04-Switch Statement

The switch statement is used to evaluate a variable against a series of values, offering a more readable alternative to lengthy if-else-if chains. Each block in a switch statement is represented by a case, and typically, the execution of a case block is terminated using the break keyword.

Syntax of Switch Conditional Statement:

```
switch (variable) {
    case value1:
        // code to execute if variable equals value1
        break;
    case value2:
        // code to execute if variable equals value2
        break;
    default:
        // code to execute if variable doesn't match any case
}
```

Use Cases of Switch Statement:

- **Selecting a Code Block**: Efficiently choose one among many code blocks to execute based on the value of a variable.
- **Handling Multiple Cases**: Manage multiple scenarios clearly and concisely.

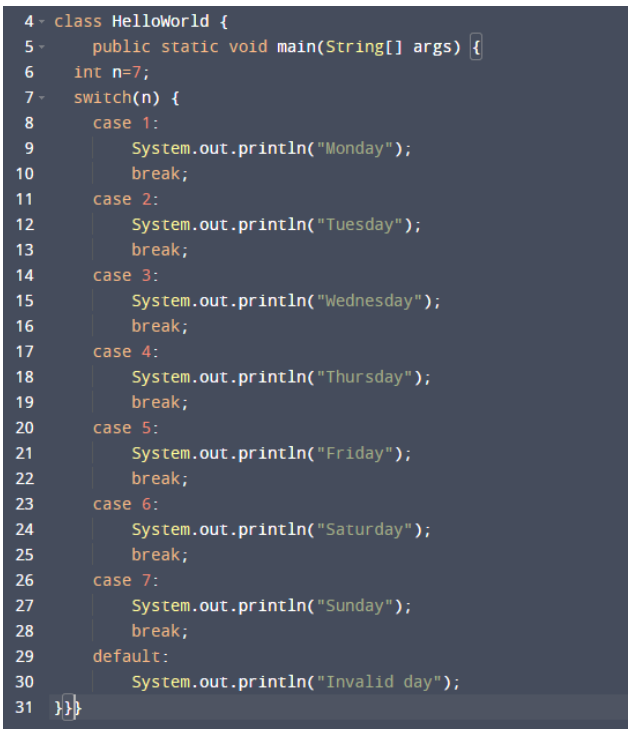
Example: Printing Weekdays

Let's consider an example where we want to print the name of the day based on a number:

- **Using if-else-if approach**: If n = 1 to 7, we need to use multiple if-else blocks to print the corresponding weekday.
- **Using switch statement**: We can replace the complex if-else chain with a switch statement for clarity and efficiency.

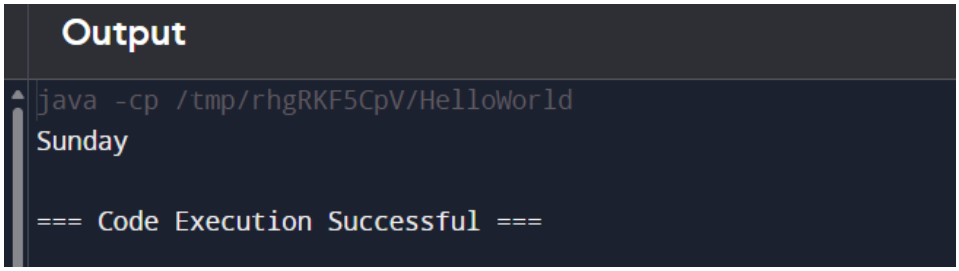
```
switch(n) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
```

```
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    case 4:
        System.out.println("Thursday");
        break;
    case 5:
        System.out.println("Friday");
        break;
    case 6:
        System.out.println("Saturday");
        break;
    case 7:
        System.out.println("Sunday");
        break;
    default:
        System.out.println("Invalid day");
}
```



```
class HelloWorld {
    public static void main(String[] args) {
   int n=7;
   switch(n) {
     case 1:
         System.out.println("Monday");
         break;
     case 2:
         System.out.println("Tuesday");
         break;
     case 3:
         System.out.println("Wednesday");
         break;
     case 4:
         System.out.println("Thursday");
         break;
     case 5:
         System.out.println("Friday");
         break;
     case 6:
         System.out.println("Saturday");
         break;
     case 7:
         System.out.println("Sunday");
         break;
     default:
         System.out.println("Invalid day");
}}}
```

Output:



In this example, if n is 2, the output will be "Tuesday".

```
class HelloWorld {
    public static void main(String[] args) {
  int n=2;
  switch(n) {
    case 1:
        System.out.println("Monday");
        //break;
    case 2:
        System.out.println("Tuesday");
       // break;
    case 3:
        System.out.println("Wednesday");
       // break;
    case 4:
        System.out.println("Thursday");
       // break;
    case 5:
        System.out.println("Friday");
       // break;
    case 6:
        System.out.println("Saturday");
      //  break;
    case 7:
        System.out.println("Sunday");
     //   break;
    default:
        System.out.println("Invalid day");
}}}
```

Output:

```
java -cp /tmp/x9N2MjZZQr/HelloWorld
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Invalid day

=== Code Execution Successful ===
```

The break statement prevents the execution from falling through to subsequent cases. Without the break statement, the program would continue executing the following cases until it encounters a break or reaches the end of the switch block.

The default case is executed if none of the specified cases match the variable's value, ensuring that the program handles unexpected values gracefully.

Note: The break keyword is crucial in Java to prevent fall-through behaviour.