Prácticas: Sesión 3

Programación Orientada a Objetos

Grado en Ingeniería Informática

Curso: 2024-2025

Profesor: Juan José López Jiménez

DIS
Departamento de
Informática y Sistemas

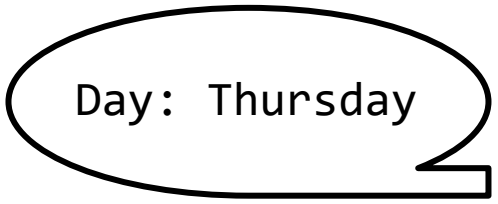
Contenido

- Switch tradicional y switch expression en Java
- Record en Java

Switch

- Tradicional:

```
int day = 4;
String dayName;
switch(day) {
    case 1:
        dayName = "Monday";
        break;
    case 2:
        dayName = "Tuesday";
        break;
    case 3:
        dayName = "Wednesday";
        break;
    case 4:
        dayName = "Thursday";
        break;
    case 5:
        dayName = "Friday";
        break;
    default:
        dayName = "Unknown";
        break;
}
System.out.println("Day: " + dayName);
```



Switch

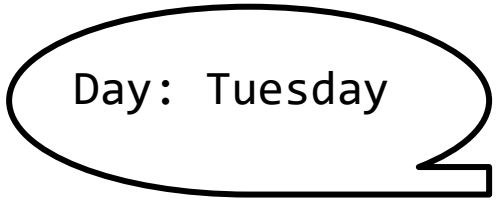
- Tradicional (múltiples casos):

```
int day = 6;
String dayName;
switch(day) {
    case 1:
        dayName = "Monday";
        break;
    case 2:
        dayName = "Tuesday";
        break;
    case 3:
        dayName = "Wednesday";
        break;
    case 4:
        dayName = "Thursday";
        break;
    case 5:
        dayName = "Friday";
        break;
    case 6:
    case 7:
        dayName = "Weekend";
        break;
    default:
        dayName = "Unknown";
        break;
}
System.out.println("Day: " + dayName);
```

Switch

- Tradicional:

¿Qué ocurre si se nos olvida un break?




```
int day = 1;
String dayName;
switch(day) {
    case 1:
        dayName = "Monday";
    case 2:
        dayName = "Tuesday";
        break;
    case 3:
        dayName = "Wednesday";
        break;
    case 4:
        dayName = "Thursday";
        break;
    case 5:
        dayName = "Friday";
        break;
    default:
        dayName = "Unknown";
        break;
}
System.out.println("Day: " + dayName);
```

Switch

- ¿Posible solución?
  - Java introdujo en Java 14 en adelante "**Switch expression**".

```
int day = 4;
String dayName = switch (day) {
    case 1 -> "Monday";
    case 2 -> "Tuesday";
    case 3 -> "Wednesday";
    case 4 -> {
        System.out.println("This is Thursday");
        yield "Thursday";
    }
    case 5 -> "Friday";
    default -> "Unknown";
};

System.out.println("Day: " + dayName);
```

```
This is Thursday
Day: Thursday
```

Switch

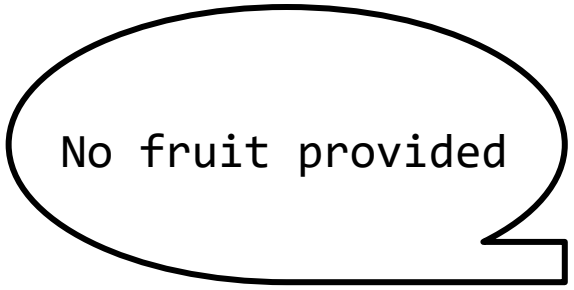
- **Switch expression** con **yield** o no:

```
int day = 4;
String dayName = switch (day) {
    case 1 -> "Monday";
    case 2 -> "Tuesday";
    case 3 -> "Wednesday";
    case 4 -> {
        System.out.println("This is Thursday");
        yield "Thursday";
    }
    case 5 -> "Friday";
    default -> "Unknown";
};

System.out.println("Day: " + dayName);
```

