**Common statements in Java**

Common statements in Java include assignment statement (Sequence Structure), conditional statement (selection structure), loop statement, and switch statement, etc. These statements are basic grammar that composes Java, thus a must-learn for programmers. Now we will learn about these statements step by step:

**Assignment statement (Sequence Structure)**

Assignment statements are the most commonly used statements. As this part has been explained in the previous class, we won't talk about it here.

**Conditional statements**

Conditional statement is a set of rules performed in programme if a certain condition is met. The difficulties in using conditional statements are to abstract its conditions accurately. For example, to implement an application log-in function, its conditional statement should be: if the account name and password are correctly input, the user can enter the system; otherwise the system will pop up the box indicating "Password is wrong". This part will focus on grammar and basic usage.

**If statements**

*Grammar:*

**if** (**boolean** expression) {

The statements to be executed when **boolean** expressions are **true**

}

* If the values of the boolean expressions are true, please execute the code blocks in the if statement; otherwise please execute the codes following the if statement.
* We strongly recommend you to add {} after if. Though it is acceptable not writing {}, omitting this will reduce the codes' readability and maintainability.

*For example:*

**int** num = 1;

**if** (num < 2) {

System.out.print("smaller than 2");

}

* Output: smaller than 2

**if...else statements**

*Grammar:*

**if** (**boolean** expression) {

The statements to be executed when **boolean** expressions are **true**

} **else** {

The statements to be executed when **boolean** expressions are **false**

}

* If the values of the boolean expressions are true, please execute the code blocks in the if statement; otherwise please execute the codes following the else statement block.
* We strongly recommend you to add {} after if and else. Though it is acceptable not writing {}, omitting this will reduce the codes' readability and maintainability.

*For example:*

**int** num = 3;

**if** (num < 2) {

System.out.print("smaller than 2");

} **else** {

System.out.print("bigger or equivalent to 2")；

}

* Output: bigger or equivalent to 2

**if...else if...else...statement**

*Grammar:*

**if** (**boolean** expression 1) {

The statements to be executed when **boolean** expression 1 is **true**

} **else** **if** (**boolean** expression 2) {

The statements to be executed when **boolean** expression 2 is **true**

} **else** {

The statements to be executed when all above **boolean** expression are untrue

}

* Each if statement has at most one else statement. Else statements should be after elseif statements.
* Each if statement can have multiple elseif statements. But elseif statements must be before an else statement.
* When one of the else if statements is true, other else if and else statements won't be executed.
* We strongly recommend you to add {} after if, else if and else. Though it is acceptable not writing {}, omitting this will reduce the codes' readability and maintainability.

*For example:*

**int** num = 2;

**if** (num < 2) {

System.out.print("smaller than 2");

} **else** **if** (num == 2) {

System.out.print("equivalent to 2");

} **else** {

System.out.print("bigger than 2");

}

* Output: equivalent to 2

**Switch statements**

Switch statements use switch cases to decide whether a variable is equivalent to a certain variable in a series of values, of which each value is called a switch.

*Grammar:*

**switch** (expression) {

**case** value1:

statement

**break**;

**case** value2:

statement

**break**;

...

**default** :

statement

}

* The variable types used in switch statement include: byt, short, int, char and [enum](https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html). Since Java SE 7, switch statement began to support character strings of the String type. Meanwhile the case labels should be character string constants or literals.
* Switch statements may have multiple case statements, with each case followed by a value or colon to be compared with.
* The data type of the values in a case statement should be the same with that of the variable, which should be no other than constants or literal constants.
* When the value of a variable is the same with that in a case statement, the statement after the case statement will start to be executed. Until you come across a break statement does a switch statement pop up.
* When a break statement appears, the switch statement comes to an end. The programme will then execute the statement after the switch statement. If no break statement on the way, the programme will continue to execute the next case statement until coming to a break statement.
* A switch statement can have an optional default case, which is better to be at the end of the switch (there is no fixed position, but it is recommended to be the last). The default case can be used for performing a task when none of the cases is true. The default case doesn't need break statement.
* Though optional, default case is strongly recommended to be included in each switch statement.

*For example:*

**char** grade = 'C';

**switch**(grade)

{

**case** 'A' :

System.out.println("Excellent");

**break**;

**case** 'B' :

**case** 'C' :

System.out.println("Good");

**break**;

**case** 'D' :

System.out.println("Pass");

**case** 'F' :

System.out.println("You should work harder");

**break**;

**default** :

System.out.println("Unknown grade");

}

* Output: Good

**Loop statements**

A loop statement allows us to execute a statement or group of statements multiple times. Compared to humans, computers are much better at looping. There are three types of loops often used in Java:

**For loop**

*Grammar:*

**for** (INIT statement; **boolean** expression; update) {

*//loop body*

}

Execution steps：

1. The first step is to execute initialization process. You can only declare one type, but can intialize one or multiple loop control variables. It can also be null statement.
2. The second step is to test the value of the boolean expression. If it's true, the loop body will be executed. If false, the loop ends here and start to execute the statements after the loop body.
3. The third step is to execute the loop for one time and then update the loop control variable.
4. The fourth step is to test again the value of the boolean expression.
5. The fifth step is to do looping execution for above process.

*For example:*

**for** (**int** num = 0; num < 10; num++) {

System.out.print(num);

}

Running result: print the integer from 0 to 9

To learn more about the loop for, pleae visit[here](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html)。

**while loop**

*Grammar:*

**while** (**boolean** expression) {

*//loop body*

}

Notes on execution procedures:

* As long as the value of the boolean expression is true, the loop will keep running.

*For example*

**int** num = 0;

**while** (num < 10) {

System.out.print(num);

num ++;

}

Running result: print the integer from 0 to 9

To learn more about the loop while, please visit[Here](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html).

**do...while loop**

*Grammar:*

**do** {

*//loop body*

} **while** (**boolean** expression)

Notes on execution procedures:

* Boolean expression is placed after loop body, so the statement block has already been executed before checking the boolean expression.
* If the value of the boolean expression is true, the statement blocks will keep running until it comes to a boolean expression whose value is false.

*For example:*

**int** num = 0;

**do** {

System.out.print(num);

num++;

} **while** (num < 10);

Running result: print the integer from 0 to 9

For more information about the while loop: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html>.

**break 和 continue**

* break：Jump out of the whole loop and execute the statement after the loop statement block.
* continue：Jump out of the current loop and execute the next loop.
* Please refer to the detailed distinction： <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/branch.html>.