Java Control Statements | Control Flow in Java

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.

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.

Loop Statements

In programming, sometimes we need to execute the block of code repeatedly while some condition evaluates to true. However, loop statements are used to execute the set of instructions in a repeated order. The execution of the set of instructions depends upon a particular condition.

In Java, we have three types of loops that execute similarly. However, there are differences in their syntax and condition checking time.

1. for loop
2. while loop
3. do-while loop

Let's understand the loop statements one by one.

Java for loop

In Java, [for loop](https://www.javatpoint.com/java-for-loop) is similar to [C](https://www.javatpoint.com/c-programming-language-tutorial) and [C++](https://www.javatpoint.com/cpp-tutorial). It enables us to initialize the loop variable, check the condition, and increment/decrement in a single line of code. We use the for loop only when we exactly know the number of times, we want to execute the block of code.

1. **for**(initialization, condition, increment/decrement) {
2. //block of statements
3. }

The flow chart for the for-loop is given below.



Consider the following example to understand the proper functioning of the for loop in java.

**Calculation.java**

1. **public** **class** Calculattion {
2. **public** **static** **void** main(String[] args) {
3. // TODO Auto-generated method stub
4. **int** sum = 0;
5. **for**(**int** j = 1; j<=10; j++) {
6. sum = sum + j;
7. }
8. System.out.println("The sum of first 10 natural numbers is " + sum);
9. }
10. }

**Output:**

The sum of first 10 natural numbers is 55

Java for-each loop

Java provides an enhanced for loop to traverse the data structures like array or collection. In the for-each loop, we don't need to update the loop variable. The syntax to use the for-each loop in java is given below.

1. **for**(data\_type var : array\_name/collection\_name){
2. //statements
3. }

Consider the following example to understand the functioning of the for-each loop in Java.

**Calculation.java**

1. **public** **class** Calculation {
2. **public** **static** **void** main(String[] args) {
3. // TODO Auto-generated method stub
4. String[] names = {"Java","C","C++","Python","JavaScript"};
5. System.out.println("Printing the content of the array names:\n");
6. **for**(String name:names) {
7. System.out.println(name);
8. }
9. }
10. }

**Output:**

Printing the content of the array names:

Java

C

C++

Python

JavaScript

Java while loop

The [while loop](https://www.javatpoint.com/java-while-loop) is also used to iterate over the number of statements multiple times. However, if we don't know the number of iterations in advance, it is recommended to use a while loop. Unlike for loop, the initialization and increment/decrement doesn't take place inside the loop statement in while loop.

It is also known as the entry-controlled loop since the condition is checked at the start of the loop. If the condition is true, then the loop body will be executed; otherwise, the statements after the loop will be executed.

The syntax of the while loop is given below.

1. **while**(condition){
2. //looping statements
3. }

The flow chart for the while loop is given in the following image.



Consider the following example.

**Calculation .java**

1. **public** **class** Calculation {
2. **public** **static** **void** main(String[] args) {
3. // TODO Auto-generated method stub
4. **int** i = 0;
5. System.out.println("Printing the list of first 10 even numbers \n");
6. **while**(i<=10) {
7. System.out.println(i);
8. i = i + 2;
9. }
10. }
11. }

**Output:**

Printing the list of first 10 even numbers

0

2

4

6

8

10

Java do-while loop

The [do-while loop](https://www.javatpoint.com/java-do-while-loop) checks the condition at the end of the loop after executing the loop statements. When the number of iteration is not known and we have to execute the loop at least once, we can use do-while loop.

It is also known as the exit-controlled loop since the condition is not checked in advance. The syntax of the do-while loop is given below.

1. **do**
2. {
3. //statements
4. } **while** (condition);

The flow chart of the do-while loop is given in the following image.



Consider the following example to understand the functioning of the do-while loop in Java.

**Calculation.java**

1. **public** **class** Calculation {
2. **public** **static** **void** main(String[] args) {
3. // TODO Auto-generated method stub
4. **int** i = 0;
5. System.out.println("Printing the list of first 10 even numbers \n");
6. **do** {
7. System.out.println(i);
8. i = i + 2;
9. }**while**(i<=10);
10. }
11. }

**Output:**

Printing the list of first 10 even numbers

0

2

4

6

8

10

Jump Statements

Jump statements are used to transfer the control of the program to the specific statements. In other words, jump statements transfer the execution control to the other part of the program. There are two types of jump statements in Java, i.e., break and continue.

