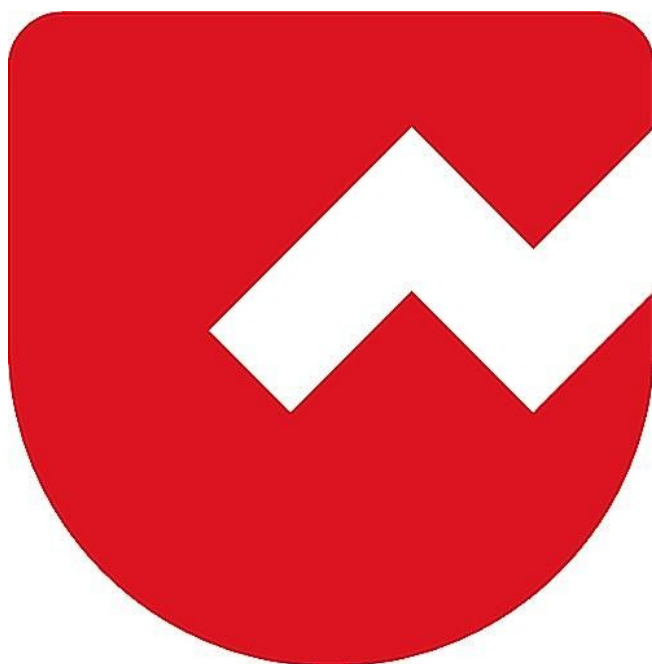




# **Zoho Schools for Graduate Studies**



**Notes**

## Day 8 - Switch Case & Iterative Statements

### 1. Switch Case Rules in Java

The switch statement allows multi-way branching based on the value of an expression.

Key Rules:

1. The switch expression must evaluate to:
  - byte, short, char, int (primitive types)
  - enum types
  - String (Java 7 onwards)
  - Wrapper classes of primitives (Byte, Short, Character, Integer)
2. Case values must be constants or literals – variables are not allowed.
3. Case values must be unique.
4. The default case is optional.
5. break is used to exit the switch, otherwise fall-through occurs.
6. Switch can have 0 or n number of cases.

#### *Example:*

---

```
public class SwitchDemo {  
    public static void main(String[] args) {  
        int x = 2;  
        switch(x) {  
            case 1: System.out.println("One"); break;  
            case 2: System.out.println("Two"); break;  
            case 3: System.out.println("Three"); break;  
            default: System.out.println("Invalid");  
        }  
    }  
}
```

**Output:**  
**Two**

## 2. Fall Through in Switch

If break is not used, control passes from the matched case to the next case(s) until it finds a break or the switch ends. This behavior is called fall-through.

```
int x = 2;
switch(x) {
    case 1: System.out.println("One");
    case 2: System.out.println("Two");
    case 3: System.out.println("Three");
    default: System.out.println("Default");
}
```

**Output:**

**Two**

**Three**

**Default**

## 3. Iterative Statements

Used to execute a set of statements repeatedly until a condition is satisfied.

Types of Loops in Java:

- for loop
- while loop
- do-while loop
- Enhanced for loop (for-each)

### 3.1 For Loop

Structure:

```
for (initialization; condition; increment/decrement) {

}
```

Explanation:

1. Initialization → executed once before loop starts.
2. Condition → checked before every iteration. If false, loop exits.
3. Increment/Decrement → updates the loop variable after every iteration.

Example:

```
for(int i = 1; i <= 5; i++) {  
    System.out.println(i);  
}
```

**Output:**

```
1  
2  
3  
4  
5
```

### 3.2 Enhanced For Loop

Used for iterating over arrays or collections.

Cannot modify the collection while iterating.

Heterogeneous objects can be stored in a collection, but iteration will treat them as Object type if not generic.

Syntax:

```
for (dataType variable : collection) {  
  
}
```

Example:

```
int[] arr = {10, 20, 30};  
for(int x : arr) {  
    System.out.println(x);  
}
```

**Output:**

```
10  
20  
30
```

### 4. Default in Switch

The default block executes when no case matches.

- default is optional.

- It can appear anywhere in the switch, not only at the end.

Example:

```
int x = 5;
switch(x) {
    default: System.out.println("Default");
    case 1: System.out.println("One");
}
```

**Output:**

**Default**

**One**

## 5. Case Count in Switch

A switch can have:

- 0 cases → only default (or even empty).
- n number of cases → depends on requirements.

## Iterative Examples

### 1. Do-While Loop Example

```
public class IterativeDemo {
    public static void main(String[] args) {
        int n = 16;
        do {
            n *= 2; // n = n * 2
        } while (n < 100);
        System.out.println(n);
    }
}
```

**Output: 128**

**Explanation:** The loop multiplies n by 2 until it reaches 128. The condition fails when n = 128 since 128 < 100 is false.

### 2. Nested For Loop Example

```
public class IterativeDemo {
    public static void main(String[] args) {
        int count = 0;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 2; j++) {
                for (int k = 0; k < 2; k++) {
                    count++;
                }
            }
        }
    }
}
```

```
        }  
    }  
}  
System.out.println(count);  
}  
}
```

**Output: 12**

**Explanation:** The innermost loop executes  $3 \times 2 \times 2 = 12$  times, so count becomes 12.