Lecture 19

Arrow labels & Switch Expression

- These 2 features in switch were added in Java 14.
- There were some limitations in basic syntax of switch like in every case you need to add break otherwise there will be fall through behaviour.

→ New features added in Switch in Java 14:-

① Grouping multiple case labels using commas is introduced in Java 14 as part of enhanced switch expression.

e.g:-

```
char ch = 'a';
switch (ch) {
    case 'a', 'e', 'i', 'o', 'u':
        System.out.println ("Vowels");
        break;
    case 'b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q',
         'r', 's', 't', 'v', 'w', 'x', 'y', 'z' :
        System.out.println ("Consonant");
        break;
    default:
        System.out.println ("Not a valid alphabet letter.");
}
```

grouping multiple case labels, not allowed before Java 14.

Still here we are not using arrow labels in this program

With enhanced switch expression if we use arrow labels then no need to write break. So with arrow labels we basically will not have fall through.

e.g:- In the previous example we can write:-

```
case 'a', 'e', 'i', 'o', 'u' ->
        System.out.println ("vowel");
```

General Syntax:-

case value -> anexpression or block of statements

NOTE:- After arrow in that case we can write only a single expression if you want to write more than one expression or statements then we need to put those in block { }.

→ So now code becomes more readable, more compact & more safe (no fall through)

Practice Time:-

```
case 1 -> int a=5; //invalid
case 1 : int a=5; //valid
case 1 -> if (condition) {
            // if block
          }                    => invalid
case 1 : if (condition) {
            // if block
          }                    valid
case 1 -> fun1(); // valid method calling
case 1 -> 7; //error
case1 -> "Monday"; //error
```

Switch Expression :-

- In Java 14 switch expression were introduced to simply switch structure.
- Unlike the traditional switch statement, a switch expression can return a value. making the code more concise and readable

(Basic) Traditional Switch statement	Switch expression
① will not return any value	Can return a value
② break is must	not required any break statement
③ Multiple cases can not be grouped with commas	Multiple cases can be grouped with commas.
④ Default is optional.	default must cover any unmatched cases when used with switch expression.

e.g:-

```
int day = 1;
String result = switch (day) {
    case 1, 2, 3, 4, 5 -> "Weekday";
    case 6, 7 -> "Weekend";
    default -> "Not a valid choice";
};
System.out.println ("It's " + result);
```

Case I

default is must when switch is used as an expression.

← Comma is must

→ We can use this switch expression in return statement as well.

like:- return expression;

↓

this expression can be switch expression.

Eg -

```
public class SwitchDemo {

    public static String dayofWeek(String day) {
        return switch (day) {
            case "Mon", "Tue", "Wed", "Thur", "Fri" -> "weekday";
            case "Sat", "Sun" -> "Weekend";
            default -> "Invalid Day choice";
        };

    public static void main(String[] args) {
        String day = "Fri";
        System.out.println(dayofWeek(day));
    }
```

Case II

⇒ it is implicitly translated to
→ { yield "Weekend"; }

NOTE:- In both caseI & CaseII switch is acting as an expression meaning its value can be evaluated & in case I we assign that evaluated value of switch to a variable "result" and in case II its values is being returned from a method dayofWeek( ).

Practice Time:-

in above example if I write -

```
case "Sat", "Sun" -> {
    System.out.println("You entered " + day);
    return "Weekend";  -> //error
}
```

⇒ Here we use another keyword yield instead of return.

yield can be used with switch expression only.

But if we write :-

case "Sat", "Sun" -> yield "Weekend"; ⇒ will give error

yield should be in code block { }

⇒ case "Sat", "Sun" -> { yield "Weekend"; } ⇒ not error.

NOTE:- Default is must to cover all the unmatched cases when we use switch expression.

But let's take following example:-

```
enum Day { Mon, Tue, Wed, Thu, Fri, Sat, Sun, Fun }
Day day = Day.Fun;
String dayofWeek = switch (day) {
    case Mon, Tue, Wed, Thu, Fri -> "Weekday";
    case Sat, Sun -> "Weekend";
    case Fun -> "You are in Parallel Universe!";
};
```

In this case No default but it will not give any error. because we have covered all the possible cases.