Java break statement

As the name suggests, the [break statement](https://www.javatpoint.com/java-break) is used to break the current flow of the program and transfer the control to the next statement outside a loop or switch statement. However, it breaks only the inner loop in the case of the nested loop.

The break statement cannot be used independently in the Java program, i.e., it can only be written inside the loop or switch statement.

**The break statement example with for loop**

Consider the following example in which we have used the break statement with the for loop.

**BreakExample.java**

1. **public** **class** BreakExample {
3. **public** **static** **void** main(String[] args) {
4. // TODO Auto-generated method stub
5. **for**(**int** i = 0; i<= 10; i++) {
6. System.out.println(i);
7. **if**(i==6) {
8. **break**;
9. }
10. }
11. }
12. }

**Output:**

0

1

2

3

4

5

6

**break statement example with labeled for loop**

**Calculation.java**

1. **public** **class** Calculation {
3. **public** **static** **void** main(String[] args) {
4. // TODO Auto-generated method stub
5. a:
6. **for**(**int** i = 0; i<= 10; i++) {
7. b:
8. **for**(**int** j = 0; j<=15;j++) {
9. c:
10. **for** (**int** k = 0; k<=20; k++) {
11. System.out.println(k);
12. **if**(k==5) {
13. **break** a;
14. }
15. }
16. }
18. }
19. }

22. }

**Output:**

0

1

2

3

4

5

Java continue statement

Unlike break statement, the [continue statement](https://www.javatpoint.com/java-continue) doesn't break the loop, whereas, it skips the specific part of the loop and jumps to the next iteration of the loop immediately.

Consider the following example to understand the functioning of the continue statement in Java.

1. **public** **class** ContinueExample {
3. **public** **static** **void** main(String[] args) {
4. // TODO Auto-generated method stub
6. **for**(**int** i = 0; i<= 2; i++) {
8. **for** (**int** j = i; j<=5; j++) {
10. **if**(j == 4) {
11. **continue**;
12. }
13. System.out.println(j);
14. }
15. }
16. }
18. }

**Output:**

0

1

2

3

5

1

2

3

5

2

3

5

# Java If-else Statement

The [Java](https://www.javatpoint.com/java-tutorial) if statement is used to test the condition. It checks [boolean](https://www.javatpoint.com/boolean-keyword-in-java) 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:**

1. **if**(condition){
2. //code to be executed
3. }



**Example:**

1. //Java Program to demonstate the use of if statement.
2. **public** **class** IfExample {
3. **public** **static** **void** main(String[] args) {
4. //defining an 'age' variable
5. **int** age=20;
6. //checking the age
7. **if**(age>18){
8. System.out.print("Age is greater than 18");
9. }
10. }
11. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=IfExample)

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:**

1. **if**(condition){
2. //code if condition is true
3. }**else**{
4. //code if condition is false
5. }



**Example:**

1. //A Java Program to demonstrate the use of if-else statement.
2. //It is a program of odd and even number.
3. **public** **class** IfElseExample {
4. **public** **static** **void** main(String[] args) {
5. //defining a variable
6. **int** number=13;
7. //Check if the number is divisible by 2 or not
8. **if**(number%2==0){
9. System.out.println("even number");
10. }**else**{
11. System.out.println("odd number");
12. }
13. }
14. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=IfElseExample)

Output:

odd number

**Leap Year Example:**

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

1. **public** **class** LeapYearExample {
2. **public** **static** **void** main(String[] args) {
3. **int** year=2020;
4. **if**(((year % 4 ==0) && (year % 100 !=0)) || (year % 400==0)){
5. System.out.println("LEAP YEAR");
6. }
7. **else**{
8. System.out.println("COMMON YEAR");
9. }
10. }
11. }

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:**

1. **public** **class** IfElseTernaryExample {
2. **public** **static** **void** main(String[] args) {
3. **int** number=13;
4. //Using ternary operator
5. String output=(number%2==0)?"even number":"odd number";
6. System.out.println(output);
7. }
8. }

Output:

odd number

## **Java if-else-if ladder Statement**

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

**Syntax:**

1. **if**(condition1){
2. //code to be executed if condition1 is true
3. }**else** **if**(condition2){
4. //code to be executed if condition2 is true
5. }
6. **else** **if**(condition3){
7. //code to be executed if condition3 is true
8. }
9. ...
10. **else**{
11. //code to be executed if all the conditions are false
12. }



