### Java control statements

### Java If-else Statement

The Java *if statement* is used to test the condition. It checks boolean condition: *true* or *false*. There are various types of if statement in java.

if statement

if-else statement

if-else-if ladder

nested if statement

### Java if Statement

The Java if statement tests the condition. It executes the *if block* if condition is true.

**Syntax:**

**if**(condition){

//code to be executed

}



**Example:**

//Java Program to demonstate the use of if statement.

**public** **class** IfExample {

**public** **static** **void** main(String[] args) {

    //defining an 'age' variable

**int** age=20;

    //checking the age

**if**(age>18){

        System.out.print("Age is greater than 18");

    }

}

}

Output:

Age is greater than 18

### Java if-else Statement

The Java if-else statement also tests the condition. It executes the *if block* if condition is true otherwise *else block* is executed.

**Syntax:**

**if**(condition){

//code if condition is true

}**else**{

//code if condition is false

}



**Example:**

//A Java Program to demonstrate the use of if-else statement.

//It is a program of odd and even number.

**public** **class** IfElseExample {

**public** **static** **void** main(String[] args) {

    //defining a variable

**int** number=13;

    //Check if the number is divisible by 2 or not

**if**(number%2==0){

        System.out.println("even number");

    }**else**{

        System.out.println("odd number");

    }

}

}

Output:

odd number

**Leap Year Example:**

A year is leap, if it is divisible by 4 and 400. But, not by 100.

**public** **class** LeapYearExample {

**public** **static** **void** main(String[] args) {

**int** year=2020;

**if**(((year % 4 ==0) && (year % 100 !=0)) || (year % 400==0)){

        System.out.println("LEAP YEAR");

    }

**else**{

        System.out.println("COMMON YEAR");

    }

}

}

Output:

LEAP YEAR

### Using Ternary Operator

We can also use ternary operator (? :) to perform the task of if...else statement. It is a shorthand way to check the condition. If the condition is true, the result of ? is returned. But, if the condition is false, the result of : is returned.

**Example:**

**public** **class** IfElseTernaryExample {

**public** **static** **void** main(String[] args) {

**int** number=13;

    //Using ternary operator

    String output=(number%2==0)?"even number":"odd number";

    System.out.println(output);

}

}

Output:

odd number

### Java if-else-if ladder Statement

The if-else-if ladder statement executes one condition from multiple statements.

**Syntax:**

**if**(condition1){

//code to be executed if condition1 is true

}**else** **if**(condition2){

//code to be executed if condition2 is true

}

**else** **if**(condition3){

//code to be executed if condition3 is true

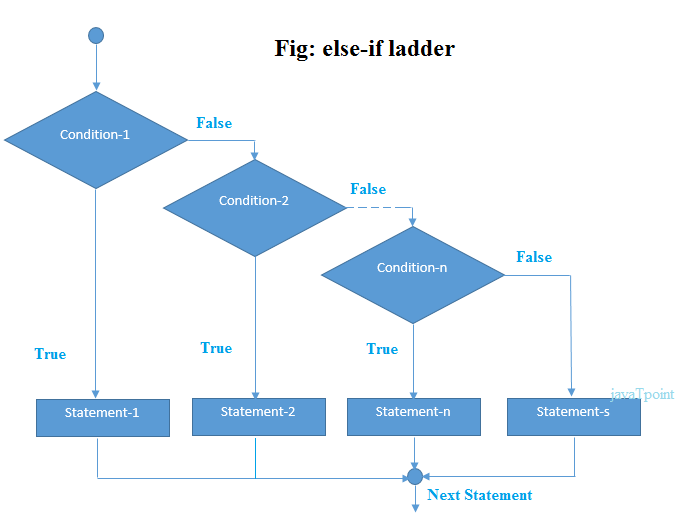
}

...

**else**{

//code to be executed if all the conditions are false

}



**Example:**

//Java Program to demonstrate the use of If else-if ladder.

//It is a program of grading system for fail, D grade, C grade, B grade, A grade and A+.

**public** **class** IfElseIfExample {

**public** **static** **void** main(String[] args) {

**int** marks=65;

**if**(marks<50){

        System.out.println("fail");

    }

**else** **if**(marks>=50 && marks<60){

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

    }

**else** **if**(marks>=60 && marks<70){

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

    }

**else** **if**(marks>=70 && marks<80){

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

    }

**else** **if**(marks>=80 && marks<90){

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

    }**else** **if**(marks>=90 && marks<100){

        System.out.println("A+ grade");

    }**else**{

        System.out.println("Invalid!");

    }

}

}

Output:

C grade

**Program to check POSITIVE, NEGATIVE or ZERO:**

**public** **class** PositiveNegativeExample {

**public** **static** **void** main(String[] args) {

**int** number=-13;

**if**(number>0){

    System.out.println("POSITIVE");

    }**else** **if**(number<0){

    System.out.println("NEGATIVE");

    }**else**{

    System.out.println("ZERO");

   }

}

}

Output:

NEGATIVE

### Java Nested if statement

The nested if statement represents the *if block within another if block*. Here, the inner if block condition executes only when outer if block condition is true.

**Syntax:**

**if**(condition){

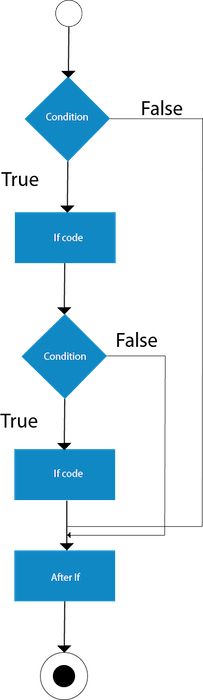
     //code to be executed

**if**(condition){

             //code to be executed

    }

}



**Example:**

//Java Program to demonstrate the use of Nested If Statement.

**public** **class** JavaNestedIfExample {

**public** **static** **void** main(String[] args) {

    //Creating two variables for age and weight

**int** age=20;

**int** weight=80;

    //applying condition on age and weight

**if**(age>=18){

**if**(weight>50){

            System.out.println("You are eligible to donate blood");

        }

    }

}}

Output:

You are eligible to donate blood

**Example 2:**

//Java Program to demonstrate the use of Nested If Statement.

**public** **class** JavaNestedIfExample2 {

**public** **static** **void** main(String[] args) {

    //Creating two variables for age and weight

**int** age=25;

**int** weight=48;

    //applying condition on age and weight

**if**(age>=18){

**if**(weight>50){

            System.out.println("You are eligible to donate blood");

        } **else**{

            System.out.println("You are not eligible to donate blood");

        }

    } **else**{

      System.out.println("Age must be greater than 18");

    }

}  }

Output:

You are not eligible to donate blood

### Java Switch Statement

The Java switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement. The switch statement works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long. Since Java 7, you can use strings in the switch statement.

In other words, the switch statement tests the equality of a variable against multiple values.

#### Points to Remember

There can be one or N number of case values for a switch expression.

The case value must be of switch expression type only. The case value must be literal or constant. It doesn't allow variables.

The case values must be unique. In case of duplicate value, it renders compile-time error.

The Java switch expression must be of byte, short, int, long (with its Wrapper type), enums and string.

* Each case statement can have a break statement which is optional. When control reaches to the break statement, it jumps the control after the switch expression. If a break statement is not found, it executes the next case.
* The case value can have a default label which is optional.

