First semester 2022/2023
Advanced topics in software development
Homework Assignment # 1
Design patterns
ETA: 28 Nov midnight.

You are provided with an initial codebase inside the following repository path:

[Cap2025/src/edu/najah/cap/advance/assignments/assignment1]

This code represents a **naive and intentionally flawed** implementation of the **TaskMaster Processing System (TMPS)** — a job processing and resource-management application.
Your mission is to enhance and refactor this system according to the specifications below.

---

1. Understand the Provided Code

Begin by reading and analyzing all classes in the provided folder.

The current implementation contains multiple:

- design problems
- code smells
- violations of object-oriented principles
- incomplete TODO items
- structural issues that are *not* explicitly marked

It is your responsibility to identify these problems and document them.

2. Identify the Current Issues

As part of your submission, you must provide a **"Current Issues Report"** that clearly explains:

- **What is wrong with the provided code?**
  (e.g., SOLID violations, tight coupling, duplication, misuse of inheritance, poor cohesion, etc.)

Your report must be **specific** — reference actual:

- classes
- methods
- code segments
- symptoms of the identified issues

3. Improve & Refactor the System

Your main task is to transform the naive implementation into a clean, modular, extensible system by applying the following **four mandatory design patterns**.
Your refactored system must satisfy the official TaskMaster Inc. requirements.

✔ Requirement 1 — Connection Pool

Implement a proper `ConnectionPool` with the following behavior:

- Supports **up to 10** reusable connections
- `acquire()` returns an available connection (and blocks or queues if none are available)
- `release(connection)` returns the connection to the pool
- **All** job executions must obtain their connection through the pool
  (via the Proxy pattern)

✔ Requirement 2 — Prototype Pattern for Job Templates

The current implementation recreates large job templates from scratch.
You must replace this with the Prototype pattern.

You must implement:

- A `JobPrototype` interface
- Concrete prototypes:
    - `EmailJobTemplate`
    - `DataProcessingJobTemplate`
    - `ReportJobTemplate`
- A `JobTemplateRegistry` for storing reusable prototypes
- Efficient cloning of templates instead of rebuilding them

(You may also propose improvements or enhancements where appropriate.)

✔ Requirement 3 — Strategy Pattern for Job Execution

The existing `JobExecutor` relies on long `if/else` blocks with hard-coded logic.
Refactor it to use the Strategy Pattern.

You must:

- Create a `JobStrategy` interface
- Implement the following strategies:
    - `EmailJobStrategy`
    - `DataProcessingStrategy`
    - `ReportGenerationStrategy`
- Create a `JobStrategyFactory` that maps job type → strategy
- Remove **all** type-checking logic from `JobExecutor`

✔ Requirement 4 — Proxy Pattern for Controlled Job Execution

Job execution is currently unprotected and unmonitored.

You must introduce a new **controlled execution flow** using a Proxy.
The Proxy must:

- Validate user permissions
- Log job start/end
- Measure execution time
- Acquire a DB connection from the ConnectionPool
- Delegate to the real executor
- Release the connection afterwards

The **existing** execution method should remain accessible
(because it may be used internally where permission checks are unnecessary).

4. Deliverables

✔ 1. Refactored Code (GitHub)

Push your improved version to your GitHub repository and add both reviewers:

- **mustafakassaf@gmail.com**
- **Muhammadghqzih@gmail.com**

✔ 2. Updated Class Diagram

You must include a UML class diagram showing:

- All major components
- Application of the Prototype, Strategy, Proxy, and Connection Pool patterns
- Key relationships between classes

✔ 3. Design Decision Document

Describe:

- Why each design pattern was chosen
- What alternatives you considered
- How the patterns interact
- How your architecture improves scalability, flexibility, and maintainability

✔ 4. Issue Analysis Report

Provide a concise document listing:

- All issues you identified in the original code
- Why each issue is problematic
- How your refactoring resolves the issue

5. Deadlines

- **Final Submission: 28 / 11**
- **Design Review (with Mohammad Qzeh): 1–4 / 12**