Ministry of Higher Education
Jami University
Scientific assistantship
Faculty (Computer Science)
Department of Information Systems

Switch in java

In Java, the `switch` statement is a control structure that allows you to execute different parts of code based on the value of a variable. It is commonly used when you need to compare a single variable against a series of constant values. The `switch` statement can be used with primitive data types (`int`, `char`, `byte`, `short`) and also with `String` and `enum` types.

General Structure of a `switch` Statement:

```java
switch (variable) {
    case value1:
        // Code to execute if variable == value1
        break;
    case value2:
        // Code to execute if variable == value2
        break;
    ...
    default:
        // Code to execute if none of the cases match
}
```

Important Points:

1. **`switch` statement** tests the value of a variable.

2. **`case` blocks** specify possible values for the variable and the corresponding code to execute.

3. **`break` statement** exits the `switch` block after executing the code for a case.

4. **`default` block** executes if no case matches the value of the variable.

Saifullah Haidari

---

10 Examples of `switch` in Java:

Example 1: Basic `switch` with `int`
```java
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("Invalid day");
}
```
Output:
```
Wednesday
```

Example 2: `switch` with `char`

```java
char grade = 'B';
switch (grade) {
    case 'A':
        System.out.println("Excellent");
        break;
    case 'B':
        System.out.println("Good");
        break;
    case 'C':
        System.out.println("Fair");
        break;
    default:
        System.out.println("Fail");
}
```

Output:

```
Good
```

Example 3: `switch` with `String`

```java
String fruit = "Apple";
switch (fruit) {
    case "Apple":
        System.out.println("It's an Apple");
        break;
    case "Banana":
        System.out.println("It's a Banana");
        break;
    default:
        System.out.println("Unknown fruit");
}
```
Output:
```
It's an Apple
```

Example 4: `switch` with `fall-through` behavior
If `break` is not used, execution continues to the next case until it encounters a `break`.

```java
int month = 2;
switch (month) {
    case 1:
    case 2:
    case 3:
        System.out.println("First Quarter");
        break;
    case 4:
    case 5:
```

```
    case 6:
        System.out.println("Second Quarter");
        break;
    default:
        System.out.println("Other Quarter");
}
```

Output:

```
First Quarter
```

Example 5: `switch` with `enum`

```java
enum Day { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY }

Day day = Day.FRIDAY;
switch (day) {
    case MONDAY:
    case TUESDAY:
        System.out.println("It's a weekday");
        break;
    case FRIDAY:
        System.out.println("It's almost the weekend!");
        break;
    case SATURDAY:
    case SUNDAY:
```

        System.out.println("It's the weekend!");
        break;
    default:
        System.out.println("Invalid day");
}
```
Output:
```
It's almost the weekend!
~~~

Example 6: `switch` without `break` (fall-through)
```java
int number = 2;
switch (number) {
    case 1:
        System.out.println("One");
    case 2:
        System.out.println("Two");
    case 3:
        System.out.println("Three");
    default:
        System.out.println("Invalid number");
}
~~~
Output:
```
```

```
Two
Three
Invalid number
```

Explanation: Since there is no `break`, execution "falls through" all remaining cases.

Example 7: `switch` with `default` case at the start

```java
int age = 25;
switch (age) {
    default:
        System.out.println("Age is neither 18 nor 21");
        break;
    case 18:
        System.out.println("You are 18");
        break;
    case 21:
        System.out.println("You are 21");
        break;
}
```

Output:

```
Age is neither 18 nor 21
```

Saifullah Haidari

Example 8: `switch` with multiple cases sharing the same action

```java
int number = 5;
switch (number) {
    case 1:
    case 2:
    case 3:
        System.out.println("Low number");
        break;
    case 4:
    case 5:
    case 6:
        System.out.println("Medium number");
        break;
    default:
        System.out.println("High number");
}
```

Output:

```
Medium number
```

Example 9: `switch` inside a method

```java
public class Test {
```

    public static void printDay(int day) {
        switch (day) {
            case 1: System.out.println("Monday"); break;
            case 2: System.out.println("Tuesday"); break;
            default: System.out.println("Other day");
        }
    }

    public static void main(String[] args) {
        printDay(2);
    }
}
```
Output:
```
Tuesday
~~~

Example 10: `switch` with variable initialization
```java
int hour = 10;
String timeOfDay;

switch (hour) {
    case 6: case 7: case 8: case 9: case 10:
        timeOfDay = "Morning";
        break;
~~~

```
    case 11: case 12: case 13: case 14:
        timeOfDay = "Noon";
        break;
    default:
        timeOfDay = "Evening";
}

System.out.println("It's " + timeOfDay);
```

Output:

```
It's Morning
```

Summary of Key Points:

- The `switch` statement is useful for handling multiple cases with the same variable.
- It works with `int`, `char`, `String`, `enum`, and other supported types.
- The `break` statement prevents fall-through behavior, and the `default` case provides a fallback if no cases match.

Saifullah Haidari