



Control Statements in Java



OBJECTIVES OF THIS CHAPTER

- 6.1 Introduction
- 6.2 Branching Statements: if else, switch case
- 6.3 Looping Statements: for, while, do while
- 6.4 Jumping Statements: break, continue
- 6.5 Type Conversion

6.1 INTRODUCTION

The statements inside our source code files are generally executed from top to bottom, in the order they appear. Control statements, however, break up the flow of execution by employing decision making, looping and jumping statements which enables our program to conditionally execute specific blocks of code. This chapter describes the decision-making statements (if, if-else, switch case etc.), the looping statements (for, while, do-while), and the jumping statements (break, continue, return) supported by Java programming language.

Whenever we want to change the execution flow of statements, we can use Control Statements in the program. These statements change the execution flow based on some test condition. Based on different ways of control flows, Control Statements can be divided into three categories, namely:

1. Conditional (Decision Making) Statements
2. Looping (Iterative) Statements
3. Jumping Statements

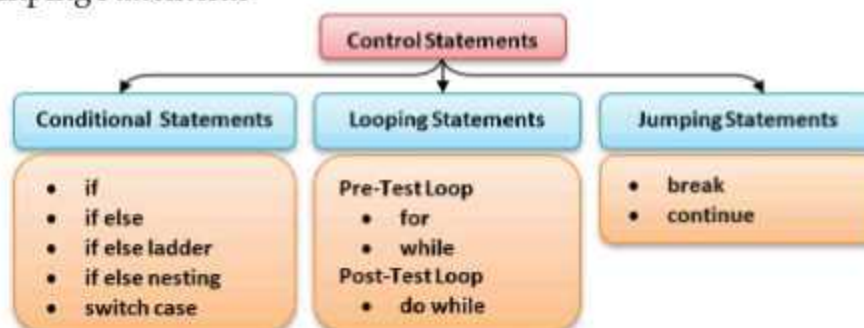


Fig: 6.1 Types of Control Statements

Now, let's begin explaining these statements in detail.

6.2 CONDITIONAL (DECISION MAKING) STATEMENTS

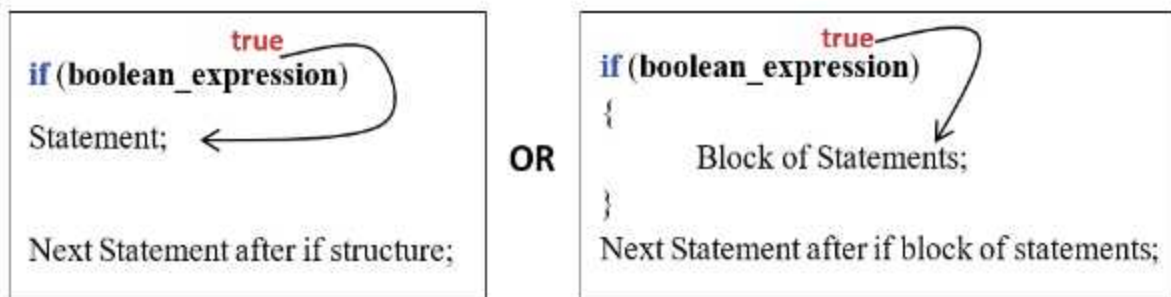
These statements can be used for decision – making purpose or for making multi-way selection. We can make decisions for executing the statements based upon the result of a condition. These statements evaluate the test-condition (Boolean Expression) to control the execution flow in the program. Result of condition determines which block of statements will be executed. That is why these statements are also known as Decision-Making statements. These statements are also called branching statements because the program chooses to follow one branch or another during execution. Various conditional flow statements in java are given following:

1. Only if statement
2. if-else statement
3. if-else ladder statement
4. if-else nesting statement
5. switch case statement

6.2.1 Only if Statement:

It is the simplest form of if else statement. The 'if statement' evaluates the `boolean_expression` (condition) and then proceed to carry out the set of actions only if the test condition is evaluated to true. It is terminated when the test condition evaluates to false. The syntax for using this statement is given below:

