**1. Mô tả mục tiêu của đoạn code**

Hàm addNewTaskWithViolations có nhiệm vụ thêm một nhiệm vụ mới vào cơ sở dữ liệu với các thông tin: tiêu đề, mô tả, ngày đến hạn, mức độ ưu tiên, trạng thái, ngày tạo, và thông tin lặp lại nếu có.

2. Các vấn đề vi phạm nguyên tắc trong code ban đầu

| **Vấn đề** | **Mô tả chi tiết** | **Nguyên tắc bị vi phạm** |
| --- | --- | --- |
| **1. Hàm quá dài và phức tạp** | Hàm addNewTaskWithViolations chứa >70 dòng mã, làm quá nhiều việc: kiểm tra dữ liệu, kiểm tra trùng, tạo object, xử lý JSON, hiển thị lỗi. | **SRP** (Single Responsibility Principle - Nguyên lý đơn nhiệm vụ) |
| **2. Lặp lại logic kiểm tra dữ liệu** | Logic kiểm tra tiêu đề rỗng, ngày hợp lệ, mức ưu tiên hợp lệ được viết thủ công trong mỗi lần sử dụng, gây dư thừa mã. | **DRY** (Don't Repeat Yourself - Không lặp lại chính mình) |
| **3. Tạo ID phức tạp không cần thiết** | Sử dụng UUID để tạo ID trong khi có thể chỉ cần một số nguyên tự tăng đơn giản. | **YAGNI** (You Ain't Gonna Need It - Đừng làm thứ chưa cần) |
| **5. Thông báo lỗi trực tiếp trong logic xử lý** | In lỗi trực tiếp thay vì ném exception hoặc trả về kết quả kiểm tra. | **Low Cohesion, Tight Coupling** - logic khó kiểm thử và tái sử dụng |
| **6. Khó kiểm thử đơn vị (Unit Test)** | Vì tất cả xử lý gộp trong 1 hàm lớn, rất khó viết unit test cho từng phần riêng biệt như kiểm tra hợp lệ, kiểm tra trùng. | **Testability** (Khả năng kiểm thử kém) |
| **7. Vi phạm nguyên lý mở rộng – đóng** | Muốn mở rộng logic (ví dụ: thêm loại mức độ ưu tiên mới, validate nâng cao) thì phải sửa trực tiếp trong hàm lớn. | **OCP** (Open-Closed Principle - Mở để mở rộng, đóng để sửa đổi) |

3. Giải thích các nguyên tắc đã áp dụng

**- SRP** (Single Responsibility Principle – Nguyên lý đơn nhiệm vụ):

Mỗi hàm chỉ nên thực hiện **một công việc duy nhất**, nhằm dễ hiểu, dễ bảo trì, và dễ kiểm thử.

**- DRY** (Don't Repeat Yourself – Không lặp lại chính mình):

Tránh lặp lại cùng một đoạn mã nhiều lần ở các vị trí khác nhau trong chương trình.

**- YAGNI** (You Ain’t Gonna Need It – Đừng làm thứ chưa cần):

Không thêm các chức năng, thành phần khi chưa thật sự cần thiết.

**- KISS** (Keep It Simple, Stupid – Giữ mọi thứ đơn giản):

Code càng đơn giản, càng dễ hiểu và dễ bảo trì.

**- Separation of Concerns** (Tách biệt mối quan tâm):

Chia nhỏ chương trình thành các phần với mục đích rõ ràng, không chồng chéo.

**- OCP** (Open-Closed Principle – Mở để mở rộng, đóng để sửa đổi):

Code nên dễ dàng **mở rộng** mà **không cần chỉnh sửa phần đã có**.

**- Testability** (Khả năng kiểm thử):

Code cần được thiết kế sao cho dễ viết **unit test**.

**4. Trình bày các bước refactor cụ thể**

* **Bước 1:** Tách hàm kiểm tra đầu vào validateInputs(...)
* **Bước 2:** Tách hàm kiểm tra trùng lặp isDuplicate(...)
* **Bước 3:** Tách hàm tạo đối tượng nhiệm vụ createTask(...)
* **Bước 4:** Tách hàm đọc và ghi cơ sở dữ liệu loadTasksFromDb() và saveTasksToDb()
* **Bước 5:** Loại bỏ recurrence\_pattern và giữ lại is\_recurring đơn giản
* **Bước 6:** Đơn giản hóa hàm chính addNewTask(...) chỉ gọi các hàm phụ

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

| **Thay đổi** | **Lý do thực hiện** |
| --- | --- |
| **Tách hàm validateInputs(...)** | Giúp gom toàn bộ logic kiểm tra đầu vào vào một nơi, dễ bảo trì, dễ tái sử dụng và dễ viết unit test. |
| **Tách hàm isDuplicate(...)** | Giúp tránh lặp lại mã kiểm tra trùng lặp và dễ kiểm tra logic so khớp hơn. |
| **Tách hàm createTask(...)** | Làm rõ chức năng tạo nhiệm vụ, tránh để hàm chính xử lý quá nhiều chi tiết. |
| **Tách loadTasksFromDb() và saveTasksToDb()** | Phân tách phần xử lý dữ liệu và logic nghiệp vụ, giúp mã gọn hơn và dễ thay thế nếu thay đổi cơ chế lưu trữ. |
| **Loại bỏ recurrence\_pattern và giữ is\_recurring đơn giản** | Tránh thêm các chức năng chưa sử dụng (theo nguyên tắc YAGNI), giữ code ngắn gọn, dễ kiểm soát. |

