## 🔰 ****1. Mục tiêu của đoạn code****

Đoạn mã PersonalTaskManagerViolations cung cấp chức năng:

* Thêm một nhiệm vụ cá nhân vào cơ sở dữ liệu dạng JSON (tasks\_database.json).
* Kiểm tra dữ liệu đầu vào (tiêu đề, ngày đến hạn, mức ưu tiên).
* Phát hiện nhiệm vụ trùng lặp (cùng tiêu đề và ngày đến hạn).
* Ghi nhận thông tin như mức độ ưu tiên, trạng thái, thời điểm tạo, và xử lý nhiệm vụ lặp lại.

## ⚠️ ****2. Các vấn đề và nguyên tắc thiết kế bị vi phạm****

|  |  |  |
| --- | --- | --- |
| Vấn đề | Nguyên tắc bị vi phạm | Giải thích |
| Hàm addNewTaskWithViolations() quá dài | **SRP (Single Responsibility Principle)** | Làm quá nhiều việc: kiểm tra, xử lý logic, lưu file |
| Lặp lại đoạn đọc/ghi file | **DRY (Don't Repeat Yourself)** | Cùng logic xử lý nhưng lặp lại ở nhiều nơi |
| Dùng UUID, thêm trường is\_recurring mà chưa dùng | **YAGNI (You Aren’t Gonna Need It)** | Thêm tính năng chưa cần thiết |
| Mảng mức độ ưu tiên khai báo cứng | **Avoid Hardcoded Values** | Khó mở rộng, không tái sử dụng |
| Logic kiểm tra trùng lặp viết tay | **KISS (Keep It Simple, Stupid)** | Cần trừu tượng hóa cho dễ đọc và bảo trì |

## 🧱 ****3. Nguyên tắc thiết kế áp dụng khi refactor****

* **SRP**: Mỗi hàm chỉ có một trách nhiệm rõ ràng.
* **DRY**: Tách các đoạn mã lặp lại thành phương thức tái sử dụng.
* **KISS**: Giữ code ngắn gọn, dễ đọc, dễ hiểu.
* **Encapsulation**: Gói dữ liệu và hành vi vào các đối tượng/phương thức liên quan.
* **Modularization**: Phân chia logic thành các khối chuyên biệt.

## 🔧 ****4. Các bước refactor cụ thể****

### ✅ ****Bước 1: Tạo class**** Task ****để đại diện nhiệm vụ****

java

Sao chépChỉnh sửa

public class Task {

private String id;

private String title;

private String description;

private LocalDate dueDate;

private String priority;

private String status;

private LocalDateTime createdAt;

private LocalDateTime lastUpdatedAt;

// Các constructor, getter/setter, toJSON() sẽ hỗ trợ việc serialize ra JSON

}

**Lý do:** Giúp mã rõ ràng, dễ mở rộng và tránh thao tác trực tiếp với JSONObject.

### ✅ ****Bước 2: Tách logic kiểm tra đầu vào thành phương thức riêng****

java

Sao chépChỉnh sửa

private boolean isValidInput(String title, String dueDateStr, String priorityLevel) { ... }

**Lý do:** Tránh nhồi nhét logic kiểm tra vào hàm chính.

### ✅ ****Bước 3: Tách logic kiểm tra trùng nhiệm vụ****

java

Sao chépChỉnh sửa

private boolean isDuplicateTask(JSONArray tasks, String title, LocalDate dueDate) { ... }

**Lý do:** Đơn giản hóa hàm chính và tách trách nhiệm rõ ràng.

### ✅ ****Bước 4: Tách phương thức đọc/ghi JSON****

java

Sao chépChỉnh sửa

private JSONArray loadTasksFromFile() { ... }

private void saveTasksToFile(JSONArray tasks) { ... }

**Lý do:** Tránh lặp lại mã đọc/ghi file.

### ✅ ****Bước 5: Áp dụng Enum cho mức độ ưu tiên****

java

Sao chépChỉnh sửa

public enum Priority {

THAP, TRUNG\_BINH, CAO;

public static boolean isValid(String input) { ... }

}

**Lý do:** Dễ kiểm soát và mở rộng thay vì dùng mảng chuỗi cứng.

## 📌 ****5. Giải thích lý do cho từng thay đổi****

|  |  |
| --- | --- |
| Thay đổi | Lý do chính |
| Tạo class Task | Gói dữ liệu, dễ serialize/deserialize |
| Tách hàm kiểm tra input | SRP + dễ kiểm thử |
| Enum Priority | Tránh lỗi chính tả, dễ mở rộng |
| Hàm kiểm tra trùng lặp | KISS + SRP |
| Tách logic JSON | DRY, dễ sửa đổi backend sau này |

## 🔍 ****6. So sánh trước và sau refactor****

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Trước refactor | Sau refactor |
| **Độ dài hàm chính** | Dài (~80 dòng) | Ngắn gọn (~10-15 dòng) |
| **Mức độ phân tách trách nhiệm** | Không rõ ràng | Rõ ràng theo từng hàm |
| **Tính mở rộng** | Khó thêm tính năng (ví dụ: cập nhật nhiệm vụ) | Dễ mở rộng qua class Task |
| **Tái sử dụng mã** | Không có | Có (validate, save/load, duplicate check) |
| **Tính dễ đọc và bảo trì** | Thấp | Cao |