# Laborator 6

## Java 21 Advanced Features: Sealed Classes, Enums, Records

**Exercise 1: Define and Use a Sealed Class Hierarchy**

Create a sealed class `**Shape**` with permitted subclasses `**Circle**`, `**Rectangle**`, and `**Triangle**`. Implement a method to describe each shape.

**Exercise 2: Use Enums with Custom Fields and Methods**

Create an enum `**Day**` with a field to indicate if it’s a weekend and a method to describe the day.

**Exercise 3: Combine Records and Sealed Classes\*\***

Create a sealed class `Result` with two record subclasses: `Success(String message)` and `Error(String reason)`.

**What is a Marker Interface?**

A **marker interface** is an interface with **no methods or fields** - its purpose is to **mark** a class as having a certain property, typically to signal special behavior to some other component (like a framework, library, or utility method).

Classic examples in Java:

- `Serializable`

- `Cloneable`

- `Remote`

**Exercise 4: Create a Custom Marker Interface**

Create a marker interface `ImportantTask`. Create two task classes: `RegularTask` and `CriticalTask`. Mark `CriticalTask` with the marker interface.

**Exercise 5: Use Marker Interface in Business Logic**

Write a method that accepts a generic `Object` and checks if it’s an `ImportantTask` before printing a warning.

**Exercise 6: Combine Marker Interface with Sealed Classes**

Use a marker interface to distinguish important subclasses within a sealed class hierarchy.

**Exercise 7: Use Marker Interface to Restrict Operations**

Simulate a scenario where only objects marked as `Trackable` can be logged.