3. Java Language Structure

Data types

Operators

Control Statements

Operators

- Java has a rich collection of operators and are classified into mainly 4 groups:
  - Arithmetic Operators
  - Bitwise Operators
  - Relational Operators
  - Logical Operators

Arithmetic Operators – ‘Binary’

Operator	Use	Description
+	op1+ op2	Adds op1 and op2
-	op1 - op2	Subtracts op2 from op1
*	op1* op2	Multiplies op1 by op2
/	op1 / op2	Divides op1 by op2
%	op1%op2	Computes the remainder

```
class ArithmeticTest {
    public static void main (String[] args) {
            short x = 6;
            int y = 4;
            float a = 12.5f;
            float b = 7.0f;
            System.out.println(“x is “ + x + “, y is “ + y);
            System.out.println(“x + y = “ + (x + y));
            System.out.println(“x - y = “ + (x - y));
            System.out.println(“x / y = “ + (x / y));
            System.out.println(“x % y = “ + (x % y));
            System.out.println(“a is “ + a + “, b is “+ b);
            System.out.println(“a / b = “ + (a / b));
    }
}
```

Result:

```
x is 6, y is 4
x + y = 10
x - y = 2
x / y = 1
x % y = 2
a is 12.5, b is 7.0
a / b = 1.7857143
```

Arithmetic Operators – ‘Unary’

Operator	**Use**	**Meaning**
++ (increment)	op**++**	op = op + 1
- - (decrement)	op**- -**	op = op - 1

Arithmetic Operators – ‘Unary’

Operator	**Use**	**Description**
++	op++	Post-increment
++	++op	Pre-increment
– –	op– –	Post-decrement
– –	– –op	Pre-decrement

Examples:

```
x = 42;          x = x + 1;
y = ++x;    →    y = x;        →    y = 43

x = 42;          y = x;
y = x++;    →    x = x + 1;    →    y = 42
```

```
int x = 0; int y = 0;
            System.out.println(“x and y are “ + x + “ and “ + y );
 x++;
            System.out.println(“x++ results in “ + x);

++x;
            System.out.println(“++x results in “ + x);
            System.out.println(“Resetting x back to 0.”);
 x = 0;
y = x++;
            System.out.println(“y = x++ (postfix) results in:”);
            System.out.println(“x is “ + x);
            System.out.println(“y is “ + y);
y = ++x;
            System.out.println(“y = ++x (prefix) results in:”);
             System.out.println(“x is “ + x);
            System.out.println(“y is “ + y);
```

Exercise

Consider the following code snippet:

```
int i = 10;
int n = i++%5;
```

Question: What are the values of i and n after the code is executed?

Question: What are the final values of i and n if instead of using the postfix increment operator (i++), you use the prefix version (++i))?

Exercise

```
int i = 3;
i++;
System.out.println(i);
++i;
System.out.println(i);
System.out.println(++i);
System.out.println(i++);
System.out.println(i);
```

Exercise

```
int a = 26, b = 37, c, d;
a = ++b;
c = ++a;
b = c++;
d = b++;
c = ++d;
System.out.println("a: " + a);
System.out.println("b: " + b);
System.out.println("c: " + c);
System.out.println("d: " + d);
```

Bitwise Operators

Operator	Use	Operation
&	op1 & op2	Bitwise AND
|	op1 | op2	Bitwise OR
^	op1 ^ op2	Bitwise XOR
~	~op2	Bitwise NOT (unary)

Bitwise Operators

These operators operate on all integer types like **long**, **int**, **short**, **byte** and **char**.

op1	op2	Result
0	0	0
0	1	0
1	0	0
1	1	1

AND

op1	op2	Result
0	0	0
0	1	1
1	0	1
1	1	1

OR

op1	op2	Result
0	0	0
0	1	1
1	0	1
1	1	0

XOR

Shift operators

Operator	Use	Operation
>>	op1 >> op2	shift bits of op1 right by distance op2
<<	op1 << op2	shift bits of op1 left by distance op2

Relational Operators

Returns Boolean value (true / false)

Operator	**Use**	**Result**
>	op1 > op2	greater than
>=	op1 >= op2	greater than or equal to
<	op1 < op2	less than
<=	op1 <= op2	less than or equal to
==	op1 == op2	equal to
!=	op1 != op2	not equal to

Logical Operators

Operates only on boolean operands and return boolean value.

Operator	Use	Returns true if
&	op1 & op2	op1 and op2 are both true, always evaluates op1 and op2
|	op1 | op2	either op1 or op2 is true, always evaluates op1 and op2
^	op1 ^ op2	if op1 and op2 are different - that is if one or the other of the operands is true but not both

Logical Operators

Operator	**Use**	**Returns true if**
&&	op1 **&&** op2	op1 and op2 are both true, conditionally evaluates op2
||	op1 **||** op2	either op1 or op2 is true, conditionally evaluates op2
!	**!** op	op is false

Assignment Operators

Operator	**Use**	**Equivalent to**
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2
&=	op1 &= op2	op1 = op1 & op2
|=	op1 |= op2	op1 = op1 | op2
^=	op1 ^= op2	op1 = op1 ^ op2

Assignment Operators

Operator	Use	Equivalent to
<<=	op1 <<= op2	op1 = op1 << op2
>>=	op1 >>= op2	op1 = op1 >> op2

Other Operators

Operator	Description
`? :`	Ternary operator
`[]`	Used to in arrays
`.`	Used to form qualified names
`(` *params* `)`	Comma-separated list of parameters
`(` *type* `)`	Casts a value to the specified type
new	Creates a new object or a new array

Ternary operator

- This is equivalent to if-then-else statements.
- General form:

```
exprn1 ? exprn2 : exprn3;
```

Example:

```
ratio = denom == 0 ? 0 : num/denom;
```