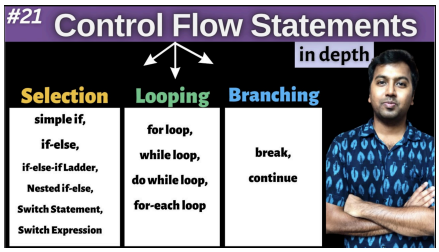
Java 14: Switch Enhancements

Thursday, 4 September 2025 2:41 PM

We have already covered Switch statement



Classic way of writing a Switch Statement:

```
enum Days{
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;
}
```

```
//Switch statement
Days day = Days.FRIDAY;

switch (day) {
    case MONDAY:
        System.out.println(6);  //no of characters
        break;
    case TUESDAY:
        System.out.println(7);
        break;
    case WEDNESDAY:
        System.out.println(9);
        break;
    case THURSDAY:
        System.out.println(8);
        break;
    case FRIDAY:
        System.out.println(6);
        break;
    case SATURDAY:
        System.out.println(8);
        break;
    case SUNDAY:
        System.out.println(6);
        break;
}
```

OR →

```
//Switch statement
Days day = Days.FRIDAY;

switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        System.out.println(6);
        break;
    case TUESDAY:
        System.out.println(7);
        break;
    case THURSDAY:
    case SATURDAY:
        System.out.println(8);
        break;
    case WEDNESDAY:
        System.out.println(9);
        break;
}
```

There are many problems with Old Switch Statement, Lets see one by one and also how Java 14 Switch features resolved those:

1st: Multiple lines of Case Staking:

old Switch Statement:

Each Case need its own line.

```
//Switch statement
Days day = Days.FRIDAY;

switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        System.out.println(6);
        break;
    case TUESDAY:
        System.out.println(7);
        break;
    case THURSDAY:
    case SATURDAY:
        System.out.println(8);
        break;
    case WEDNESDAY:
        System.out.println(9);
        break;
}
```

Java14, Switch:

Commas separated case grouping

```
Days day = Days.FRIDAY;

switch (day) {
    case MONDAY, FRIDAY, SUNDAY:
        System.out.println(6);
        break;
    case TUESDAY:
        System.out.println(7);
        break;
    case THURSDAY, SATURDAY:
        System.out.println(8);
        break;
    case WEDNESDAY:
        System.out.println(9);
        break;
}
```

2nd: Fall through by default:

old Switch Statement:

```
//Switch statement
Days day = Days.FRIDAY;

switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        System.out.println(6);
    case TUESDAY:
        System.out.println(7);
        break;
    case THURSDAY:
    case SATURDAY:
        System.out.println(8);
        break;
    case WEDNESDAY:
        System.out.println(9);
        break;
}
```

Missed "break"

Output:

```
/Users/shrayanshjain/Library/Java/
6
7

Process finished with exit code 0
```

Because of "break" missing, it fallback to next case and executed that also.

Java14, Switch:

New form of Switch Label i.e. "case L ->" , tells that only the code to the right of the label need to be executed, if the label is matched.

No need of add "break" for each case now.

```
Days day = Days.FRIDAY;

switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> System.out.println(6);
    case TUESDAY -> System.out.println(7);
    case THURSDAY, SATURDAY -> System.out.println(8);
    case WEDNESDAY -> System.out.println(9);
}
```

3rd: No Value Return:

old Switch Statement:

```
//Switch statement
Days day = Days.FRIDAY;
int count = 0;
switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        count = 6;
        break;
    case TUESDAY:
        count = 7;
        break;
    case THURSDAY:
    case SATURDAY:
        count = 8;
        break;
    case WEDNESDAY:
        count = 9;
        break;
}

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

Java14, Switch:

- **Can be used as expression**

If there is only 1 statement :

```
Days day = Days.FRIDAY;

int count = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> 6;
    case TUESDAY -> 7;
    case THURSDAY, SATURDAY -> 8;
    case WEDNESDAY -> 9;
}; //semicolon required here (end of assignment statement)

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

If there is a block, we can use "yield"

```
int count = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> {

        if (day == Days.SUNDAY) {
            throw new IllegalArgumentException("Sunday is holiday");
        }
        yield 6;
    }
    case TUESDAY -> 7;
    case THURSDAY, SATURDAY -> 8;
    case WEDNESDAY -> 9;
}; //semicolon required here (end of assignment statement)

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

4rd: No exhaustiveness check:

```
enum Days{
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;
}
```

old Switch Statement:

Compiler doesn't force to cover all values

```
//Switch statement
Days day = Days.FRIDAY;
int count = 0;
switch (day) {
    case MONDAY:
        count = 6;
        break;
    case TUESDAY:
        count = 7;
        break;
}
System.out.println(count);
```

If you run above code, you will see output: 0, as Days.FRIDAY is not matched with any case.

Java14, Switch:

When we use Switch as an expression, compiler will force us to cover all the cases

```
Days day = Days.FRIDAY;

int count = switch (day) {
    case MONDAY -> 6;
    case THURSDAY -> 8;
    case WEDNESDAY -> 9;
};

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

'switch' expression does not cover all possible input values

So, we should cover all possible cases, so use default, if we are not sure what all cases can come

```
Days day = Days.FRIDAY;

int count = switch (day) {
    case MONDAY -> 6;
    case THURSDAY -> 8;
    case WEDNESDAY -> 9;
    default -> 0;
};
System.out.println(count);
```

OR

```
Days day = Days.FRIDAY;

int count = switch (day) {
    case MONDAY -> 6;
    case THURSDAY -> 8;
    case WEDNESDAY -> 9;
    default -> throw  new IllegalArgumentException("not allowed");
};
System.out.println(count);
```

5th: All Case blocks share the same scope:

old Switch Statement:

```
// Switch statement
 Days day = Days.FRIDAY;

 switch (day) {
     case MONDAY:
         String stringVal = "Monday";
         System.out.println(stringVal);
         break;

     case TUESDAY: {
         String stringVal = "Tuesday";
         System.out.println
         break;
     }
 }
```

Variable 'stringVal' is already defined in the scope

Java14, Switch:

Each 'case' with '->' is treated as independent scope

```
String msg = switch (day) {
    case MONDAY -> {
        String stringVal = "Monday";
        yield stringVal;
    }
    case TUESDAY -> {
        String stringVal = "Tuesday";
        yield stringVal;
    }
    default -> "others";
};
```

To resolve this, we have to specifically make sure that, we used the blocks

```java
// Switch statement
Days day = Days.FRIDAY;

switch (day) {
    case MONDAY: {
        String stringVal = "Monday";
        System.out.println(stringVal);
        break;
        }

    case TUESDAY: {
        String stringVal = "Tuesday";
        System.out.println(stringVal);
        break;
    }
}
```

Old Style "case L:" with new feature:

```java
int day = 1;

String typeOfDay = switch (day) {
    case 1, 7:
        yield "Weekend";    // new feature: yield
    case 2, 3, 4, 5, 6:
        yield "Weekday";
    default:
        yield "Invalid";
};
```

- Before Java 14, **colon cases could not return values**. Now they can, via yield.
- We still need break if it's just a **statement switch** (not returning) else it will fall through.