**Syntax:**

**switch**(expression){

**case** value1:

 //code to be executed;

**break**;  //optional

**case** value2:

 //code to be executed;

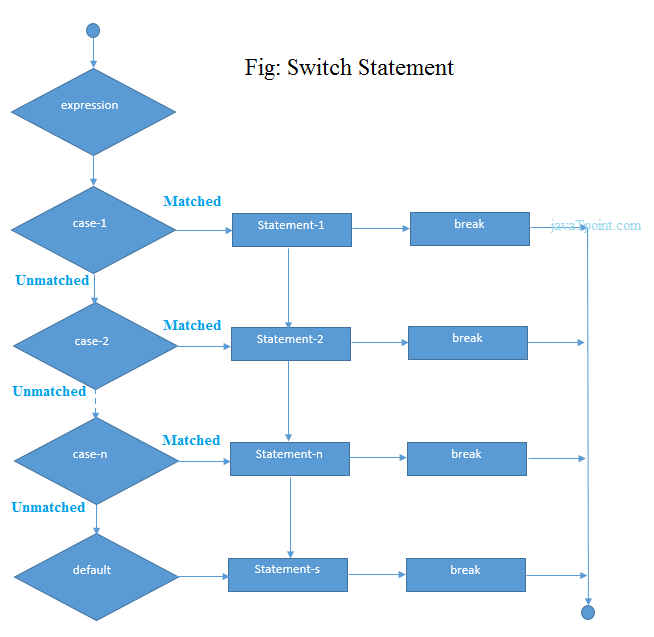
**break**;  //optional

......

**default**:

 code to be executed **if** all cases are not matched;

}



**Example:**

**public** **class** SwitchExample {

**public** **static** **void** main(String[] args) {

    //Declaring a variable for switch expression

**int** number=20;

    //Switch expression

**switch**(number){

    //Case statements

**case** 10: System.out.println("10");

**break**;

**case** 20: System.out.println("20");

**break**;

**case** 30: System.out.println("30");

**break**;

    //Default case statement

**default**:System.out.println("Not in 10, 20 or 30");

    }

}

}

Output:

20

**Finding Month Example:**

//Java Program to demonstrate the example of Switch statement

//where we are printing month name for the given number

**public** **class** SwitchMonthExample {

**public** **static** **void** main(String[] args) {

    //Specifying month number

**int** month=7;

    String monthString="";

    //Switch statement

**switch**(month){

    //case statements within the switch block

**case** 1: monthString="1 - January";

**break**;

**case** 2: monthString="2 - February";

**break**;

**case** 3: monthString="3 - March";

**break**;

**case** 4: monthString="4 - April";

**break**;

**case** 5: monthString="5 - May";

**break**;

**case** 6: monthString="6 - June";

**break**;

**case** 7: monthString="7 - July";

**break**;

**case** 8: monthString="8 - August";

**break**;

**case** 9: monthString="9 - September";

**break**;

**case** 10: monthString="10 - October";

**break**;

**case** 11: monthString="11 - November";

**break**;

**case** 12: monthString="12 - December";

**break**;

**default**:System.out.println("Invalid Month!");

    }

    //Printing month of the given number

    System.out.println(monthString);

}

}

Output:

7 - July

**Program to check Vowel or Consonant:**

If the character is A, E, I, O, or U, it is vowel otherwise consonant. It is not case-sensitive.

**public** **class** SwitchVowelExample {

**public** **static** **void** main(String[] args) {

**char** ch='O';

**switch**(ch)

    {

**case** 'a':

            System.out.println("Vowel");

**break**;

**case** 'e':

            System.out.println("Vowel");

**break**;

**case** 'i':

            System.out.println("Vowel");

**break**;

**case** 'o':

            System.out.println("Vowel");

**break**;

**case** 'u':

            System.out.println("Vowel");

**break**;

**case** 'A':

            System.out.println("Vowel");

**break**;

**case** 'E':

            System.out.println("Vowel");

**break**;

**case** 'I':

            System.out.println("Vowel");

**break**;

**case** 'O':

            System.out.println("Vowel");

**break**;

**case** 'U':

            System.out.println("Vowel");

**break**;

**default**:

            System.out.println("Consonant");

    }

}

}

Output:

20

### Java Switch Statement is fall-through

The Java switch statement is fall-through. It means it executes all statements after the first match if a break statement is not present.

**Example:**

//Java Switch Example where we are omitting the

//break statement

**public** **class** SwitchExample2 {

**public** **static** **void** main(String[] args) {

**int** number=20;

    //switch expression with int value

**switch**(number){

    //switch cases without break statements

**case** 10: System.out.println("10");

**case** 20: System.out.println("20");

**case** 30: System.out.println("30");

**default**:System.out.println("Not in 10, 20 or 30");

    }

}

}

Output:

20

30

Not in 10, 20 or 30

### Java Switch Statement with String

Java allows us to use strings in switch expression since Java SE 7. The case statement should be string literal.

**Example:**

//Java Program to demonstrate the use of Java Switch

//statement with String

**public** **class** SwitchStringExample {

**public** **static** **void** main(String[] args) {

    //Declaring String variable

    String levelString="Expert";

**int** level=0;

    //Using String in Switch expression

**switch**(levelString){

    //Using String Literal in Switch case

**case** "Beginner": level=1;

**break**;

**case** "Intermediate": level=2;

**break**;

**case** "Expert": level=3;

**break**;

**default**: level=0;

**break**;

    }

    System.out.println("Your Level is: "+level);

}

}

Output:

Your Level is: 3

### Java Nested Switch Statement

We can use switch statement inside other switch statement in Java. It is known as nested switch statement.

**Example:**

//Java Program to demonstrate the use of Java Nested Switch

**public** **class** NestedSwitchExample {

**public** **static** **void** main(String args[])

      {

      //C - CSE, E - ECE, M - Mechanical

**char** branch = 'C';

**int** collegeYear = 4;

**switch**( collegeYear )

        {

**case** 1:

                System.out.println("English, Maths, Science");

**break**;

**case** 2:

**switch**( branch )

                {

**case** 'C':

                        System.out.println("Operating System, Java, Data Structure");

**break**;

**case** 'E':

                        System.out.println("Micro processors, Logic switching theory");

**break**;

**case** 'M':

                        System.out.println("Drawing, Manufacturing Machines");

**break**;

                }

**break**;

**case** 3:

**switch**( branch )

                {

**case** 'C':

                        System.out.println("Computer Organization, MultiMedia");

**break**;

**case** 'E':

                        System.out.println("Fundamentals of Logic Design, Microelectronics");

**break**;

**case** 'M':

                        System.out.println("Internal Combustion Engines, Mechanical Vibration");

**break**;

                }

**break**;

**case** 4:

**switch**( branch )

                {

**case** 'C':

                        System.out.println("Data Communication and Networks, MultiMedia");

**break**;

**case** 'E':

                        System.out.println("Embedded System, Image Processing");

**break**;

**case** 'M':

                        System.out.println("Production Technology, Thermal Engineering");

**break**;

                }

**break**;

        }

    }

}

Output:

Data Communication and Networks, MultiMedia

### Java Enum in Switch Statement

Java allows us to use enum in switch statement.