Syntax:



Here, the `boolean_expression` also referred so as condition must be enclosed in parenthesis, which causes the `boolean_expression` (test condition) to be evaluated first. Most often, the expression used to control the if, will involve the relational OR/ AND logical operators. If this expression is evaluated true, then the “Block of Statements” will be executed otherwise the “Block of Statements” connected with if statement will be ignored and control will be passed to the next statement after if block of statements. Consider the following program example:

Program 6.1: Write a program in Java to show "Excellent Score!!" message to the student only if marks scored by the student are above 80%.

```
1 class cs1
2 {
3     public static void main(String args[])
4     {
5         int math=85, sci=90;
6         int total=math+sci;
7         double per=total/200.0*100.0;
8
9         if(per>80)
10        {
11            System.out.println("Excellent Score!!");
12        }
13        System.out.println("Your Percentage Marks are: "+per);
14    }
15 }
```

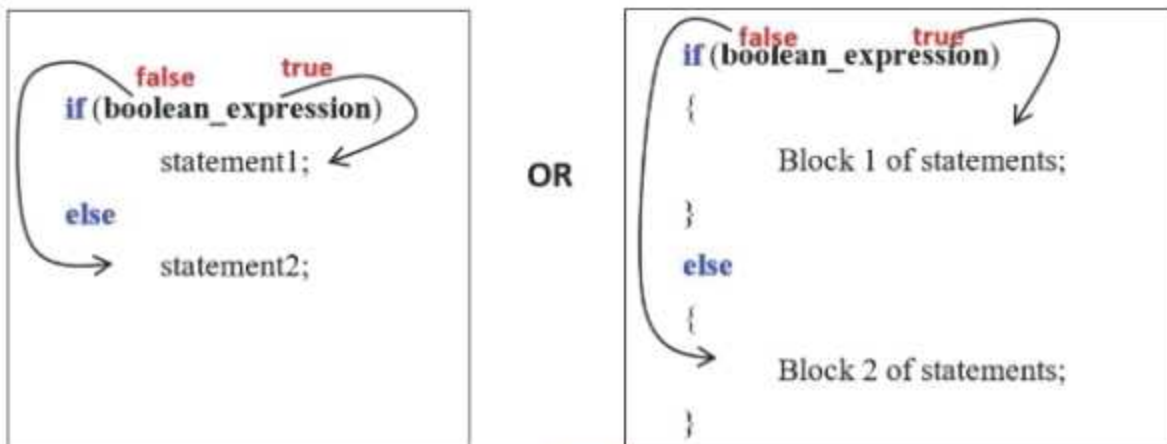
Compilation, Execution and Output of Program 6.1:

```
D:\Java Programs>javac cs1.java
D:\Java Programs>java cs1
Excellent Score!!
Your Percentage Marks are: 87.5
D:\Java Programs>
```

6.2.2 if else Statement:

In this conditional control statement, "statement" or "block of statements" associated with **if** statement gets executed only when the given **boolean_expression** (test condition) is evaluated to **true** while the statements in **else** block gets executed only when the **boolean_expression** (test condition) returns **false**. The syntax for using if else statement is given below:

Syntax:



As shown in the above syntax, If the **boolean_expression** (condition) is **true**, then "statement1" or "Block 1 of statements" is executed. Otherwise (If **boolean_expression** is **false**), "statement2" or "Block 2 of statements" is executed. In no case, both statements will be executed.