Switch

- **Switch expression** con nulos:




```
String fruit = null;

String result = switch (fruit) {
    case "Apple" -> "It's an apple";
    case null -> "No fruit provided";
    default -> "Unknown fruit";
};

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

Switch

- **Cuidado:** Switch expression con “:”:

```
int day = 4;
String dayName = switch (day) {
    case 1 : "Monday";
    case 2 -> "Tuesday";
    case 3 -> "Wednesday";
    case 4 -> {
        System.out.println("This is Thursday");
        yield "Thursday";
    }
    case 5 -> "Friday";
    default -> "Unknown";
};

System.out.println("Day: " + dayName);
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
	Syntax error, insert "AssignmentOperator Expression" to complete Expression
	The left-hand side of an assignment must be a variable
	Mixing of different kinds of case statements '->' and  ':' is not allowed within a switch

	at pruebas.Switch.main(Switch.java:107)
```

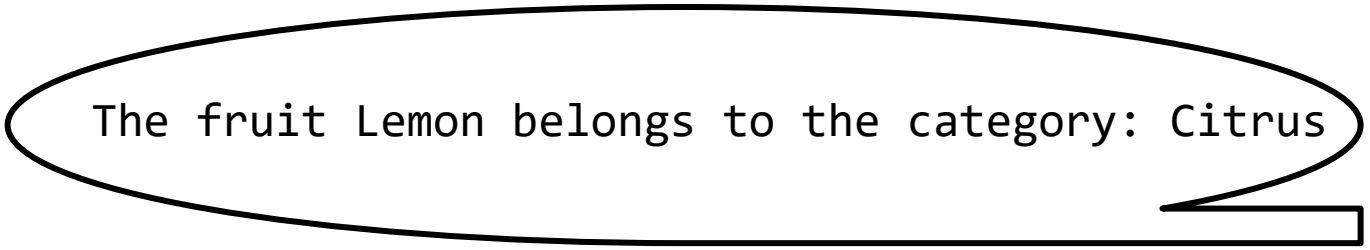
Switch

- Switch expresión permite concatenar varios casos:

```
String fruit = "Lemon";

String category = switch (fruit) {
    case "Orange", "Lemon", "Lime" -> "Citrus";
    case "Apple", "Banana", "Grapes" -> "Non-Citrus";
    default -> "Unknown Category";
};

System.out.println("The fruit " + fruit + " belongs to the category: " +
category);
```



Switch

Aspecto	switch tradicional	switch expression
Uso de break	Requiere break.	No se requiere break.
Devolución de valor	No devuelve valor directamente.	Devuelve un valor que se puede asignar.
Sintaxis	Propensa errores con break.	Más concisa y legible.
Bloques de código	Se puede usar, pero es más complejo.	Usa {} y yield para devolver valores.
Manejo de null	No puede manejar null.	Se permite manejar null explícitamente.
Concatenación de casos	Puede concatenar casos sin break.	Puede concatenar varios casos, usando ,.

Record

```
class Point {
    private final int x;
    private final int y;

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    int x() { return x; }
    int y() { return y; }

    public boolean equals(Object o) {
        if (!(o instanceof Point)) return false;
        Point other = (Point) o;
        return other.x == x && other.y == y;
    }

    public int hashCode() {
        return Objects.hash(x, y);
    }

    public String toString() {
        return String.format("Point[x=%d, y=%d]", x, y);
    }
}
```

Clase inmutable

Record

```
class Point {
    private final int x;
    private final int y;

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    int x() { return x; }
    int y() { return y; }

    public boolean equals(Object o) {
        if (!(o instanceof Point)) return false;
        Point other = (Point) o;
        return other.x == x && other.y == y;
    }

    public int hashCode() {
        return Objects.hash(x, y);
    }

    public String toString() {
        return String.format("Point[x=%d, y=%d]", x, y);
    }
}
```

```
record Point(int x, int y) { }
```