**Example:**

//Java Program to demonstrate the use of Enum

//in switch statement

**public** **class** JavaSwitchEnumExample {

**public** **enum** Day {  Sun, Mon, Tue, Wed, Thu, Fri, Sat  }

**public** **static** **void** main(String args[])

       {

         Day[] DayNow = Day.values();

**for** (Day Now : DayNow)

           {

**switch** (Now)

                {

**case** Sun:

                        System.out.println("Sunday");

**break**;

**case** Mon:

                        System.out.println("Monday");

**break**;

**case** Tue:

                        System.out.println("Tuesday");

**break**;

**case** Wed:

                        System.out.println("Wednesday");

**break**;

**case** Thu:

                        System.out.println("Thursday");

**break**;

**case** Fri:

                        System.out.println("Friday");

**break**;

**case** Sat:

                        System.out.println("Saturday");

**break**;

                }

            }

        }

}

**[Test it Now](https://compiler.javatpoint.com/opr/test.jsp?filename=JavaSwitchEnumExample" \t "_blank)**

Output:

Sunday

Monday

Twesday

Wednesday

Thursday

Friday

Saturday

### Java Wrapper in Switch Statement

Java allows us to use four wrapper classes: Byte, Short, Integer and Long in switch statement.

**Example:**

//Java Program to demonstrate the use of Wrapper class

//in switch statement

**public** **class** WrapperInSwitchCaseExample {

**public** **static** **void** main(String args[])

       {

            Integer age = 18;

**switch** (age)

            {

**case** (16):

                    System.out.println("You are under 18.");

**break**;

**case** (18):

                    System.out.println("You are eligible for vote.");

**break**;

**case** (65):

                    System.out.println("You are senior citizen.");

**break**;

**default**:

                    System.out.println("Please give the valid age.");

**break**;

            }

        }

}

Output:

You are eligible for vote.

### Loops in Java

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true. There are three types of loops in java.

* for loop
* while loop
* do-while loop



Java For Loop vs While Loop vs Do While Loop

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison** | **for loop** | **while loop** | **do while loop** |
| Introduction | The Java for loop is a control flow statement that iterates a part of the programs multiple times. | The Java while loop is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition. | The Java do while loop is a control flow statement that executes a part of the programs at least once and the further execution depends upon the given boolean condition. |
| When to use | If the number of iteration is fixed, it is recommended to use for loop. | If the number of iteration is not fixed, it is recommended to use while loop. | If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use the do-while loop. |
| Syntax | for(init;condition;incr/decr){  // code to be executed  } | while(condition){  //code to be executed  } | do{  //code to be executed  }while(condition); |
| Example | //for loop  for(int i=1;i<=10;i++){  System.out.println(i);  } | //while loop  int i=1;  while(i<=10){  System.out.println(i);  i++;  } | //do-while loop  int i=1;  do{  System.out.println(i);  i++;  }while(i<=10); |
| Syntax for infinitive loop | for(;;){  //code to be executed  } | while(true){  //code to be executed  } | do{  //code to be executed  }while(true); |

### Java For Loop

The Java *for loop* is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.

There are three types of for loops in java.

Simple For Loop

For-each or Enhanced For Loop

Labeled For Loop

Java Simple For Loop

A simple for loop is the same as C/C++. We can initialize the variable, check condition and increment/decrement value. It consists of four parts:

**Initialization**:

It is the initial condition which is executed once when the loop starts. Here, we can initialize the variable, or we can use an already initialized variable. It is an optional condition.

**Condition**:

It is the second condition which is executed each time to test the condition of the loop. It continues execution until the condition is false. It must return boolean value either true or false. It is an optional condition.

**Statement**:

The statement of the loop is executed each time until the second condition is false.

**Increment/Decrement**: It increments or decrements the variable value. It is an optional condition.

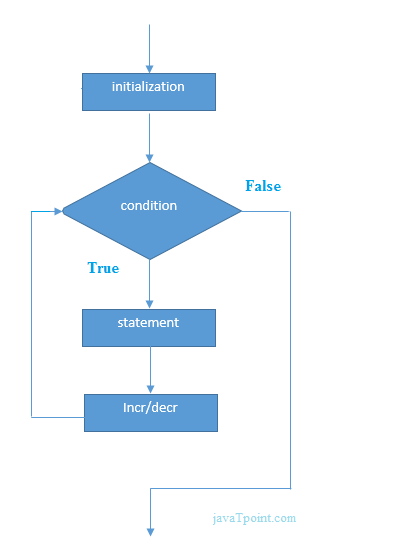
**Syntax:**

**for**(initialization;condition;incr/decr){

//statement or code to be executed

}

**Flowchart:**



**Example:**

//Java Program to demonstrate the example of for loop

//which prints table of 1

**public** **class** ForExample {

**public** **static** **void** main(String[] args) {

    //Code of Java for loop

**for**(**int** i=1;i<=10;i++){

        System.out.println(i);

    }

}

}

Output:

1

2

3

4

5

6

7

8

9

10

## Java Nested For Loop

If we have a for loop inside the another loop, it is known as nested for loop. The inner loop executes completely whenever outer loop executes.

**Example:**

**public** **class** NestedForExample {

**public** **static** **void** main(String[] args) {

//loop of i

**for**(**int** i=1;i<=3;i++){

//loop of j

**for**(**int** j=1;j<=3;j++){

        System.out.println(i+" "+j);

}//end of i

}//end of j

}

}

Output:

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

**Pyramid Example 1:**

**public** **class** PyramidExample {

**public** **static** **void** main(String[] args) {

**for**(**int** i=1;i<=5;i++){

**for**(**int** j=1;j<=i;j++){

        System.out.print("\* ");

}

System.out.println();//new line

}

}

}

Output:

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

**Pyramid Example 2:**

**public** **class** PyramidExample2 {

**public** **static** **void** main(String[] args) {

**int** term=6;

**for**(**int** i=1;i<=term;i++){

**for**(**int** j=term;j>=i;j--){

        System.out.print("\* ");

}

System.out.println();//new line

}

}

}

Output:

\* \* \* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

## Java for-each Loop

The for-each loop is used to traverse array or collection in java. It is easier to use than simple for loop because we don't need to increment value and use subscript notation.

It works on elements basis not index. It returns element one by one in the defined variable.

**Syntax:**

**for**(Type var:array){

//code to be executed

}

**Example:**

//Java For-each loop example which prints the

//elements of the array

**public** **class** ForEachExample {

**public** **static** **void** main(String[] args) {

    //Declaring an array

**int** arr[]={12,23,44,56,78};

    //Printing array using for-each loop

**for**(**int** i:arr){

        System.out.println(i);

    }

}

}

Output:

12

23

44

56

78

## Java Labeled For Loop

We can have a name of each Java for loop. To do so, we use label before the for loop. It is useful if we have nested for loop so that we can break/continue specific for loop.

Usually, break and continue keywords breaks/continues the innermost for loop only.

**Syntax:**

labelname:

**for**(initialization;condition;incr/decr){

//code to be executed

}

**Example:**

//A Java program to demonstrate the use of labeled for loop

**public** **class** LabeledForExample {

**public** **static** **void** main(String[] args) {

    //Using Label for outer and for loop

    aa:

**for**(**int** i=1;i<=3;i++){

            bb:

**for**(**int** j=1;j<=3;j++){

**if**(i==2&&j==2){

**break** aa;

                    }

                    System.out.println(i+" "+j);

                }

        }

}

}

Output:

1 1

1 2

1 3

2 1

If you use **break bb;**, it will break inner loop only which is the default behavior of any loop.

**public** **class** LabeledForExample2 {

**public** **static** **void** main(String[] args) {

    aa:

**for**(**int** i=1;i<=3;i++){

            bb:

**for**(**int** j=1;j<=3;j++){

**if**(i==2&&j==2){

**break** bb;

                    }

                    System.out.println(i+" "+j);

                }

        }

}

}

Output:

1 1

1 2

1 3

2 1

3 1

3 2

3 3

Java Infinitive For Loop

If you use two semicolons ;; in the for loop, it will be infinitive for loop.