**Example:**

1. //Java Program to demonstrate the use of If else-if ladder.
2. //It is a program of grading system for fail, D grade, C grade, B grade, A grade and A+.
3. **public** **class** IfElseIfExample {
4. **public** **static** **void** main(String[] args) {
5. **int** marks=65;
7. **if**(marks<50){
8. System.out.println("fail");
9. }
10. **else** **if**(marks>=50 && marks<60){
11. System.out.println("D grade");
12. }
13. **else** **if**(marks>=60 && marks<70){
14. System.out.println("C grade");
15. }
16. **else** **if**(marks>=70 && marks<80){
17. System.out.println("B grade");
18. }
19. **else** **if**(marks>=80 && marks<90){
20. System.out.println("A grade");
21. }**else** **if**(marks>=90 && marks<100){
22. System.out.println("A+ grade");
23. }**else**{
24. System.out.println("Invalid!");
25. }
26. }
27. }

Output:

C grade

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

1. **public** **class** PositiveNegativeExample {
2. **public** **static** **void** main(String[] args) {
3. **int** number=-13;
4. **if**(number>0){
5. System.out.println("POSITIVE");
6. }**else** **if**(number<0){
7. System.out.println("NEGATIVE");
8. }**else**{
9. System.out.println("ZERO");
10. }
11. }
12. }

Output:

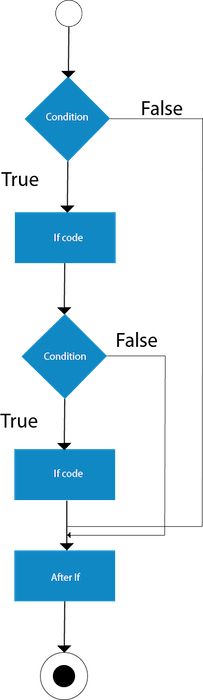
NEGATIVE

## **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:**

1. **if**(condition){
2. //code to be executed
3. **if**(condition){
4. //code to be executed
5. }
6. }



**Example:**

1. //Java Program to demonstrate the use of Nested If Statement.
2. **public** **class** JavaNestedIfExample {
3. **public** **static** **void** main(String[] args) {
4. //Creating two variables for age and weight
5. **int** age=20;
6. **int** weight=80;
7. //applying condition on age and weight
8. **if**(age>=18){
9. **if**(weight>50){
10. System.out.println("You are eligible to donate blood");
11. }
12. }
13. }}

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=JavaNestedIfExample)

Output:

You are eligible to donate blood

**Example 2:**

1. //Java Program to demonstrate the use of Nested If Statement.
2. **public** **class** JavaNestedIfExample2 {
3. **public** **static** **void** main(String[] args) {
4. //Creating two variables for age and weight
5. **int** age=25;
6. **int** weight=48;
7. //applying condition on age and weight
8. **if**(age>=18){
9. **if**(weight>50){
10. System.out.println("You are eligible to donate blood");
11. } **else**{
12. System.out.println("You are not eligible to donate blood");
13. }
14. } **else**{
15. System.out.println("Age must be greater than 18");
16. }
17. }  }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=JavaNestedIfExample2)

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](https://www.javatpoint.com/java-if-else) 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](https://www.javatpoint.com/java-string) in the switch statement.

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

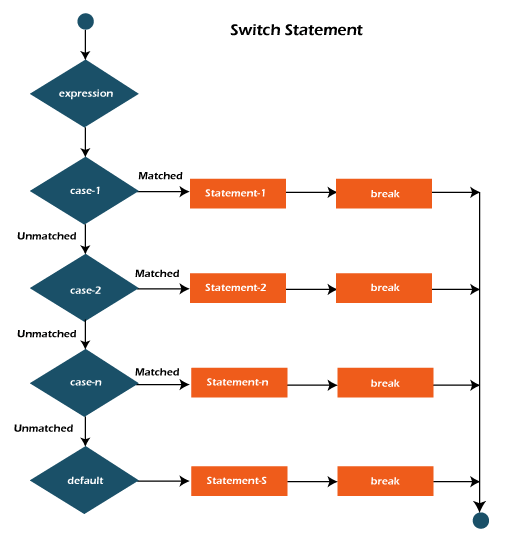
### Points to Remember

* 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](https://www.javatpoint.com/java-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](https://www.javatpoint.com/java-switch)* and string.
* Each case statement can have a break statement which is optional. When control reaches to the [break statement](https://www.javatpoint.com/java-break), 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:**

