🧪 Spring AOP Hands-On Questions – Customer Model

☑ Pre-requisites

- Java 8+
- Spring Boot with AOP
- Basic understanding of Beans and Annotations

◇ Hands-On 1: Logging Before Adding Customer

Objective:
Create a basic aspect that logs a message **before** a customer is added.

Task:
- Create a `CustomerService` class with a method to add a customer.
- Define an aspect to log a message **before** this method executes.

◇ Hands-On 2: After and AfterReturning Advice for Update Operation

Objective:
Use AOP to log after a method completes and also log its return value.

Task:
- Add a method in `CustomerService` that updates customer details and returns a success message.
- Create an aspect that logs a message **after** the method runs.
- Add another advice that captures and logs the **returned value**.

◇ Hands-On 3: Handling Exceptions Using `@AfterThrowing`

Objective:
Log exceptions thrown from a method using `@AfterThrowing`.

Task:
- Create a method in `CustomerService` that deletes a customer based on ID.
- If the ID is invalid (e.g., 0 or negative), throw an exception.
- Create an aspect that logs the exception message using AOP.

◇ Hands-On 4: Use `@Around` to Log Execution Time

Objective:
Use `@Around` advice to measure the execution time of a method.

Task:
- Add a method in `CustomerService` to simulate a purchase process (e.g., use `Thread.sleep()`).
- Use an `@Around` aspect to measure and log how long the method takes to execute.

◇ Hands-On 5: Create a Reusable Pointcut

Objective:
Create a reusable `@Pointcut` expression for logging across multiple methods.

Task:
- Define a pointcut that matches all methods in `CustomerService`.
- Use that pointcut to log a generic message before **any method** in the class is executed.