**Syntax:**

**for**(;;){

//code to be executed

}

**Example:**

//Java program to demonstrate the use of infinite for loop

//which prints an statement

**public** **class** ForExample {

**public** **static** **void** main(String[] args) {

    //Using no condition in for loop

**for**(;;){

        System.out.println("infinitive loop");

    }

}

}

Output:

infinitive loop

infinitive loop

infinitive loop

infinitive loop

infinitive loop

ctrl+c

Now, you need to press ctrl+c to exit from the program.

### Java While Loop

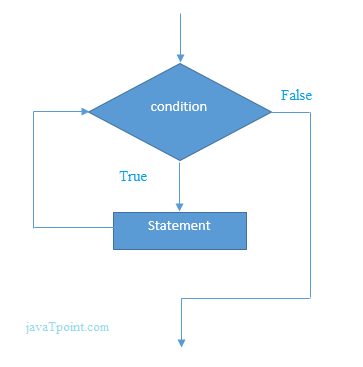
The Java *while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

**Syntax:**

**while**(condition){

//code to be executed

}



**Example:**

1. **public** **class** WhileExample {
2. **public** **static** **void** main(String[] args) {
3. **int** i=1;
4. **while**(i<=10){
5. System.out.println(i);
6. i++;
7. }
8. }
9. }

Output:

1

2

3

4

5

6

7

8

9

10

Java Infinitive While Loop

If you pass **true** in the while loop, it will be infinitive while loop.

**Syntax:**

while(true){

//code to be executed

}

**Example:**

**public** **class** WhileExample2 {

**public** **static** **void** main(String[] args) {

**while**(**true**){

        System.out.println("infinitive while loop");

    }

}

}

Output:

infinitive while loop

infinitive while loop

infinitive while loop

infinitive while loop

infinitive while loop

ctrl+c

### Java do-while Loop

The Java *do-while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.

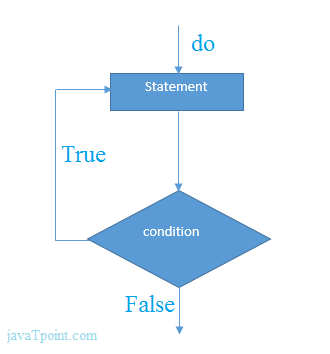
The Java *do-while loop* is executed at least once because condition is checked after loop body.

**Syntax:**

**do**{

//code to be executed

}**while**(condition);



**Example:**

**public** **class** DoWhileExample {

**public** **static** **void** main(String[] args) {

**int** i=1;

**do**{

        System.out.println(i);

    i++;

    }**while**(i<=10);

}

}

Output:

1

2

3

4

5

6

7

8

9

10

### Java Infinitive do-while Loop

If you pass **true** in the do-while loop, it will be infinitive do-while loop.

**Syntax:**

**do**{

//code to be executed

}**while**(**true**);

**Example:**

**public** **class** DoWhileExample2 {

**public** **static** **void** main(String[] args) {

**do**{

        System.out.println("infinitive do while loop");

    }**while**(**true**);

}

}

Output:

infinitive do while loop

infinitive do while loop

infinitive do while loop

ctrl+c

### Java Break Statement

When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

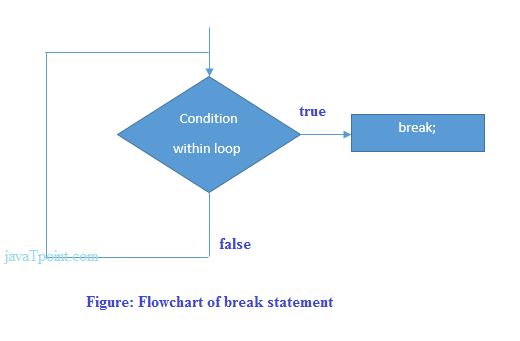
The Java *break* is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

We can use Java break statement in all types of loops such as for loop, while loop and do-while loop.

**Syntax:**

jump-statement;

**break**;



### Java Break Statement with Loop

**Example:**

//Java Program to demonstrate the use of break statement

//inside the for loop.

**public** **class** BreakExample {

**public** **static** **void** main(String[] args) {

    //using for loop

**for**(**int** i=1;i<=10;i++){

**if**(i==5){

            //breaking the loop

**break**;

        }

        System.out.println(i);

    }

}

}

Output:

1

2

3

4

## Java Break Statement with Inner Loop

It breaks inner loop only if you use break statement inside the inner loop.

**Example:**

//Java Program to illustrate the use of break statement

//inside an inner loop

**public** **class** BreakExample2 {

**public** **static** **void** main(String[] args) {

            //outer loop

**for**(**int** i=1;i<=3;i++){

                    //inner loop

**for**(**int** j=1;j<=3;j++){

**if**(i==2&&j==2){

                            //using break statement inside the inner loop

**break**;

                        }

                        System.out.println(i+" "+j);

                    }

            }

}

}

Output:

1 1

1 2

1 3

2 1

3 1

3 2

3 3

Java Break Statement with Labeled For Loop

We can use break statement with a label. This feature is introduced since JDK 1.5. So, we can break any loop in Java now whether it is outer loop or inner.

**Example:**

//Java Program to illustrate the use of continue statement

//with label inside an inner loop to break outer loop

**public** **class** BreakExample3 {

**public** **static** **void** main(String[] args) {

            aa:

**for**(**int** i=1;i<=3;i++){

                    bb:

**for**(**int** j=1;j<=3;j++){

**if**(i==2&&j==2){

                            //using break statement with label

**break** aa;

                        }

                        System.out.println(i+" "+j);

                    }

            }

}

}

Output:

1 1

1 2

1 3

2 1

Java Break Statement in while loop

**Example:**

//Java Program to demonstrate the use of break statement

//inside the while loop.

**public** **class** BreakWhileExample {

**public** **static** **void** main(String[] args) {

    //while loop

**int** i=1;

**while**(i<=10){

**if**(i==5){

            //using break statement

            i++;

**break**;//it will break the loop

        }

        System.out.println(i);

        i++;

    }

}

}

Output:

1

2

3

4

Java Break Statement in do-while loop

**Example:**

//Java Program to demonstrate the use of break statement

//inside the Java do-while loop.

**public** **class** BreakDoWhileExample {

**public** **static** **void** main(String[] args) {

    //declaring variable

**int** i=1;

    //do-while loop

**do**{

**if**(i==5){

           //using break statement

           i++;

**break**;//it will break the loop

        }

        System.out.println(i);

        i++;

    }**while**(i<=10);

}

}

Output:

1

2

3

4

### Java Continue Statement

The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately. It can be used with for loop or while loop.

The Java *continue statement* is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only.

We can use Java continue statement in all types of loops such as for loop, while loop and do-while loop.

**Syntax:**

jump-statement;

**continue**;

## Java Continue Statement Example

**Example:**

//Java Program to demonstrate the use of continue statement

//inside the for loop.

**public** **class** ContinueExample {

**public** **static** **void** main(String[] args) {

    //for loop

**for**(**int** i=1;i<=10;i++){

**if**(i==5){

            //using continue statement

**continue**;//it will skip the rest statement

        }

        System.out.println(i);

    }

}

}

Output:

1

2

3

4

6

7

8

9

10

As you can see in the above output, 5 is not printed on the console. It is because the loop is continued when it reaches to 5.

Java Continue Statement with Inner Loop

It continues inner loop only if you use the continue statement inside the inner loop.