Program 6.2: Write a program to find whether the student is "Pass" or "Fail" according to the value of percentage marks.

```
1 class cs2
2 {
3     public static void main(String args[])
4     {
5         int math=35, sci=50;
6         int total=math+sci;
7         double per=total/200.0*100.0;
8         if(per>35)
9         {
10            System.out.println("Congrates!! You are Pass");
11        }
12        else
13        {
14            System.out.println("Sorry!! You are Fail");
15        }
16        System.out.println("Your Percentage Marks are: "+per);
17    }
18 }
19 }
```

Compilation, Execution and Output of Program 6.2:

```
C:\Programs>javac cs2.java
D:\Java Programs>java cs2
Congrates!! You are Pass
Your Percentage Marks are: 42.5
D:\Java Programs>_
```

6.2.3 if else ladder Statement:

It is a chain of multiple if-else statements. It is used to execute the if-else condition (**boolean_expression**) where each else part has associated set of if-else statement. For each else part, we need to evaluate another set of if-else conditions. Here, a particular "statement" or "block of statements" get executed when the corresponding condition is true. Statements in final else block gets executed when all other conditions are false. We can use any number of else if blocks between if and else.

Syntax:

```
if (boolean_expression1)
```

```
statement 1;
```

```
else if (boolean_expression2)
```

```
statement 2;
```

```
.....
```

```
.....
```

```
else
```

```
statement n;
```

OR

```
if (boolean_expression1)
```

```
{
```

```
Block 1 of statements;
```

```
}
```

```
else if (boolean_expression2)
```

```
{
```

```
Block 2 of statements;
```

```
}
```

```
.....
```

```
.....
```

```
else
```

```
{
```

```
Block n of statements;
```

```
}
```

Program 6.3: Write a program to find Grade of the student If marks \geq 80 then Grade A, if marks \geq 60 and marks $<$ 80 then Grade B, if marks \geq 40 and marks $<$ 60 then Grade C, otherwise Grade D.

```
1 class cs3
2 {
3     public static void main(String ar[])
4     {
5         int math=35, sci=50;
6         int total=math+sci;
7         double per=total/200.0*100.0;
8         if(per>=80)
9             System.out.println("Grade A");
10        else if(per>=60)
11            System.out.println("Grade B");
12        else if(per>=40)
13            System.out.println("Grade C");
14        else
15            System.out.println("Grade D");
16
17        System.out.println("Your Percentage Marks are: "+per);
18    }
19 }
```

Compilation, Execution and Output of Program 6.3:

```
C:\Windows\system32\cmd.exe
D:\Java Programs>javac cs3.java

D:\Java Programs>java cs3
Grade C
Your Percentage Marks are: 42.5
```

6.2.4 if-else nesting statement:

if else nesting is an if or if-else statement that is the target of another if or else or if-else. The nested if-else statements include multiple if and/or if-else statements. In other words, we can say that writing the **if** or **if-else statement with-in another if or else or if-else** is called as if-else nesting statement. Inner if or if-else is executed after the evaluation of outer boolean_expression (condition).

Syntax:

<pre>if(boolean_expression1) { if(boolean_expression2) { Block 1 of Statements; } else { Block 2 of Statements; } } else { Block 3 of statements; }</pre>	<pre>if(boolean_expression1) { Block 1 of Statements; } else { if(boolean_expression2) { Block 2 of Statements; } else { Block 3 of Statements; } }</pre>	<pre>if(boolean_expression1) { if(boolean_expression2) { Block 1 of Statements; } else { Block 2 of Statements; } } else { if(boolean_expression3) { Block 3 of Statements; } else { Block 4 of Statements; } }</pre>
---	---	---

Program 6.4: Write a program to find largest of three numbers.

```
1 class CS4 {
2     public static void main(String args[])
3     {
4         int a=15,b=56,c=34;
5         if(a>b)
6         {
7             if(a>c)
8                 System.out.println("Largest Number is:" + a);
9             else
10                System.out.println("Largest Number is:" + c);
11        }
12        else
13        {
14            if(b>c)
15                System.out.println("Largest Number is:" + b);
16            else
17                System.out.println("Largest Number is:" + c);
18        }
19    }
20 }
```

Compilation, Execution and Output of Program 6.4:

```
D:\Java Programs>javac cs4.java

D:\Java Programs>java cs4
Largest Number is:56

D:\Java Programs>_
```

6.2.5 switch-case Statement:

It is a multi-way branching statement. The statement switch-case is similar to if-else ladder. It is a matter of preference which we use. Switch statement can be slightly more efficient and easier to read.

switch-case statement provides the most efficient method of passing control to one of the different labels of our code based on the value of an expression. The expression must be of type byte, short, int, String or char. The values specified in the case statements must be of a type compatible with the expression. Each case value must be a constant, not a variable. Duplicate case values are not allowed.

