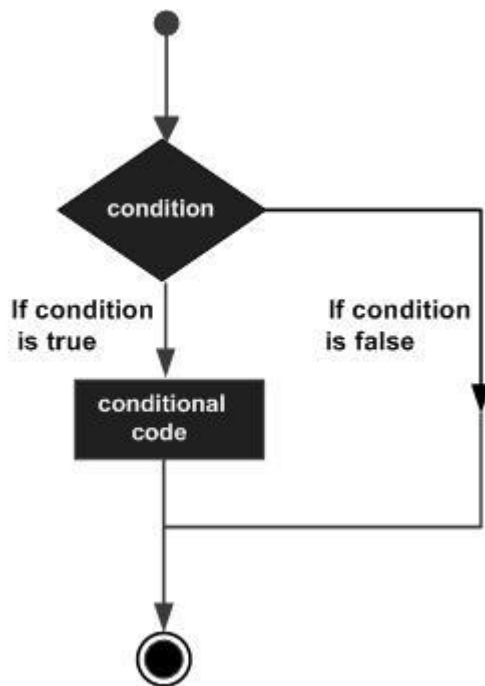


Java - Decision Making

Decision making structures have one or more conditions to be evaluated or tested by the program, along with a statement or statements that are to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following is the general form of a typical decision making structure found in most of the programming languages –



Java programming language provides following types of decision making statements. Click the following links to check their detail.

Sr.No.	Statement & Description
1	<u>if statement</u> An if statement consists of a boolean expression followed by one or more statements.
2	<u>if...else statement</u> An if statement can be followed by an optional else statement , which executes when the boolean expression is false.

3	<u>nested if statement</u> You can use one if or else if statement inside another if or else if statement(s).
4	<u>switch statement</u> A switch statement allows a variable to be tested for equality against a list of values.

if statement in java

An **if** statement consists of a Boolean expression followed by one or more statements.

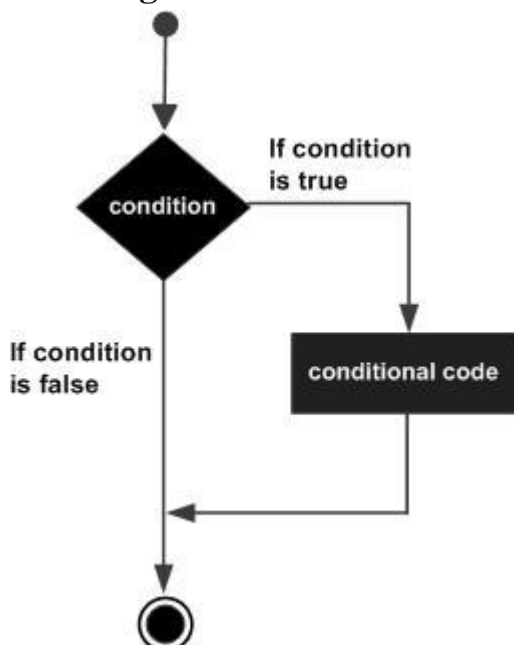
Syntax

Following is the syntax of an if statement –

```
if(Boolean_expression) {
    // Statements will execute if the Boolean expression is true
}
```

If the Boolean expression evaluates to true then the block of code inside the if statement will be executed. If not, the first set of code after the end of the if statement (after the closing curly brace) will be executed.

Flow Diagram



Example

```
public class Test {  
  
    public static void main(String args[]) {  
        int x = 10;  
  
        if( x < 20 ) {  
            System.out.print("This is if statement");  
        }  
    }  
}
```

Output

This is if statement.

if-else statement in java

An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false.

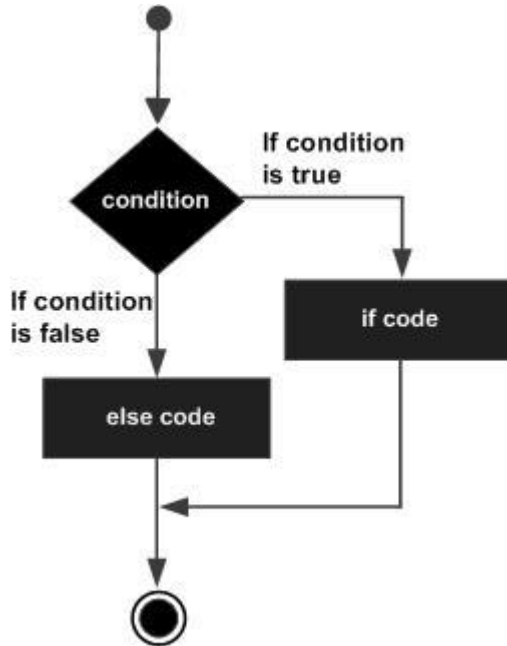
Syntax

Following is the syntax of an if...else statement –

```
if(Boolean_expression) {  
    // Executes when the Boolean expression is true  
}else {  
    // Executes when the Boolean expression is false  
}
```

If the boolean expression evaluates to true, then the if block of code will be executed, otherwise else block of code will be executed.

