

JAVA



What you learn ?

Java

- | | |
|------------------------------|------------------------------|
| ➤ Decision Making statements | ➤ Looping statements |
| ➤ Jump statements | ➤ <code>exit()</code> method |

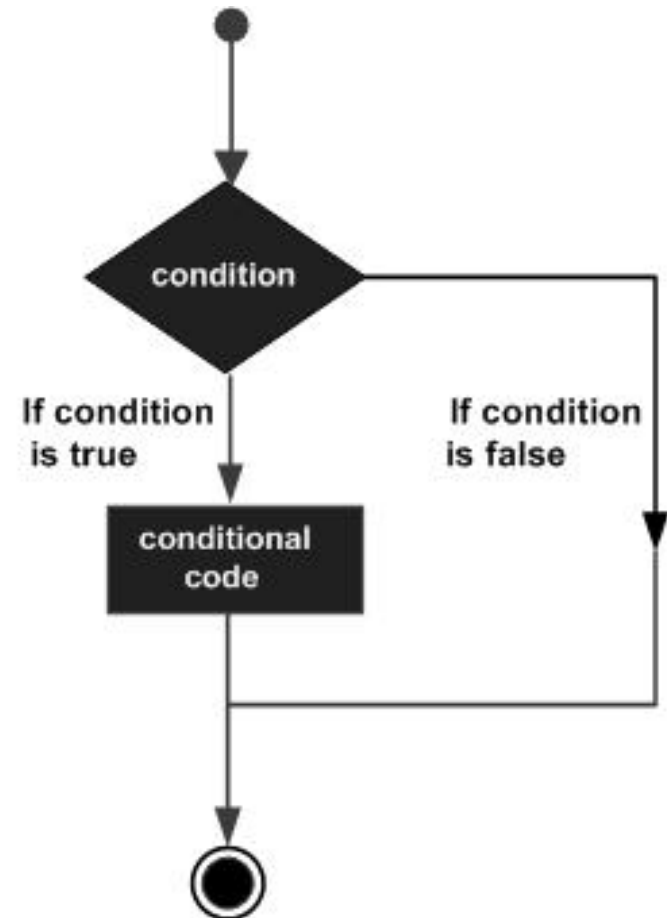
Decision Making Statements



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

–



Decision Making Statements



- **If-else statement:** The if-else statement is used when you want to execute a block of code if a certain condition is true, and a different block of code if the condition is false.
- **Switch statement:** The switch statement is used when you want to execute different blocks of code based on different possible values of a variable.
- **Ternary operator:** The ternary operator is a shorthand way of writing an if-else statement in a single line

Types of if statement

- **The if statement** is the most basic type of conditional statement in Java. It is used to test a single condition, and if that condition is true, the statements inside the if block are executed.
- **The if-else statement** is used when there are two possible outcomes based on the condition being tested. If the condition is true, the statements inside the if block are executed, otherwise, the statements inside the else block are executed.
- **The if-else-if ladder** is another type of conditional statement in Java. It's used when there are multiple possible outcomes based on the condition being tested. It consists of a series of if-else statements chained together.
- **A nested if statement** is a conditional statement that is placed inside another conditional statement. It's used when there is a need to test a more complex condition that involves multiple sub-conditions

if statement



Syntax-

```
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.

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



Syntax-

```
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.

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");  
        }  
    }  
}
```

Output:

This is else statement

if-else-if ladder



```
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");  
        }  
    }  
}
```

Output:

Value of X is 30

Nested if statement



Syntax-

```
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");  
            }  
        }  
    }  
}
```

Output:

X = 30 and Y = 10

Ternary operator



Java ternary operator is the only conditional operator that takes three operands. It's a one-liner replacement for the if-then-else statement and is used a lot in Java programming.

Syntax:

`variable = Expression1 ? Expression2: Expression3`

If operates similarly to that of the if-else statement as in Expression2 is executed if Expression1 is true else Expression3 is executed.


Example:

```
num1 = 10;
```

```
num2 = 20;
```

```
res=(num1>num2) ? (num1+num2):(num1-num2)
```

switch statement

 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-

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


Rules of switch statement



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.
- The value for a case must be the 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.
- Not every case needs to contain a break.
- A switch statement can have an optional default case, which must appear at the end of the switch.

switch statement



```
public class Test {  
  
    public static void main(String args[]) {  
  
        char grade = 'C';  
  
        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");  
  
            default :  
  
                System.out.println("Invalid grade");  
  
        }  
  
        System.out.println("Your grade is " + grade);  
  
    }  
}
```

Output:

Well done

Your grade is C

Looping Statements

Sr.No.	Statement & Description
1	while loop Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2	for loop Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	do...while loop Like a while statement, except that it tests the condition at the end of the loop body.
4	for-each loop The for-each loop is used to traverse array or collection in Java

while loop



Syntax-

```
while (condition) {  
    // code block to be executed  
}
```

The while loop loops through a block of code as long as a specified condition is true:

Example-

```
public class Main {  
    public static void main(String[] args) {  
        int i = 0;  
        while (i < 5) {  
            System.out.print(i);  
            i++;  
        }  
    }  
}
```

Output:

01234

for loop



Syntax-

```
for (statement 1; statement 2; statement 3) {  
    // code block to be executed  
}
```

- ✓ Statement 1 is executed (one time) before the execution of the code block.
- ✓ Statement 2 defines the condition for executing the code block.
- ✓ Statement 3 is executed (every time) after the code block has been executed.

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

Example-

```
public class Main {  
    public static void main(String[] args) {  
  
        for (int i = 0; i < 5; i++) {  
            System.out.print(i);  
        }  
    }  
}
```

Output:

01234

do...while loop



Syntax-

```
do{  
    //code to be executed / loop body  
    //update statement  
}while (condition);
```

The Java do-while loop is used to iterate a part of the program repeatedly, until the specified condition is true.

If the number of iteration is not fixed and you must have to execute the loop at least once, use a do-while loop. it is an exit control loop.

Example-

```
public class Main {  
    public static void main(String[] args) {  
        int i=1;  
        do{  
            System.out.print(i+" ");  
            i++;  
        }while(i<=3);  
    }  
}
```

Output: 1 2 3

for-each loop



Syntax-

```
for (type variableName : arrayName)
{
    // code block to be executed
}
```

for-each loop, which is used exclusively to loop through elements in an array:

Example-

```
public class Main {
    public static void main(String[] args) {
        String[] lang = {"CPP", "JAVA", "PYTHON",
"RUBY"};
        for (String i : lang) {
            System.out.println(i);
        }
    }
}
```

Output:

```
CPP
JAVA
PYTHON
RUBY
```

Jump statements



Jumping statements are control statements that transfer execution control from one point to another point in the program.

There are two Jump statements that are provided in the Java programming language:

1. **Break** statement.
2. **Continue** statement.

Break statement



The break statement used to jump out of a loop or switch statement.

- **Example:**

```
public class Main {  
    public static void main(String[] args) {  
        for (int i = 0; i < 10; i++) {  
            if (i == 4) {  
                break; // stops the loop when i is equal to 4:  
            }  
            System.out.print(i);  
        }  
    }  
}
```

Output:

0123

Continue statement



The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

- **Example:**

```
public class Main {  
    public static void main(String[] args) {  
        for (int i = 0; i < 10; i++) {  
            if (i == 4) {  
                continue; // skips the value of 4  
            }  
            System.out.print(i);  
        }  
    }  
}
```

Output:

012356789

exit() method in JAVA



The `exit()` method of `System` class terminates the current Java virtual machine running on system. This method takes status code as an argument.

Syntax

```
public static void exit(int status)
```

Status - `exit(0)` - indicates Successful termination

Status - `exit(-1)` - indicates unsuccessful termination with Exception

Status - `exit(1)` - indicates Unsuccessful termination

Example:

```
System.exit(0); //terminate the JVM
```



Thanks

Anirudha Gaikwad