Syntax:

```
switch (expression)
{
    case constant1:
        statements1; break;
    case constant2:
        statements2; break;
    ...
    ...
    case constant n:
        statements n; break;
    default:
        statements;
}
```


The value of the expression is compared with each of the constant values in the case statements. If a match is found, the code sequence following that case statement is executed. If none of the constants matches the value of the expression, then the default statement is executed. However, the default statement is optional. If no case matches and no default is present, then no action is taken by switch statement.

The break statement is used inside the switch to terminate a statement sequence. When a break statement is encountered, execution branches to the first line of code that follows the entire switch statement.

Program 6.5: Write a program using switch case to print the name of the day corresponding to the number of the day specified by the user.

```
1 class cs5
2 {
3     public static void main(String args[])
4     {
5         int day=2;
6         switch(day)
7         {
8             case 1: System.out.println("1st Day is Monday"); break;
9             case 2: System.out.println("2nd Day is Tuesday"); break;
10            case 3: System.out.println("3rd Day is Wednesday"); break;
11            case 4: System.out.println("4th Day is Thursday"); break;
12            case 5: System.out.println("5th Day is Friday"); break;
13            case 6: System.out.println("6th Day is Saturday"); break;
14            case 7: System.out.println("7th Day is Sunday"); break;
15            default: System.out.println("Wrong value of the day");
16        }
17    }
18 }
```

Compilation, Execution and Output of Program 6.5

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac cs5.java
D:\Java Programs>java cs5
2nd Day is Tuesday
D:\Java Programs>java cs5_
```

6.3 LOOPING CONTROL STATEMENTS

Looping statements are also called **Iterative** Statements. There may be situations when we need to execute a block of statements several number of times. In such situations, loops provide a way to repeat statements. A loop repeatedly executes the same set of instructions until a termination condition is met. Java loops can be categorized into two categories as given following:

6.3.1 Pre-Test Loops

- ❖ for loop
- ❖ while loop

6.3.2 Post-Test Loop

- ❖ do-while loop

Any loop usually consists of following three expressions along with the body of the loop that contains statements to be repeated:

- ❖ **Expression1** includes initialization of counter variable that controls the loop
- ❖ **Expression2** includes boolean_expression that defines the termination condition of loop
- ❖ **Expression3** includes step value which can be represented by increasing or decreasing the value of counter variable.

6.3.1 Pre-Test Loops:

Pre-Test loops are also called Entry-Controlled loops. In these loops, the control conditions are tested before execution of the body of loop. The 'for' and 'while' loops are the entry-controlled loops in Java. These loops are explained below:

for loop: A for loop is perhaps the most commonly used form of looping. This pre-test iterative control structure allows us to write a loop that needs to be executed for a specific number of times. The 'for loop' statements get executed as long as condition is true.

Syntax:

```
for(initialization;boolean_expression;step)
{
    Body of loop;
}
```

If only one statement is being repeated, there is no need for the curly braces.

Working of 'for' loop:

1. When the loop first starts, the initialization portion of the loop is executed. Generally, this is an expression that sets the value of the loop control variable, which acts as a counter that controls the loop. It is important to understand that the initialization expression is only executed once.
2. Next, condition is evaluated. This must be a Boolean expression. It usually tests the loop control variable against a target value. If this expression is true, then the body of the loop is executed. If it is false, the loop terminates.
3. Next, the step portion of the loop is executed. This is usually an expression that increments or decrements the loop control variable.

4. The loop then iterates, first evaluating the conditional expression, then executing the body of the loop, and then executing the iteration expression with each pass. This process repeats until the controlling expression is false.

Program 6.6: Write a program to print first n natural numbers using for loop.