**Example:**

//Java Program to illustrate the use of continue statement

//inside an inner loop

**public** **class** ContinueExample2 {

**public** **static** **void** main(String[] args) {

            //outer loop

**for**(**int** i=1;i<=3;i++){

                    //inner loop

**for**(**int** j=1;j<=3;j++){

**if**(i==2&&j==2){

                            //using continue statement inside inner loop

**continue**;

                        }

                        System.out.println(i+" "+j);

                    }

            }

}

}

Output:

1 1

1 2

1 3

2 1

2 3

3 1

3 2

3 3

Java Continue Statement with Labeled For Loop

We can use continute statement with a label. This feature is introduced since JDK 1.5. So, we can continue any loop in Java now whether it is outer loop or inner.

**Example:**

//Java Program to illustrate the use of continue statement

//with label inside an inner loop to continue outer loop

**public** **class** ContinueExample3 {

**public** **static** **void** main(String[] args) {

            aa:

**for**(**int** i=1;i<=3;i++){

                    bb:

**for**(**int** j=1;j<=3;j++){

**if**(i==2&&j==2){

                            //using continue statement with label

**continue** aa;

                        }

                        System.out.println(i+" "+j);

                    }

            }

}

}

Output:

1 1

1 2

1 3

2 1

3 1

3 2

3 3

Java Continue Statement in while loop

**Example:**

//Java Program to demonstrate the use of continue statement

//inside the while loop.

**public** **class** ContinueWhileExample {

**public** **static** **void** main(String[] args) {

    //while loop

**int** i=1;

**while**(i<=10){

**if**(i==5){

            //using continue statement

            i++;

**continue**;//it will skip the rest statement

        }

        System.out.println(i);

        i++;

    }

}

}

Output:

1

2

3

4

6

7

8

9

10

Java Continue Statement in do-while loop

**Example:**

//Java Program to demonstrate the use of continue statement

//inside the Java do-while loop.

**public** **class** ContinueDoWhileExample {

**public** **static** **void** main(String[] args) {

    //declaring variable

**int** i=1;

    //do-while loop

**do**{

**if**(i==5){

                //using continue statement

                 i++;

**continue**;//it will skip the rest statement

        }

        System.out.println(i);

        i++;

    }**while**(i<=10);

}

}

Output:

1

2

3

4

6

7

8

9

10

### Java Comments

The java comments are statements that are not executed by the compiler and interpreter. The comments can be used to provide information or explanation about the variable, method, class or any statement. It can also be used to hide program code for specific time.

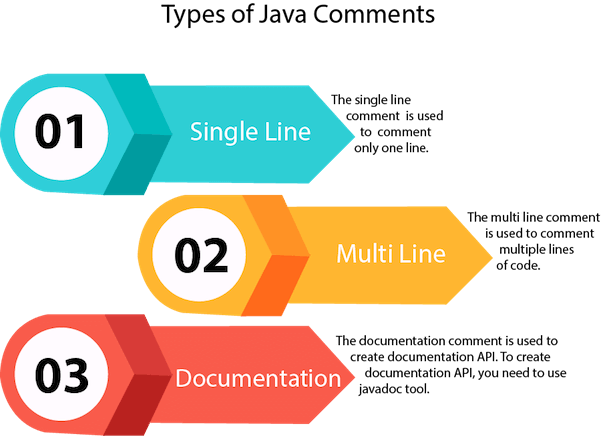
Types of Java Comments

There are 3 types of comments in java.

Single Line Comment

Multi Line Comment

Documentation Comment



1) Java Single Line Comment

The single line comment is used to comment only one line.

**Syntax:**

//This is single line comment

**Example:**

**public** **class** CommentExample1 {

**public** **static** **void** main(String[] args) {

**int** i=10;//Here, i is a variable

    System.out.println(i);

}

}

Output:

10

2) Java Multi Line Comment

The multi line comment is used to comment multiple lines of code.

**Syntax:**

/\*

This

is

multi line

comment

\*/

**Example:**

**public** **class** CommentExample2 {

**public** **static** **void** main(String[] args) {

/\* Let's declare and

 print variable in java. \*/

**int** i=10;

    System.out.println(i);

}

}

Output:

10

3) Java Documentation Comment

The documentation comment is used to create documentation API. To create documentation API, you need to use **javadoc tool**.

**Syntax:**

/\*\*

This

is

documentation

comment

\*/

**Example:**

/\*\* The Calculator class provides methods to get addition and subtraction of given 2 numbers.\*/

**public** **class** Calculator {

/\*\* The add() method returns addition of given numbers.\*/

**public** **static** **int** add(**int** a, **int** b){**return** a+b;}

/\*\* The sub() method returns subtraction of given numbers.\*/

**public** **static** **int** sub(**int** a, **int** b){**return** a-b;}

}

Compile it by javac tool:

javac Calculator.java

Create Documentation API by javadoc tool:

javadoc Calculator.java

Now, there will be HTML files created for your Calculator class in the current directory. Open the HTML files and see the explanation of Calculator class provided through documentation comment.

### Java Programs

Java programs are frequently asked in the interview. These programs can be asked from control statements, array, string, oops etc. Let's see the list of java programs.

[Java Sorting Programs](https://www.javatpoint.com/java-programs#java-sorting-programs)

[Java Searching Programs](https://www.javatpoint.com/java-programs#java-searching-programs)

[Java Array Programs](https://www.javatpoint.com/java-programs#java-array-programs)

[Java Matrix Programs](https://www.javatpoint.com/java-programs#java-matrix-programs)

[Java String programs](https://www.javatpoint.com/java-programs#java-string-programs)

[Pattern programs](https://www.javatpoint.com/java-programs#java-pattern-programs)

[Singly Linked List Programs](https://www.javatpoint.com/java-programs#singly-linked-list-programs)

[Circular Linked List Programs](https://www.javatpoint.com/java-programs#circular-linked-list-programs)

[Doubly Linked List Programs](https://www.javatpoint.com/java-programs#doubly-linked-list-programs)

[Tree Programs](https://www.javatpoint.com/java-programs#tree-programs)

## 1) [Fibonacci series](https://www.javatpoint.com/fibonacci-series-in-java)

Write a java program to print fibonacci series without using recursion and using recursion.

Input: 10

**Output:** 0 1 1 2 3 5 8 13 21 34

# 2) [Prime number](https://www.javatpoint.com/prime-number-program-in-java)

Write a java program to check prime number.

Input: 44

**Output:** not prime number

Input: 7

**Output:** prime number

# 3) [Palindrome number](https://www.javatpoint.com/palindrome-program-in-java)

Write a java program to check palindrome number.

**Input:** 329

**Output:** not palindrome number

**Input:** 12321

**Output:** palindrome number

# 4) [Factorial number](https://www.javatpoint.com/factorial-program-in-java)

Write a java program to print factorial of a number.

**Input:** 5

Output: 120

**Input:** 6

Output: 720

## 5) [Armstrong number](https://www.javatpoint.com/armstrong-number-in-java)

Write a java program to check Armstrong number.

**Input:** 153

**Output:** Armstrong number

**Input:** 22

**Output:** not Armstrong number

## Java Sorting Programs

1) [Bubble Sort](https://www.javatpoint.com/bubble-sort-in-java)

Write a java program to sort an array elements using bubble sort algorithm.

**Input:** 18 9 33 4 84 32

**Output:** 4 9 18 32 33 84

2) [Selection Sort](https://www.javatpoint.com/selection-sort-in-java)

Write a java program to sort an array elements using selection sort algorithm.