Flow Diagram



Example

```
public class Test {  
  
    public static void main(String args[]) {  
        int x = 30;  
  
        if( x < 20 ) {  
            System.out.print("This is if statement");  
        }else {  
            System.out.print("This is else statement");  
        }  
    }  
}
```

This will produce the following result –

Output

This is else statement

The if...else if...else Statement

An if statement can be followed by an optional *else if...else* statement, which is very useful to test various conditions using single if...else if statement.

When using if, else if, else statements there are a few points to keep in mind.

- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of the remaining else if's or else's will be tested.

Syntax

Following is the syntax of an if...else statement –

```
if(Boolean_expression 1) {  
    // Executes when the Boolean expression 1 is true  
}else if(Boolean_expression 2) {  
    // Executes when the Boolean expression 2 is true  
}else if(Boolean_expression 3) {  
    // Executes when the Boolean expression 3 is true  
}else {  
    // Executes when the none of the above condition is true.  
}
```

Example

```
public class Test {  
  
    public static void main(String args[]) {  
  
        int x = 30;  
  
        if( x == 10 ) {  
            System.out.print("Value of X is 10");  
        }else if( x == 20 ) {  
            System.out.print("Value of X is 20");  
        }else if( x == 30 ) {  
            System.out.print("Value of X is 30");  
        }  
    }  
}
```

```
}else {  
    System.out.print("This is else statement");  
}  
}  
}
```

This will produce the following result –

Output

Value of X is 30

nested if statement in java

It is always legal to nest if-else statements which means you can use one if or else if statement inside another if or else if statement.

Syntax

The syntax for a nested if...else is as follows –

```
if(Boolean_expression 1) {  
    // Executes when the Boolean expression 1 is true  
    if(Boolean_expression 2) {  
        // Executes when the Boolean expression 2 is true  
    }  
}
```

You can nest **else if...else** in the similar way as we have nested *if* statement.

Example

```
public class Test {  
  
    public static void main(String args[]) {  
        int x = 30;  
        int y = 10;
```

```
if( x == 30 ) {  
    if( y == 10 ) {  
        System.out.print("X = 30 and Y = 10");  
    }  
}  
}  
}
```

This will produce the following result –

Output

```
X = 30 and Y = 10
```

switch statement in java

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

Syntax

The syntax of enhanced for loop is –

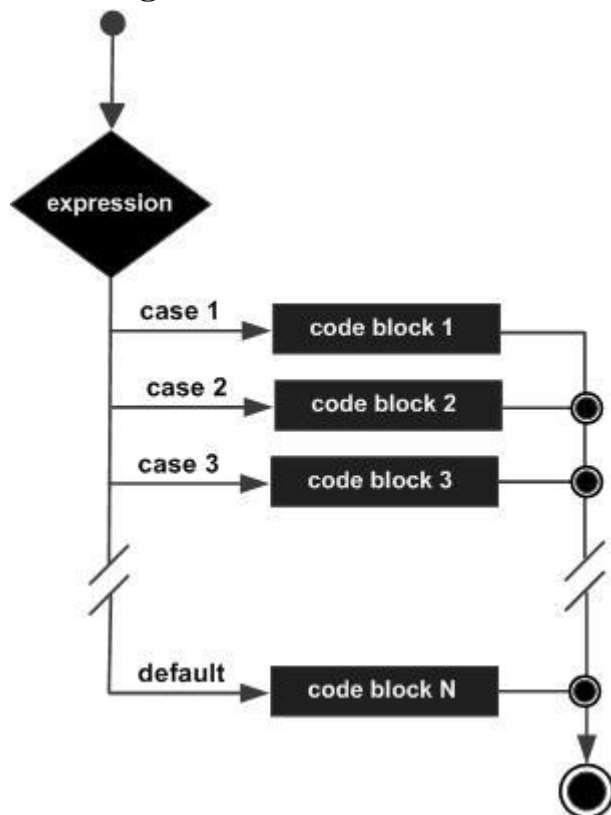
```
switch(expression) {  
    case value :  
        // Statements  
        break; // optional  
  
    case value :  
        // Statements  
        break; // optional  
  
    // You can have any number of case statements.  
    default : // Optional  
        // Statements  
}
```

The following rules apply to a **switch** statement –

- The variable used in a switch statement can only be integers, convertible integers (byte, short, char), strings and enums.

- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The value for a case must be the same data type as the variable in the switch and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a *break* statement is reached.
- When a *break* statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will *fall through* to subsequent cases until a break is reached.
- A *switch* statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

Flow Diagram



Example

```
public class Test {  
  
    public static void main(String args[]) {  
        char grade = args[0].charAt(0);  
  
        switch(grade) {  
            case 'A':  
                System.out.println("Excellent!");  
                break;  
            case 'B':  
            case 'C':  
                System.out.println("Well done");  
                break;  
            case 'D':  
                System.out.println("You passed");  
            case 'F':  
                System.out.println("Better try again");  
                break;  
            default :  
                System.out.println("Invalid grade");  
        }  
        System.out.println("Your grade is " + grade);  
    }  
}
```

Compile and run the above program using various command line arguments. This will produce the following result –

Output

```
$ java Test a
```

```
Invalid grade
```

```
Your grade is a
```

```
$ java Test A
```

```
Excellent!
```

```
Your grade is A
```

```
$ java Test C
```

```
Welldone
```

```
| Your grade is C
```