```
cs6.java
1 class cs6
2 {
3     public static void main(String args[])
4     {
5         int i,n;
6         n=7;
7         for(i=1;i<=n;i++)
8         {
9             System.out.println("i=" + i);
10        }
11    }
12 }
```

Compilation, Execution and Output of Program 6.6:

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac cs6.java

D:\Java Programs>java cs6
i=1
i=2
i=3
i=4
i=5
i=6
i=7

D:\Java Programs>
```

While loop:

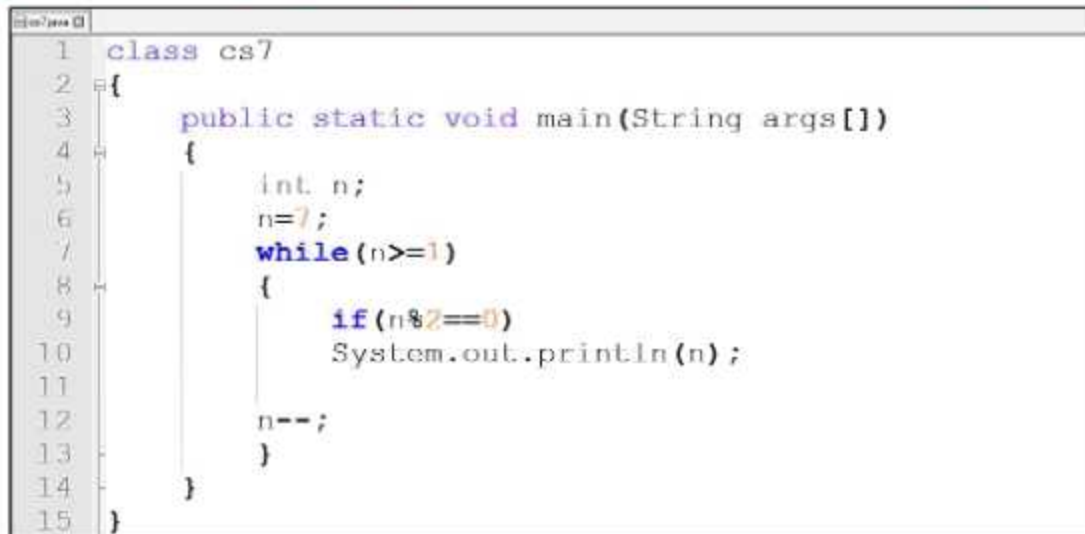
It is also a pre-test loop. This loop checks the boolean_expression (condition) before the execution of its body. The body of the loop will be executed as long as the conditional expression is true. When condition becomes false, control passes to the next line of code immediately following the loop. if the conditional expression controlling a while loop is initially false, then the body of the loop will not be executed at all. So, minimum number

of executions of the body of while loop is zero. The general syntax of for loop is given below:

Syntax:

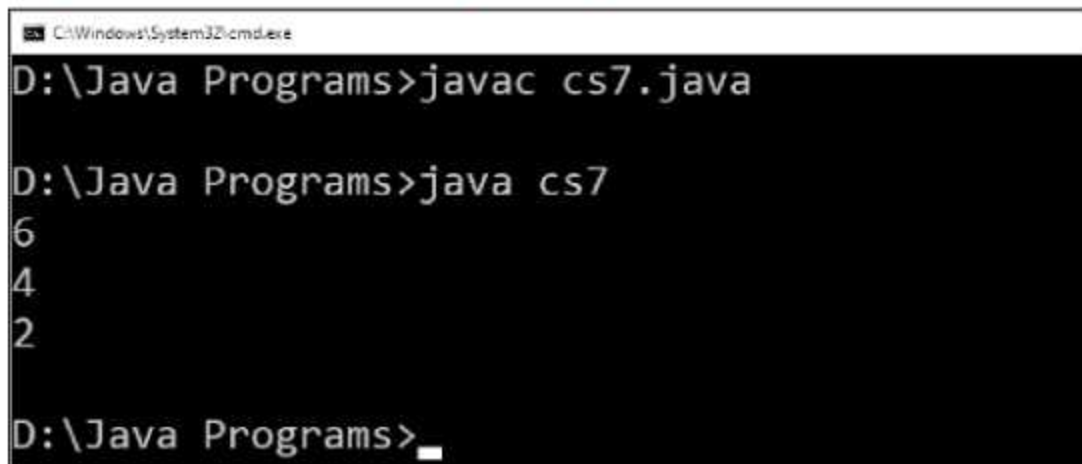
```
while(boolean_expression)
{
    Body of loop;
}
```