**Input:** 18 9 33 4 84 32

**Output:** 4 9 18 32 33 84

3) [Insertion Sort](https://www.javatpoint.com/insertion-sort-in-java)

Write a java program to sort an array elements using insertion sort algorithm.

**Input:** 18 9 33 4 84 32

**Output:** 4 9 18 32 33 84

## Java Searching Programs

1) [Linear Search](https://www.javatpoint.com/linear-search-in-java)

Write a java program to perform linear search in java.

2) [Binary Search](https://www.javatpoint.com/binary-search-in-java)

Write a java program to perform binary search in java.

## Java Array Programs

[1) Java Program to copy all elements of one array into another array](https://www.javatpoint.com/java-program-to-copy-all-elements-of-one-array-into-another-array)

[2) Java Program to find the frequency of each element in the array](https://www.javatpoint.com/java-program-to-find-the-frequency-of-each-element-in-the-array)

[3) Java Program to left rotate the elements of an array](https://www.javatpoint.com/java-program-to-left-rotate-the-elements-of-an-array)

[4) Java Program to print the duplicate elements of an array](https://www.javatpoint.com/java-program-to-print-the-duplicate-elements-of-an-array)

[5) Java Program to print the elements of an array](https://www.javatpoint.com/java-program-to-print-the-elements-of-an-array)

[6) Java Program to print the elements of an array in reverse order](https://www.javatpoint.com/java-program-to-print-the-elements-of-an-array-in-reverse-order)

[7) Java Program to print the elements of an array present on even position](https://www.javatpoint.com/java-program-to-print-the-elements-of-an-array-present-on-even-position)

[8) Java Program to print the elements of an array present on odd position](https://www.javatpoint.com/java-program-to-print-the-elements-of-an-array-present-on-odd-position)

[9) Java Program to print the largest element in an array](https://www.javatpoint.com/java-program-to-print-the-largest-element-in-an-array)

[10) Java Program to print the smallest element in an array](https://www.javatpoint.com/java-program-to-print-the-smallest-element-in-an-array)

[11) Java Program to print the number of elements present in an array](https://www.javatpoint.com/java-program-to-print-the-number-of-elements-present-in-an-array)

[12) Java Program to print the sum of all the items of the array](https://www.javatpoint.com/java-program-to-print-the-sum-of-all-the-items-of-the-array)

[13) Java Program to right rotate the elements of an array](https://www.javatpoint.com/java-program-to-right-rotate-the-elements-of-an-array)

[14) Java Program to sort the elements of an array in ascending order](https://www.javatpoint.com/java-program-to-sort-the-elements-of-an-array-in-ascending-order)

[15) Java Program to sort the elements of an array in descending order](https://www.javatpoint.com/java-program-to-sort-the-elements-of-an-array-in-descending-order)

16) [Find 3rd Largest Number in an Array](https://www.javatpoint.com/java-program-to-find-third-largest-number-in-an-array)

Write a java program to find 3rd largest number in an array.

**Input:** 1,2,5,6,3,2

Output: 3

17) [Find 2nd Largest Number in an Array](https://www.javatpoint.com/java-program-to-find-second-largest-number-in-an-array)

Write a java program to find 2nd largest number in an array.

**Input:** 1,2,5,6,3,2

Output: 5

18) [Find Largest Number in an Array](https://www.javatpoint.com/java-program-to-find-largest-number-in-an-array)

Write a java program to find largest number in an array.

**Input:** 1,2,5,6,3,2

Output: 6

19) [Find 2nd Smallest Number in an Array](https://www.javatpoint.com/java-program-to-find-second-smallest-number-in-an-array)

Write a java program to find 2nd smallest number in an array.

**Input:** 1,2,5,6,3,2

Output: 2

20) [Find Smallest Number in an Array](https://www.javatpoint.com/java-program-to-find-smallest-number-in-an-array)

Write a java program to find smallest number in an array.

**Input:** 1,2,5,6,3,2

Output: 1

## 21) [Remove Duplicate Element in an Array](https://www.javatpoint.com/java-program-to-remove-duplicate-element-in-an-array)

Write a java program to remove duplicate element in an array.

**Input:** 10,20,20,30,30,40,50,50

**Output:** 10 20 30 40 50

22) [Add Two Matrices](https://www.javatpoint.com/java-program-to-add-two-matrices)

Write a java program to add two matrices.

**Input:**

First matrix elements:

1 1 1

2 2 2

3 3 3

Second matrix elements:

1 1 1

2 2 2

3 3 3

**Output:**

Addition of the matrix:

2 2 2

4 4 4

6 6 6

23) [Multiply Two Matrices](https://www.javatpoint.com/java-program-to-multiply-two-matrices)

Write a java program to multiply two matrices.

**Input:**

First matrix elements:

1 1 1

2 2 2

3 3 3

Second matrix elements:

1 1 1

2 2 2

3 3 3

**Output:**

Multiplication of the matrix:

6 6 6

12 12 12

18 18 18

24) [Print Odd and Even Number from an Array](https://www.javatpoint.com/java-program-to-print-odd-and-even-numbers-from-an-array)

Write a java program to print odd and even number from an array.

**Input:** 1,2,5,6,3,2

Output:

Odd Numbers:

1

5

3

Even Numbers:

2

6

2

25) [Transpose matrix](https://www.javatpoint.com/java-program-to-transpose-matrix)

Write a java program to transpose a matrix.

Output:

Printing Matrix without transpose:

1 3 4

2 4 3

3 4 5

Printing Matrix After Transpose:

1 2 3

3 4 4

4 3 5

## Java Matrix Programs

[Java Matrix Programs](https://www.javatpoint.com/java-matrix-programs)

[2) Java Program to subtract the two matrices](https://www.javatpoint.com/java-program-to-subtract-the-two-matrices)

[3) Java Program to determine whether a given matrix is an identity matrix](https://www.javatpoint.com/java-program-to-determine-whether-a-given-matrix-is-an-identity-matrix)

[4) Java Program to determine whether a given matrix is a sparse matrix](https://www.javatpoint.com/java-program-to-determine-whether-a-given-matrix-is-a-sparse-matrix)

[5) Java Program to determine whether two matrices are equal](https://www.javatpoint.com/java-program-to-determine-whether-two-matrices-are-equal)

[6) Java Program to display the lower triangular matrix](https://www.javatpoint.com/java-program-to-display-the-lower-triangular-matrix)

[7) Java Program to display the upper triangular matrix](https://www.javatpoint.com/java-program-to-display-the-upper-triangular-matrix)

[8) Java Program to find the frequency of odd & even numbers in the given matrix](https://www.javatpoint.com/java-program-to-find-the-frequency-of-odd-and-even-numbers-in-the-given-matrix)

[9) Java Program to find the product of two matrices](https://www.javatpoint.com/java-program-to-find-the-product-of-two-matrices)

[10) Java Program to find the sum of each row and each column of a matrix](https://www.javatpoint.com/java-program-to-find-the-sum-of-each-row-and-each-column-of-a-matrix)

[11) Java Program to find the transpose of a given matrix](https://www.javatpoint.com/java-program-to-find-the-transpose-of-a-given-matrix)

## Java String programs

1) [Java Program to count the total number of characters in a string](https://www.javatpoint.com/java-program-to-count-the-total-number-of-characters-in-a-string)

2) [Java Program to count the total number of characters in a string 2](https://www.javatpoint.com/java-program-to-count-the-total-number-of-characters-in-a-string-2)

