Switch Statement

Switch Statement – Basics

Purpose: Multi-way selection (alternative to nested `if`).
Advantages:

- Clear structure.
- Easy to extend.
- Optimized performance.

Basic syntax:

```
switch (variable_or_expression) {
    case value1:
        // code
        break;
    case value2:
        // code
        break;
    default:
        // default code
}
```

Switch – Key Rules

Important notes:

1. `default` block: executed if no `case` matches.
2. **`break`** keyword**:** exits switch (prevents fall-through).
3. **No fall-through:** omitting `break` leads to unintended execution.

Example

```
switch(day) {
    case 1: System.out.println("Monday"); break;
    case 2: System.out.println("Tuesday"); break;
    default: System.out.println("Unknown");
}
```

Switch Example – Integer

```
public class SwitchExample {
    public static void main(String[] args) {
        int day = 3;
        switch(day) {
            case 1: System.out.println("Monday"); break;
            case 2: System.out.println("Tuesday"); break;
            case 3: System.out.println("Wednesday"); break;
            default: System.out.println("Unknown Day");
        }
    }
}
```

Output: Wednesday

Switch Example – String

```
public class StringSwitchExample {
    public static void main(String[] args) {
        String color = "green";

        switch (color.toLowerCase()) { // Convert to lowercase for case-insensitive
comparison
            case "red":
                System.out.println("Red");
                break;
            case "blue":
                System.out.println("Blue");
                break;
            case "green":
                System.out.println("Green");
                break;
            default:
                System.out.println("Unknown color");
        }
    }
}
```

Result: Green