Program 6.7: Write a program to print even numbers between n to 1 using while loop.



```
1 class cs7
2 {
3     public static void main(String args[])
4     {
5         int n;
6         n=7;
7         while (n>=1)
8         {
9             if (n%2==0)
10                System.out.println(n);
11
12            n--;
13        }
14    }
15 }
```

Compilation, Execution and Output of Program 6.7:



```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac cs7.java

D:\Java Programs>java cs7
6
4
2

D:\Java Programs>_
```

6.3.2 Post-Test Loop:

Sometimes it is desirable to execute the body of a loop at least once, even if the conditional expression is false. In other words, there are times when we would like to test the termination expression at the end of the loop rather than at the beginning. In such situations, Post Test loops are used. Post-Test loop is also known as Exit-Controlled loop. The 'do while' loop is the only one exit-controlled loop in Java.

do while loop:

In do-while loop, statements get executed as long as condition is true. The do-while loop always executes its body at least once, because its conditional expression is at the bottom of the loop. Therefore, in do while loop, the minimum number of executions for the body of the loop will be one. Each iteration of the do-while loop first executes the body of the loop and then evaluates the conditional expression. If this expression is true, the loop will repeat. Otherwise, the loop terminates. General syntax of do while is as follows:

Syntax:

```
do
{
    ....
    statements;
    ....
} while(boolean_expression);
```

The sole difference between while and do-while is that the statement of the do-while always executes at least once, even if the expression evaluates to false for the first time. In a while, if the boolean_expression (condition) is false for the first time, the statement never executes. In practice, do-while is less common than while.

Program 6.8: Write a program to print odd numbers from 1 to n using do while loop.

```
1 class cs8
2 {
3     public static void main(String args[])
4     {
5         int i,n;
6         n=7;
7         i=1;
8         do{
9             if(i%2==1)
10                System.out.println(i);
11
12            i++;
13        }while(i<=n);
14    }
15 }
```

Compilation, Execution and Output of Program 6.8:

```

D:\Java Programs>javac cs8.java

D:\Java Programs>java cs8
1
3
5
7

D:\Java Programs>
```

6.4 JUMPING CONTROL STATEMENTS

Jumping statements are also used for altering the normal flow of a program. Using these statements, we can transfer the execution control from one location to some other location in the program. Loops perform a set of repetitive tasks (statement) until condition becomes false but it is sometimes desirable to skip some statements inside loop or terminate the loop immediately without checking the test expression. In such cases, break and continue statements are used. The break statement is also used in switch statement to exit switch statement.

Following are the Jumping control statements built in Java to alter the normal flow of execution in the program:

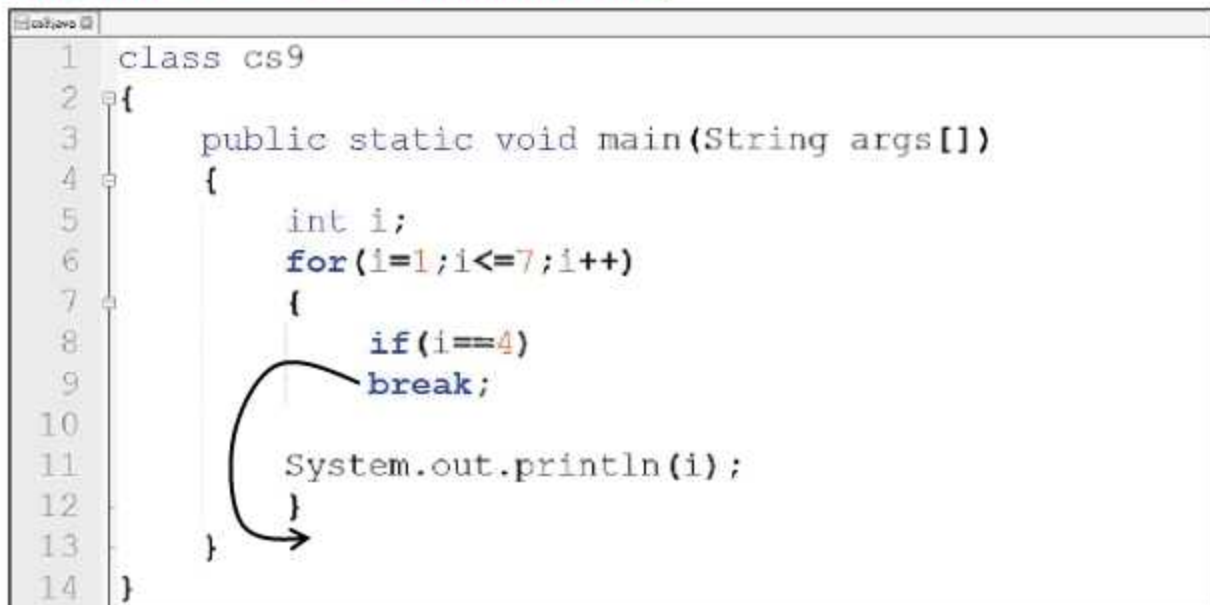
- ❖ break
- ❖ continue

6.4.1 break statement:

The break statement terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch statement. The break statement in Java programming language has the following two usages:

1. When the break statement is encountered inside a loop, the loop is immediately terminated and program execution resumes at the next statement following the loop.
2. It can be used to terminate a case in the switch statement.

Program 6.9: Use of break statement in the loop.



```
1 class cs9
2 {
3     public static void main(String args[])
4     {
5         int i;
6         for(i=1; i<=7; i++)
7         {
8             if(i==4)
9                 break;
10
11             System.out.println(i);
12         }
13     }
14 }
```

The screenshot shows a Java IDE with a file named 'cs9.java'. The code defines a class 'cs9' with a 'main' method. Inside the 'main' method, a 'for' loop iterates from i=1 to i=7. Within the loop, there is an 'if' statement that checks if i is equal to 4. If this condition is true, the 'break' statement is executed, which immediately terminates the loop. An arrow in the image points from the 'break;' statement to the closing brace of the 'for' loop, indicating the exit point. After the loop, the program prints the value of i using 'System.out.println(i);'.

Compilation, Execution and Output of Program 6.9:

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac cs9.java
D:\Java Programs>java cs9
1
2
3
D:\Java Programs>
```

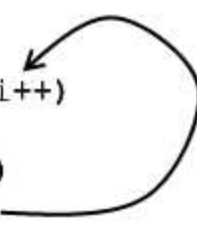
As shown in the program 6.9, Using **break**, we can force immediate termination of a loop, bypassing the conditional expression and any remaining code in the body of the loop. Here, when value of *i* becomes 4, **break** statement terminates the execution of loop and it will take the execution control out of the loop.

6.4.2 continue statement:

It is sometimes desirable to skip some statements inside the loop without exiting the loop. The **continue** statement stops the execution of the current iteration and goes back to the beginning of the loop to begin the next iteration.

Program 6.10: Use of continue statement in the loop.

```
1 class cs10
2 {
3     public static void main(String args[])
4     {
5         int i;
6         for(i=1;i<=7;i++)
7         {
8             if(i%2==0)
9                 continue;
10
11             System.out.println(i);
12         }
13     }
14 }
```



Compilation, Execution and Output of Program 6.10:

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac cs10.java

D:\Java Programs>java cs10
1
3
5
7

D:\Java Programs>_
```

In the program 6.10, When the value of 'i' is divisible by 2 (i.e. when i=2, 4 and 6), the continue statement plays its role and skip their execution but for other values of 'i', the loop will run smoothly.

