.out.println("num1 = " + num1);

num1 = 8;
result = (num1>10 && (num1++ > num2));
System.out.println("result = " + result);
System.out.println("num1 = " + num1);
```

SHORT-CIRCUIT OPERATORS

Comparison operators :

- compare 2 values
- result is a boolean value

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical operators :

Operator	Name	Description	Example
&&	Logical and ("AND GATE")	Returns true if both statements are true	x < 5 && x < 10
||	Logical or ("OR GATE")	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

- used to check whether an expression is TRUE or FALSE
- combine with rational operators to make MORE COMPLICATED BOOLEAN EXPRESSIONS
- result is a BOOLEAN value

Bitwise Operators :

Operators	Symbol	Uses
Bitwise AND	&	op1 & op2
Bitwise exclusive OR	^	op1 ^ op2
Bitwise inclusive OR	|	op1 | op2
Bitwise Compliment	~	~ op
Bitwise left shift	<<	op1 << op2
Bitwise right shift	>>	op1 >> op2
Unsigned Right Shift Operator	>>> op >>>	number of places to shift

The binary number system (base 2) uses 0s and 1s to represent numbers. Computers use binary numbers to store and perform operations on any data.

EXAMPLE :

```
int a = 11;
int b = 9;

System.out.println("bitwise and (a&b) = " + (a&b));
System.out.println("bitwise and (a|b) = " + (a|b));
System.out.println("bitwise exclusive (a^b) = " +
(a^b));
```

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12

* BITWISE SHIFT OPERATORS

The bitwise shift operators move the bit values of a binary object.

Operator	Usage
<<	Indicates the bits are to be shifted to the left.
>>	Indicates the bits are to be shifted to the right.

```
int x = 2; // 0010

System.out.println("x<<1 = " + (x<<1));
System.out.println("x<<2 = " + (x<<2));

System.out.println("x>>1 = " + (x>>1));
System.out.println("x>>2 = " + (x>>2));
```

先乘除后加减 "FIRST X & ÷ THEN ONLY + & - "

Operator Precedence

- BODMAS - Brackets, Orders, Multiplication, Addition and Subtraction
- What would be the output of the following program?
  - Divide yourself into 2 groups, one group calculates by head, the other group writes code. Which will be quicker?

```
int a = 7 * 3 + 24 / 3 - 2;
int b = (7 * 3) + (24 / 3) - 2;
System.out.println("a = " + a + " ; b = " + a);
```

```
double myCGPA = 2 + 9 / 3.0;
System.out.println("myCGPA = " + myCGPA);
```

Output?
5.0 or 3.67