**1️. Violation of the “Single Responsibility Principle” (SRP)**

* The EmployeeManager class is handling too many responsibilities, including:
* Adding employees (addEmployee)
* Promoting employees (promoteEmployee)
* Removing employees (removeEmployee)
* Each function should be separated into different classes to improve maintainability.

**2. Violation of the “Open-Closed Principle” (OCP)**

* Every time a new role (Manager, Developer, Tester) is added, the entire class must be modified.
* If new roles (e.g., HR, Designer) are introduced, all methods must be updated, making the code hard to maintain.
* **Solution**: Use inheritance or enum to avoid modifying existing code.

**3️. Excessive Code Duplication**

* In addEmployee(), promoteEmployee(), removeEmployee(), identical logic is repeated:

if (role.equals("Manager")) {

    System.out.println("Adding Manager: " + firstName + " " + lastName);

    System.out.println("Sending welcome email to " + email);

}

* **Solution:** Use encapsulation to centralize this logic in an Employee class.

**4. Hardcoded Strings for Role Checking**

* The use of: if (role.equals("Manager"))
* **Problem:** This is prone to errors (typos, inconsistencies).
* **Solution:** Use enum instead of String to define roles.
* **Improved Solution:**

1. Create an Employee class to store employee information.

2. Use an enum Role instead of String for role management.

3. Create Manager, Developer, and Tester classes that extend Employee.

4. Refactor EmployeeManager to handle employees in a cleaner way.