Points to Remember

1. Whenever we want to change the execution flow of statements, we can use Control Statements in the program.
2. Conditional statements can be used for decision – making purpose or for making multi-way selection.
3. Conditional statements are also called branching statements because the program chooses to follow one branch or another during execution.
4. Writing the **if** or **if-else** statement **with-in** another **if** or **else** or **if-else** is called as if-else nesting statement.
5. switch-case is a multi-way branching statement.
6. switch-case statement provides the most efficient method of passing control to one of the different labels of our code based on the value of an expression.

7. Looping statements are also called **Iterative** Statements.
8. Pre-Test loops are also called Entry-Controlled loops. In these loops, the control conditions are tested before execution of the body of loop.
9. The 'for' and 'while' loops are the entry-controlled loops in Java.
10. for loop is a pre-test iterative control structure that allows us to write a loop that needs to be executed for a specific number of times.
11. Minimum number of executions for the body of while loop is zero.
12. Post-Test loop is also known as Exit-Controlled loop.
13. The 'do while' loop is the only exit-controlled loop in Java.
14. The do-while loop always executes its body at least once.
15. Using Jumping statements, we can transfer the execution control from one location to some other location in the program.
16. break and continue are the jumping statements in Java.
17. The break statement terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch statement.
18. The **continue** statement stops the execution of the current iteration and goes back to the beginning of the loop to begin the next iteration.

Exercise

Q:I Multiple Choice Questions

- i. _____ statements are used to control the execution flow of a program.
 - a. Compound
 - b. Control
 - c. Comparative
 - d. None of these
- ii. _____ statements can be used for decision-making purpose or for multi-way selection.
 - a. Looping
 - b. Iterative
 - c. Conditional
 - d. Jumping
- iii. _____ is a multi-way branching statement.
 - a. switch-case
 - b. if-else
 - c. while
 - d. do-while

- iv. Looping statements are also called _____ Statements.
- | | |
|--------------------|--------------|
| a. Iterative | b. Selection |
| c. Decision-Making | d. Jumping |
- v. Which of the following is an example of Post-Test loop in Java?
- | | |
|-------------|------------------|
| a. for loop | b. while loop |
| c. do-while | d. None of these |
- vi. The 'for' and 'while' loops are the _____ loops in Java.
- | | |
|---------------------|--------------------|
| a. entry-controlled | b. exit-controlled |
| c. Post-Test | d. None of these |
- vii. break and continue are the _____ statements in Java.
- | | |
|--------------------|--------------|
| a. Iterative | b. Selection |
| c. Decision-Making | d. Jumping |

Q: II Fill in the Blanks

- i. Pre-Test loops are also called _____ loops.
- ii. Writing the if or if-else statement with-in another if-else statement is called as _____ statement.
- iii. Minimum number of executions for the body of while loop is _____.
- iv. The _____ statement terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch statement.
- v. The _____ loop always executes its body at least once.
- vi. _____ statement is an example of Jumping Statement.

Q: III Short Answer Type Questions

- i. Define Control Statements.
- ii. Write the name of different types of control statements used in Java.
- iii. What are conditional controls statements in Java?
- iv. Write the syntax of if-else statement in Java.
- v. What do you know about switch-case statement of java.
- vi. What are Iterative Statements?
- vii. What are Pre-Test loops? Give Examples.
- viii. Compare while and do-while loops.
- ix. What are Jumping Statements?

Q: IV Long Answer Type Questions

- i. What are Control Statements? Give an overview of different types of controls statements used in java.
- ii. What are Decision-Making Statements? Explain
- iii. Write a program in Java to find the largest of three values.
- iv. What are looping statements? Explain different categories of looping statements.
- v. Write a program in java to display all even values between 1 to n.
- vi. What are Jumping Statements? Explain the use of break statement in loops.