1. **switch**(expression){
2. **case** value1:
3. //code to be executed;
4. **break**;  //optional
5. **case** value2:
6. //code to be executed;
7. **break**;  //optional
8. ......
10. **default**:
11. code to be executed **if** all cases are not matched;
12. }

**Flowchart of Switch Statement**



**Example:**

**SwitchExample.java**

1. **public** **class** SwitchExample {
2. **public** **static** **void** main(String[] args) {
3. //Declaring a variable for switch expression
4. **int** number=20;
5. //Switch expression
6. **switch**(number){
7. //Case statements
8. **case** 10: System.out.println("10");
9. **break**;
10. **case** 20: System.out.println("20");
11. **break**;
12. **case** 30: System.out.println("30");
13. **break**;
14. //Default case statement
15. **default**:System.out.println("Not in 10, 20 or 30");
16. }
17. }
18. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=SwitchExample)

**Output:**

20

**Finding Month Example:**

**SwitchMonthExample.javaHTML**

1. //Java Program to demonstrate the example of Switch statement
2. //where we are printing month name for the given number
3. **public** **class** SwitchMonthExample {
4. **public** **static** **void** main(String[] args) {
5. //Specifying month number
6. **int** month=7;
7. String monthString="";
8. //Switch statement
9. **switch**(month){
10. //case statements within the switch block
11. **case** 1: monthString="1 - January";
12. **break**;
13. **case** 2: monthString="2 - February";
14. **break**;
15. **case** 3: monthString="3 - March";
16. **break**;
17. **case** 4: monthString="4 - April";
18. **break**;
19. **case** 5: monthString="5 - May";
20. **break**;
21. **case** 6: monthString="6 - June";
22. **break**;
23. **case** 7: monthString="7 - July";
24. **break**;
25. **case** 8: monthString="8 - August";
26. **break**;
27. **case** 9: monthString="9 - September";
28. **break**;
29. **case** 10: monthString="10 - October";
30. **break**;
31. **case** 11: monthString="11 - November";
32. **break**;
33. **case** 12: monthString="12 - December";
34. **break**;
35. **default**:System.out.println("Invalid Month!");
36. }
37. //Printing month of the given number
38. System.out.println(monthString);
39. }
40. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=SwitchMonthExample)

**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.

**SwitchVowelExample.java**

1. **public** **class** SwitchVowelExample {
2. **public** **static** **void** main(String[] args) {
3. **char** ch='O';
4. **switch**(ch)
5. {
6. **case** 'a':
7. System.out.println("Vowel");
8. **break**;
9. **case** 'e':
10. System.out.println("Vowel");
11. **break**;
12. **case** 'i':
13. System.out.println("Vowel");
14. **break**;
15. **case** 'o':
16. System.out.println("Vowel");
17. **break**;
18. **case** 'u':
19. System.out.println("Vowel");
20. **break**;
21. **case** 'A':
22. System.out.println("Vowel");
23. **break**;
24. **case** 'E':
25. System.out.println("Vowel");
26. **break**;
27. **case** 'I':
28. System.out.println("Vowel");
29. **break**;
30. **case** 'O':
31. System.out.println("Vowel");
32. **break**;
33. **case** 'U':
34. System.out.println("Vowel");
35. **break**;
36. **default**:
37. System.out.println("Consonant");
38. }
39. }
40. }

**Output:**

Vowel

## **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:**

**SwitchExample2.java**

1. //Java Switch Example where we are omitting the
2. //break statement
3. **public** **class** SwitchExample2 {
4. **public** **static** **void** main(String[] args) {
5. **int** number=20;
6. //switch expression with int value
7. **switch**(number){
8. //switch cases without break statements
9. **case** 10: System.out.println("10");
10. **case** 20: System.out.println("20");
11. **case** 30: System.out.println("30");
12. **default**:System.out.println("Not in 10, 20 or 30");
13. }
14. }
15. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=SwitchExample2)

**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:**

**SwitchStringExample.java**

1. //Java Program to demonstrate the use of Java Switch
2. //statement with String
3. **public** **class** SwitchStringExample {
4. **public** **static** **void** main(String[] args) {
5. //Declaring String variable
6. String levelString="Expert";
7. **int** level=0;
8. //Using String in Switch expression
9. **switch**(levelString){
10. //Using String Literal in Switch case
11. **case** "Beginner": level=1;
12. **break**;
13. **case** "Intermediate": level=2;
14. **break**;
15. **case** "Expert": level=3;
16. **break**;
17. **default**: level=0;
18. **break**;
19. }
20. System.out.println("Your Level is: "+level);
21. }
22. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=SwitchStringExample)