3) [Java Program to count the total number of punctuation characters exists in a String](https://www.javatpoint.com/java-program-to-count-the-total-number-of-punctuation-characters-exists-in-a-string)

4) [Java Program to count the total number of vowels and consonants in a string](https://www.javatpoint.com/java-program-to-count-the-total-number-of-vowels-and-consonants-in-a-string)

5) [Java Program to determine whether two strings are the anagram](https://www.javatpoint.com/java-program-to-determine-whether-two-strings-are-the-anagram)

6) [Java Program to divide a string in 'N' equal parts.](https://www.javatpoint.com/java-program-to-divide-a-string-in-n-equal-parts)

7) [Java Program to find all subsets of a string](https://www.javatpoint.com/java-program-to-find-all-subsets-of-a-string)

8) [Java Program to find the longest repeating sequence in a string](https://www.javatpoint.com/java-program-to-find-the-longest-repeating-sequence-in-a-string)

9) [Java Program to find all the permutations of a string](https://www.javatpoint.com/java-program-to-ind-all-the-permutations-of-a-string)

10) [Java Program to remove all the white spaces from a string](https://www.javatpoint.com/java-program-to-remove-all-the-white-spaces-from-a-string)

11) [Java Program to replace lower-case characters with upper-case and vice-versa](https://www.javatpoint.com/java-program-to-replace-lower-case-characters-with-upper-case-and-vice-versa)

12) [Java Program to replace the spaces of a string with a specific character](https://www.javatpoint.com/java-program-to-replace-the-spaces-of-a-string-with-a-specific-character)

13) [Java Program to determine whether a given string is palindrome](https://www.javatpoint.com/java-program-to-determine-whether-a-given-string-is-palindrome)

14) [Java Program to determine whether one string is a rotation of another](https://www.javatpoint.com/java-program-to-determine-whether-one-string-is-a-rotation-of-another)

15) [Java Program to find maximum and minimum occurring character in a string](https://www.javatpoint.com/java-program-to-find-maximum-and-minimum-occurring-character-in-a-string)

16) [Java Program to find Reverse of the string](https://www.javatpoint.com/java-program-to-find-reverse-of-the-string)

17) [Java program to find the duplicate characters in a string](https://www.javatpoint.com/java-program-to-find-the-duplicate-characters-in-a-string)

18) [Java program to find the duplicate words in a string](https://www.javatpoint.com/java-program-to-find-the-duplicate-words-in-a-string)

19) [Java Program to find the frequency of characters](https://www.javatpoint.com/java-program-to-find-the-frequency-of-characters)

20) [Java Program to find the largest and smallest word in a string](https://www.javatpoint.com/java-program-to-find-the-largest-and-smallest-word-in-a-string)

21) [Java Program to find the most repeated word in a text file](https://www.javatpoint.com/java-program-to-find-the-most-repeated-word-in-a-text-file)

22) [Java Program to find the number of the words in the given text file](https://www.javatpoint.com/java-program-to-find-the-number-of-the-words-in-the-given-text-file)

23) [Java Program to separate the Individual Characters from a String](https://www.javatpoint.com/java-program-to-separate-the-individual-characters-from-a-string)

24) [Java Program to swap two string variables without using third or temp variable.](https://www.javatpoint.com/java-program-to-swap-two-string-variables-without-using-third-or-temp-variable)

25) [Java Program to print smallest and biggest possible palindrome word in a given string](https://www.javatpoint.com/java-program-to-print-smallest-and-biggest-possible-palindrome-word-in-a-given-string)

## Java Pattern programs

1) [Java Program to count the total number of punctuation characters exists in a String](https://www.javatpoint.com/java-program-to-count-the-total-number-of-punctuation-characters-exists-in-a-string)

2) [Java program to print the following pattern](https://www.javatpoint.com/java-program-to-print-the-following-pattern)

3) [Java program to print the following pattern 2](https://www.javatpoint.com/java-program-to-print-the-following-pattern-2)

4) [Java program to print the following pattern 3](https://www.javatpoint.com/java-program-to-print-the-following-pattern-3)

5) [Java program to print the following pattern 4](https://www.javatpoint.com/java-program-to-print-the-following-pattern-4)

6) [Java program to print the following pattern 5](https://www.javatpoint.com/java-program-to-print-the-following-pattern-5)

7) [Java program to print the following pattern on the console](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console)

8) [Java program to print the following pattern on the console 2](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-2)

9) [Java program to print the following pattern on the console 3](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-3)

10) [Java program to print the following pattern on the console 4](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-4)

11) [Java program to print the following pattern on the console 5](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-5)

12) [Java program to print the following pattern on the console 6](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-6)

13) [Java program to print the following pattern on the console 7](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-7)

14) [Java program to print the following pattern on the console 8](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-8)

15) [Java program to print the following pattern on the console 9](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-9)

16) [Java program to print the following pattern on the console 10](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-10)

17) [Java program to print the following pattern on the console 11](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-11)

18) [Java program to print the following pattern on the console 12](https://www.javatpoint.com/java-program-to-print-the-following-pattern-on-the-console-12)

19) [Java program to print the following spiral pattern on the console](https://www.javatpoint.com/java-program-to-print-the-following-spiral-pattern-on-the-console)

## Java Singly Linked List Programs

1) [Singly linked list Examples in Java](https://www.javatpoint.com/java-singly-linked-list-example)

2) [Java Program to create and display a singly linked list](https://www.javatpoint.com/java-program-to-create-and-display-a-singly-linked-list)

3) [Java program to create a singly linked list of n nodes and count the number of nodes](https://www.javatpoint.com/java-program-to-create-a-singly-linked-list-of-n-nodes-and-count-the-number-of-nodes)

4) [Java program to create a singly linked list of n nodes and display it in reverse order](https://www.javatpoint.com/java-program-to-create-a-singly-linked-list-of-n-nodes-and-display-in-reverse-order)

5) [Java program to delete a node from the beginning of the singly linked list](https://www.javatpoint.com/java-program-to-delete-a-node-from-the-beginning-of-the-singly-linked-list)

6) [Java program to delete a node from the middle of the singly linked list](https://www.javatpoint.com/java-program-to-delete-a-node-from-the-middle-of-the-singly-linked-list)

7) [Java program to delete a node from the end of the singly linked list](https://www.javatpoint.com/java-program-to-delete-a-node-from-the-end-of-the-singly-linked-list)

8) [Java program to determine whether a singly linked list is the palindrome](https://www.javatpoint.com/java-program-to-determine-whether-a-singly-linked-list-is-the-palindrome)

9) [Java program to find the maximum and minimum value node from a linked list](https://www.javatpoint.com/java-program-to-find-the-maximum-and-minimum-value-node-from-a-linked-list)

10) [Java Program to insert a new node at the middle of the singly linked list](https://www.javatpoint.com/java-program-to-insert-a-new-node-at-the-middle-of-the-singly-linked-list)

11) [Java program to insert a new node at the beginning of the singly linked list](https://www.javatpoint.com/java-program-to-insert-a-new-node-at-the-beginning-of-the-singly-linked-list)

12) [Java program to insert a new node at the end of the singly linked list](https://www.javatpoint.com/java-program-to-insert-a-new-node-at-the-end-of-the-singly-linked-list)

13) [Java program to remove duplicate elements from a singly linked list](https://www.javatpoint.com/java-program-to-remove-duplicate-elements-from-a-singly-linked-list)

14) [Java Program to search an element in a singly linked list](https://www.javatpoint.com/java-program-to-search-an-element-in-a-singly-linked-list)