6. So sánh đoạn code trước và sau refactor

| **Tiêu chí** | **Trước refactor (addNewTaskWithViolations)** | **Sau refactor (addNewTask)** |
| --- | --- | --- |
| **Độ dài hàm chính** | ~70 dòng mã | ~15 dòng mã |
| **Chức năng trong một hàm** | Làm mọi thứ: kiểm tra, tạo task, đọc/ghi file | Chỉ điều phối các hàm phụ |
| **Khả năng tái sử dụng** | Không thể tái sử dụng kiểm tra dữ liệu hay tạo task | Có thể tái sử dụng validateInputs, isDuplicate, createTask |
| **Tính dễ đọc** | Khó theo dõi vì xử lý lẫn lộn | Dễ hiểu, mỗi phần xử lý nằm trong hàm riêng |
| **Khả năng kiểm thử (Testability)** | Khó viết unit test vì logic dính chặt | Có thể viết test cho từng hàm nhỏ độc lập |
| **Tuân thủ nguyên tắc SOLID/KISS/DRY/YAGNI** | Vi phạm nhiều nguyên tắc | Tuân thủ tốt, rõ ràng và có cấu trúc |
| **Tính mở rộng** | Phải sửa trực tiếp trong hàm lớn | Có thể mở rộng dễ dàng bằng cách chỉnh các hàm nhỏ |

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.time.format.DateTimeParseException;

import java.util.UUID;

import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

import org.json.simple.parser.ParseException;

public class PersonalTaskManager {

private static final String DB\_FILE\_PATH = "tasks\_database.json";

private static final DateTimeFormatter DATE\_FORMATTER = DateTimeFormatter.ofPattern("yyyy-MM-dd");

private JSONArray loadTasksFromDb() {

JSONParser parser = new JSONParser();

try (FileReader reader = new FileReader(DB\_FILE\_PATH)) {

Object obj = parser.parse(reader);

if (obj instanceof JSONArray) {

return (JSONArray) obj;

}

} catch (IOException | ParseException e) {

System.err.println("Lỗi khi đọc file database: " + e.getMessage());

}

return new JSONArray();

}

private void saveTasksToDb(JSONArray tasksData) {

try (FileWriter file = new FileWriter(DB\_FILE\_PATH)) {

file.write(tasksData.toJSONString());

file.flush();

} catch (IOException e) {

System.err.println("Lỗi khi ghi vào file database: " + e.getMessage());

}

}

private boolean validateInputs(String title, String dueDateStr, String priority) {

if (title == null || title.trim().isEmpty()) {

System.out.println("Lỗi: Tiêu đề không được để trống.");

return false;

}

if (dueDateStr == null || dueDateStr.trim().isEmpty()) {

System.out.println("Lỗi: Ngày đến hạn không được để trống.");

return false;

}

try {

LocalDate.parse(dueDateStr, DATE\_FORMATTER);

} catch (DateTimeParseException e) {

System.out.println("Lỗi: Ngày đến hạn không hợp lệ. Định dạng đúng: YYYY-MM-DD.");

return false;

}

String[] validPriorities = {"Thấp", "Trung bình", "Cao"};

for (String p : validPriorities) {

if (p.equals(priority)) return true;

}

System.out.println("Lỗi: Mức độ ưu tiên không hợp lệ. Chọn từ: Thấp, Trung bình, Cao.");

return false;

}

private boolean isDuplicate(JSONArray tasks, String title, String dueDateStr) {

for (Object obj : tasks) {

JSONObject task = (JSONObject) obj;

if (task.get("title").toString().equalsIgnoreCase(title)

&& task.get("due\_date").toString().equals(dueDateStr)) {

return true;

}

}

return false;

}

private JSONObject createTask(String title, String description, String dueDateStr,

String priority, boolean isRecurring) {

JSONObject task = new JSONObject();

task.put("id", UUID.randomUUID().toString());

task.put("title", title);

task.put("description", description);

task.put("due\_date", dueDateStr);

task.put("priority", priority);

task.put("status", "Chưa hoàn thành");

task.put("created\_at", LocalDateTime.now().format(DateTimeFormatter.ISO\_DATE\_TIME));

task.put("last\_updated\_at", LocalDateTime.now().format(DateTimeFormatter.ISO\_DATE\_TIME));

task.put("is\_recurring", isRecurring);

return task;

}

public JSONObject addNewTask(String title, String description,

String dueDateStr, String priority,

boolean isRecurring) {

if (!validateInputs(title, dueDateStr, priority)) return null;

JSONArray tasks = loadTasksFromDb();

if (isDuplicate(tasks, title, dueDateStr)) {

System.out.printf("Lỗi: Nhiệm vụ '%s' đã tồn tại với cùng ngày đến hạn.%n", title);

return null;

}

JSONObject task = createTask(title, description, dueDateStr, priority, isRecurring);

tasks.add(task);

saveTasksToDb(tasks);

System.out.printf("✅ Đã thêm nhiệm vụ '%s' với ID: %s%n", title, task.get("id"));

return task;

}

public static void main(String[] args) {

PersonalTaskManager manager = new PersonalTaskManager();

manager.addNewTask("Mua sách", "Sách Công nghệ phần mềm", "2025-07-20", "Cao", false);

manager.addNewTask("Mua sách", "Sách Công nghệ phần mềm", "2025-07-20", "Cao", false);

manager.addNewTask("Tập thể dục", "Tập gym 1 tiếng", "2025-07-21", "Trung bình", true);

manager.addNewTask("", "Không có tiêu đề", "2025-07-22", "Thấp", false);

}

}