10.9-Enum If and Switch

Java provides a special data type called **Enum**, typically used to define a collection (or set) of constants. To be more precise, an Enum is a special form of a Java class. An Enum can contain constants, attributes, methods, and more. It is commonly used when you have a fixed set of related constants, such as days of the week, directions, or statuses.

Key Features of Enum in Java:

- By default, **Enum constants are public, static, and final**.
- Enum constants are accessible using dot (.) syntax.
- Enum classes can also contain **attributes and methods**.
- Enum classes **cannot inherit other classes**, and you **cannot create objects** of Enum types.
- Enum classes can **implement interfaces**.
- Since Enums are named constants, you can compare them easily using == or control statements like **if-else** and **switch**.

Example of Enum in Java:

```
enum Status {
    Running, Failed, Pending, Success;
}
```

Here, Status is an Enum with four constants: Running, Failed, Pending, and Success.

Using Enum with if-else Statements

You can compare Enum constants using if-else statements just like primitive data types. For example:

```
public class Demo {
    public static void main(String[] args) {
        Status s = Status.Running;

        // Using if-else for comparing Enum constants
        if (s == Status.Running) {
            System.out.println("All good");
        } else if (s == Status.Failed) {
            System.out.println("Try Again");
        } else if (s == Status.Pending) {
            System.out.println("Please wait");
        } else {
            System.out.println("Done");
        }
    }
}
```

Output:

```
All good
```

In this example, the if-else statement compares the Enum constant s with the different possible statuses and prints the corresponding message.

Using Enum with Switch Statements

The switch statement is a more concise way to handle multiple conditions compared to if-else. It's especially well-suited for comparing Enum constants because the case values can directly use the Enum constants, without needing to write the Enum type.

Syntax of Switch Statement:

When you have multiple options and need to perform different actions based on those options, a switch statement is helpful. Here's how it works:

- The switch statement compares the value of a variable with several possible cases.
- Each case contains a distinct block of code.
- A break statement is typically used to exit the switch block once a match is found, although it's not mandatory.

Using Enum with Switch:

```
enum Status {
    Running, Failed, Pending, Success;
}
public class MyClass {
  public static void main(String args[]) {
    Status s = Status.Running;

        // Using switch for comparing Enum constants
        switch (s) {
            case Running:
                System.out.println("All good");
                break;

            case Failed:
                System.out.println("Try again");
                break;

            case Pending:
                System.out.println("Please wait");
                break;

            default:
                System.out.println("Done");
                break;
        }
  }
}
```

Output:

```
All good
```

In this example, the switch statement compares the Enum constant s and executes the corresponding block of code. Unlike the if-else structure, the switch statement provides a cleaner and more readable approach.

Advantages of Using Switch with Enums:

- **Readability**: The switch statement improves readability, especially when dealing with multiple conditions.
- **Clarity**: It avoids the clutter of multiple else-if conditions, making the code clearer and easier to maintain.
- **Efficiency**: In some cases, the switch statement can be faster as it's designed for comparing discrete values like Enums.