# Java Circular Linked List Programs

1) [Java program to create and display a Circular Linked List](https://www.javatpoint.com/java-program-to-create-and-display-a-circular-linked-list)

2) [Java program to create a Circular Linked List of N nodes and count the number of nodes](https://www.javatpoint.com/java-program-to-create-a-circular-linked-list-of-n-nodes-and-count-the-number-of-nodes)

3) [Java program to create a Circular Linked List of n nodes and display it in reverse order](https://www.javatpoint.com/java-program-to-create-a-circular-linked-list-of-n-nodes-and-display-it-in-reverse-order)

4) [Java program to delete a node from the beginning of the Circular Linked List](https://www.javatpoint.com/java-program-to-delete-a-node-from-the-beginning-of-the-circular-linked-list)

5) [Java program to delete a node from the end of the Circular Linked List](https://www.javatpoint.com/java-program-to-delete-a-node-from-the-end-of-the-circular-linked-list)

6) [Java program to delete a node from the middle of the Circular Linked List](https://www.javatpoint.com/java-program-to-delete-a-node-from-the-middle-of-the-circular-linked-list)

7) [Java program to find the maximum and minimum value node from a circular linked list](https://www.javatpoint.com/java-program-to-find-the-maximum-and-minimum-value-node-from-a-circular-linked-list)

8) [Java program to insert a new node at the beginning of the Circular Linked List](https://www.javatpoint.com/java-program-to-insert-a-new-node-at-the-beginning-of-the-circular-linked-list)

9) [Java program to insert a new node at the end of the Circular Linked List](https://www.javatpoint.com/java-program-to-insert-a-new-node-at-the-end-of-the-circular-linked-list)

10) [Java program to insert a new node at the middle of the Circular Linked List](https://www.javatpoint.com/java-program-to-insert-a-new-node-at-the-middle-of-the-circular-linked-list)

11) [Java program to remove duplicate elements from a Circular Linked List](https://www.javatpoint.com/java-program-to-remove-duplicate-elements-from-a-circular-linked-list)

12) [Java program to search an element in a Circular Linked List](https://www.javatpoint.com/java-program-to-search-an-element-in-a-circular-linked-list)

13) [Java program to sort the elements of the Circular Linked List](https://www.javatpoint.com/java-program-to-sort-the-elements-of-the-circular-linked-list)

# Java Doubly Linked List Programs

1) [Java program to convert a given binary tree to doubly linked list](https://www.javatpoint.com/java-program-to-convert-a-given-binary-tree-to-doubly-linked-list)

2) [Java program to create a doubly linked list from a ternary tree](https://www.javatpoint.com/java-program-to-create-a-doubly-linked-list-from-a-ternary-tree)

3) [Java program to create a doubly linked list of n nodes and count the number of nodes](https://www.javatpoint.com/java-program-to-create-a-doubly-linked-list-of-n-nodes-and-count-the-number-of-nodes)

4) [Java program to create a doubly linked list of n nodes and display it in reverse order](https://www.javatpoint.com/java-program-to-create-a-doubly-linked-list-of-n-nodes-and-display-it-in-reverse-order)

5) [Java program to create and display a doubly linked list](https://www.javatpoint.com/java-program-to-create-and-display-a-doubly-linked-list)

6) [Java program to delete a new node from the beginning of the doubly linked list](https://www.javatpoint.com/java-program-to-delete-a-new-node-from-the-beginning-of-the-doubly-linked-list)

7) [Java program to delete a new node from the end of the doubly linked list](https://www.javatpoint.com/java-program-to-delete-a-new-node-from-the-end-of-the-doubly-linked-list)

8) [Java program to delete a new node from the middle of the doubly linked list](https://www.javatpoint.com/java-program-to-delete-a-new-node-from-the-middle-of-the-doubly-linked-list)

9) [Java program to find the maximum and minimum value node from a doubly linked list](https://www.javatpoint.com/java-program-to-find-the-maximum-and-minimum-value-node-from-a-doubly-linked-list)

10) [Java program to insert a new node at the beginning of the Doubly Linked list](https://www.javatpoint.com/java-program-to-insert-a-new-node-at-the-beginning-of-the-doubly-linked-list)

10) [Java program to insert a new node at the end of the Doubly Linked List](https://www.javatpoint.com/java-program-to-insert-a-new-node-at-the-end-of-the-doubly-linked-list)

12) [Java program to insert a new node at the middle of the Doubly Linked List](https://www.javatpoint.com/java-program-to-insert-a-new-node-at-the-middle-of-the-doubly-linked-list)

13) [Java program to remove duplicate elements from a Doubly Linked List](https://www.javatpoint.com/java-program-to-remove-duplicate-elements-from-a-doubly-linked-list)

14) [Java program to rotate doubly linked list by N nodes](https://www.javatpoint.com/java-program-to-rotate-doubly-linked-list-by-n-nodes)

15) [Java program to search an element in a doubly linked list](https://www.javatpoint.com/java-program-to-search-an-element-in-a-doubly-linked-list)

16) [Java program to sort the elements of the doubly linked list](https://www.javatpoint.com/java-program-to-sort-the-elements-of-the-doubly-linked-list)

# Java Tree Programs

1) [Java Program to calculate the Difference between the Sum of the Odd Level and the Even Level Nodes of a Binary Tree](https://www.javatpoint.com/java-program-to-calculate-the-difference-between-the-sum-of-the-odd-level-and-the-even-level-nodes-of-a-binary-tree)

2) [Java program to construct a Binary Search Tree and perform deletion and In-order traversal](https://www.javatpoint.com/java-program-to-construct-a-binary-search-tree-and-perform-deletion-and-in-order-traversal)

3) [Java program to convert Binary Tree to Binary Search Tree](https://www.javatpoint.com/java-program-to-convert-binary-tree-to-binary-search-tree)

4) [Java program to determine whether all leaves are at same level](https://www.javatpoint.com/java-program-to-determine-whether-all-leaves-are-at-same-level)

5) [Java program to determine whether two trees are identical](https://www.javatpoint.com/java-program-to-determine-whether-two-trees-are-identical)

6) [Java program to find maximum width of a binary tree](https://www.javatpoint.com/java-program-to-find-maximum-width-of-a-binary-tree)

7) [Java program to find the largest element in a Binary Tree](https://www.javatpoint.com/java-program-to-find-the-largest-element-in-a-binary-tree)

8) [Java program to find the maximum depth or height of a tree](https://www.javatpoint.com/java-program-to-find-the-maximum-depth-or-height-of-a-tree)

9) [Java program to find the nodes which are at the maximum distance in a Binary Tree](https://www.javatpoint.com/java-program-to-find-the-nodes-which-are-at-the-maximum-distance-in-a-binary-tree)

10) [Java program to find the smallest element in a tree](https://www.javatpoint.com/java-program-to-find-the-smallest-element-in-a-tree)

11) [Java program to find the sum of all the nodes of a binary tree](https://www.javatpoint.com/java-program-to-find-the-sum-of-all-the-nodes-of-a-binary-tree)

12) [Java program to find the total number of possible Binary Search Trees with N keys](https://www.javatpoint.com/java-program-to-find-the-total-number-of-possible-binary-search-trees-with-n-keys)

13) [Java program to implement Binary Tree using the Linked List](https://www.javatpoint.com/java-program-to-implement-binary-tree-using-the-linked-list)

14) [Java program to search a node in a Binary Tree](https://www.javatpoint.com/java-program-to-search-a-node-in-a-binary-tree)