Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, [Java](https://www.javatpoint.com/java-tutorial) provides statements that can be used to control the flow of Java code. Such statements are called control flow statements. It is one of the fundamental features of Java, which provides a smooth flow of program.

Java provides three types of control flow statements.

1. Decision Making statements
   * if statements
   * switch statement
2. Loop statements
   * do while loop
   * while loop
   * for loop
   * for-each loop
3. Jump statements
   * break statement
   * continue statement

Decision-Making statements:

As the name suggests, decision-making statements decide which statement to execute and when. Decision-making statements evaluate the Boolean expression and control the program flow depending upon the result of the condition provided. There are two types of decision-making statements in Java, i.e., If statement and switch statement.

1) If Statement:

In Java, the "if" statement is used to evaluate a condition. The control of the program is diverted depending upon the specific condition. The condition of the If statement gives a Boolean value, either true or false. In Java, there are four types of if-statements given below.40.5M6History of Java

1. Simple if statement
2. if-else statement
3. if-else-if ladder
4. Nested if-statement

Let's understand the if-statements one by one.

1) Simple if statement:

It is the most basic statement among all control flow statements in Java. It evaluates a Boolean expression and enables the program to enter a block of code if the expression evaluates to true.

Syntax of if statement is given below.

1. **if**(condition) {
2. statement 1; //executes when condition is true
3. }

Consider the following example in which we have used the **if** statement in the java code.

Student.java

**Student.java**

1. **public** **class** Student {
2. **public** **static** **void** main(String[] args) {
3. **int** x = 10;
4. **int** y = 12;
5. **if**(x+y > 20) {
6. System.out.println("x + y is greater than 20");
7. }
8. }
9. }

**Output:**

x + y is greater than 20

2) if-else statement

The [if-else statement](https://www.javatpoint.com/java-if-else) is an extension to the if-statement, which uses another block of code, i.e., else block. The else block is executed if the condition of the if-block is evaluated as false.

**Syntax:**

1. **if**(condition) {
2. statement 1; //executes when condition is true
3. }
4. **else**{
5. statement 2; //executes when condition is false
6. }

Consider the following example.

**Student.java**

1. **public** **class** Student {
2. **public** **static** **void** main(String[] args) {
3. **int** x = 10;
4. **int** y = 12;
5. **if**(x+y < 10) {
6. System.out.println("x + y is less than      10");
7. }   **else** {
8. System.out.println("x + y is greater than 20");
9. }
10. }
11. }

**Output:**

x + y is greater than 20

3) if-else-if ladder:

The if-else-if statement contains the if-statement followed by multiple else-if statements. In other words, we can say that it is the chain of if-else statements that create a decision tree where the program may enter in the block of code where the condition is true. We can also define an else statement at the end of the chain.

Syntax of if-else-if statement is given below.

1. **if**(condition 1) {
2. statement 1; //executes when condition 1 is true
3. }
4. **else** **if**(condition 2) {
5. statement 2; //executes when condition 2 is true
6. }
7. **else** {
8. statement 2; //executes when all the conditions are false
9. }

Consider the following example.

**Student.java**

1. **public** **class** Student {
2. **public** **static** **void** main(String[] args) {
3. String city = "Delhi";
4. **if**(city == "Meerut") {
5. System.out.println("city is meerut");
6. }**else** **if** (city == "Noida") {
7. System.out.println("city is noida");
8. }**else** **if**(city == "Agra") {
9. System.out.println("city is agra");
10. }**else** {
11. System.out.println(city);
12. }
13. }
14. }

**Output:**

Delhi

4. Nested if-statement

In nested if-statements, the if statement can contain a **if** or **if-else** statement inside another if or else-if statement.

Syntax of Nested if-statement is given below.

1. **if**(condition 1) {
2. statement 1; //executes when condition 1 is true
3. **if**(condition 2) {
4. statement 2; //executes when condition 2 is true
5. }
6. **else**{
7. statement 2; //executes when condition 2 is false
8. }
9. }

Consider the following example.

**Student.java**

1. **public** **class** Student {
2. **public** **static** **void** main(String[] args) {
3. String address = "Delhi, India";
5. **if**(address.endsWith("India")) {
6. **if**(address.contains("Meerut")) {
7. System.out.println("Your city is Meerut");
8. }**else** **if**(address.contains("Noida")) {
9. System.out.println("Your city is Noida");
10. }**else** {
11. System.out.println(address.split(",")[0]);
12. }
13. }**else** {
14. System.out.println("You are not living in India");
15. }
16. }
17. }

**Output:**

Delhi

Switch Statement:

In Java, [Switch statements](https://www.javatpoint.com/java-switch) are similar to if-else-if statements. The switch statement contains multiple blocks of code called cases and a single case is executed based on the variable which is being switched. The switch statement is easier to use instead of if-else-if statements. It also enhances the readability of the program.

Points to be noted about switch statement:

* The case variables can be int, short, byte, char, or enumeration. String type is also supported since version 7 of Java
* Cases cannot be duplicate
* Default statement is executed when any of the case doesn't match the value of expression. It is optional.
* Break statement terminates the switch block when the condition is satisfied.  
  It is optional, if not used, next case is executed.
* While using switch statements, we must notice that the case expression will be of the same type as the variable. However, it will also be a constant value.

The syntax to use the switch statement is given below.

1. **switch** (expression){
2. **case** value1:
3. statement1;
4. **break**;
5. .
6. .
7. .
8. **case** valueN:
9. statementN;
10. **break**;
11. **default**:
12. **default** statement;
13. }

Consider the following example to understand the flow of the switch statement.

**Student.java**

1. **public** **class** Student **implements** Cloneable {
2. **public** **static** **void** main(String[] args) {
3. **int** num = 2;
4. **switch** (num){
5. **case** 0:
6. System.out.println("number is 0");
7. **break**;
8. **case** 1:
9. System.out.println("number is 1");
10. **break**;
11. **default**:
12. System.out.println(num);
13. }
14. }
15. }

**Output:**

2

While using switch statements, we must notice that the case expression will be of the same type as the variable. However, it will also be a constant value. The switch permits only int, string, and Enum type variables to be used.