**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:**

**NestedSwitchExample.java**

1. //Java Program to demonstrate the use of Java Nested Switch
2. **public** **class** NestedSwitchExample {
3. **public** **static** **void** main(String args[])
4. {
5. //C - CSE, E - ECE, M - Mechanical
6. **char** branch = 'C';
7. **int** collegeYear = 4;
8. **switch**( collegeYear )
9. {
10. **case** 1:
11. System.out.println("English, Maths, Science");
12. **break**;
13. **case** 2:
14. **switch**( branch )
15. {
16. **case** 'C':
17. System.out.println("Operating System, Java, Data Structure");
18. **break**;
19. **case** 'E':
20. System.out.println("Micro processors, Logic switching theory");
21. **break**;
22. **case** 'M':
23. System.out.println("Drawing, Manufacturing Machines");
24. **break**;
25. }
26. **break**;
27. **case** 3:
28. **switch**( branch )
29. {
30. **case** 'C':
31. System.out.println("Computer Organization, MultiMedia");
32. **break**;
33. **case** 'E':
34. System.out.println("Fundamentals of Logic Design, Microelectronics");
35. **break**;
36. **case** 'M':
37. System.out.println("Internal Combustion Engines, Mechanical Vibration");
38. **break**;
39. }
40. **break**;
41. **case** 4:
42. **switch**( branch )
43. {
44. **case** 'C':
45. System.out.println("Data Communication and Networks, MultiMedia");
46. **break**;
47. **case** 'E':
48. System.out.println("Embedded System, Image Processing");
49. **break**;
50. **case** 'M':
51. System.out.println("Production Technology, Thermal Engineering");
52. **break**;
53. }
54. **break**;
55. }
56. }
57. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=NestedSwitchExample)

**Output:**

Data Communication and Networks, MultiMedia

## **Java Enum in Switch Statement**

Java allows us to use enum in switch statement. Java enum is a class that represent the group of constants. (immutable such as final variables). We use the keyword enum and put the constants in curly braces separated by comma.

**Example:**

**JavaSwitchEnumExample.java**

1. //Java Program to demonstrate the use of Enum
2. //in switch statement
3. **public** **class** JavaSwitchEnumExample {
4. **public** **enum** Day {  Sun, Mon, Tue, Wed, Thu, Fri, Sat  }
5. **public** **static** **void** main(String args[])
6. {
7. Day[] DayNow = Day.values();
8. **for** (Day Now : DayNow)
9. {
10. **switch** (Now)
11. {
12. **case** Sun:
13. System.out.println("Sunday");
14. **break**;
15. **case** Mon:
16. System.out.println("Monday");
17. **break**;
18. **case** Tue:
19. System.out.println("Tuesday");
20. **break**;
21. **case** Wed:
22. System.out.println("Wednesday");
23. **break**;
24. **case** Thu:
25. System.out.println("Thursday");
26. **break**;
27. **case** Fri:
28. System.out.println("Friday");
29. **break**;
30. **case** Sat:
31. System.out.println("Saturday");
32. **break**;
33. }
34. }
35. }
36. }

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

**Output:**

Sunday

Monday

Twesday

Wednesday

Thursday

Friday

Saturday

## **Java Wrapper in Switch Statement**

Java allows us to use four [wrapper classes](https://www.javatpoint.com/wrapper-class-in-java): Byte, Short, Integer and Long in switch statement.

**Example:**

**WrapperInSwitchCaseExample.java**

1. //Java Program to demonstrate the use of Wrapper class
2. //in switch statement
3. **public** **class** WrapperInSwitchCaseExample {
4. **public** **static** **void** main(String args[])
5. {
6. Integer age = 18;
7. **switch** (age)
8. {
9. **case** (16):
10. System.out.println("You are under 18.");
11. **break**;
12. **case** (18):
13. System.out.println("You are eligible for vote.");
14. **break**;
15. **case** (65):
16. System.out.println("You are senior citizen.");
17. **break**;
18. **default**:
19. System.out.println("Please give the valid age.");
20. **break**;
21. }
22. }
23. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=WrapperInSwitchCaseExample)

**Output:**

You are eligible for vote.