Đoạn code quản lý danh sách nhiệm vụ cá nhân bằng cách lưu trữ và đọc dữ liệu từ file JSON (tasks\_database.json). Mục tiêu của hàm addNewTaskWithViolations là thêm một nhiệm vụ mới vào cơ sở dữ liệu nếu hợp lệ.

**Các vấn đề và nguyên tắc vi phạm**

|  |  |  |
| --- | --- | --- |
| **Vấn đề** | **Vi phạm nguyên tắc** | **Giải thích** |
| Lặp lại đoạn code loadTasksFromDb() và saveTasksToDb() nhiều lần | **DRY** (Don't Repeat Yourself) | Gây dư thừa và khó bảo trì |
| Kiểm tra dữ liệu đầu vào, định dạng ngày, mức độ ưu tiên… gộp chung vào 1 hàm dài | **KISS** (Keep It Simple, Stupid) | Làm hàm addNewTaskWithViolations dài, phức tạp |
| Tạo UUID, thuộc tính is\_recurring, recurrence\_pattern mà chưa có logic xử lý | **YAGNI** (You Aren't Gonna Need It) | Thêm tính năng chưa cần thiết gây dư thừa |

**Nguyên tắc refactor đã áp dụng**

* **KISS**: Chia nhỏ hàm dài thành các hàm ngắn, dễ hiểu, có nhiệm vụ cụ thể.
* **DRY**: Tránh lặp lại các đoạn kiểm tra hoặc logic giống nhau.
* **YAGNI**: Lược bỏ các thành phần không cần thiết cho chức năng hiện tại.

**Các bước refactor cụ thể**

**Bước 1: Tách hàm kiểm tra đầu vào**

* Tạo hàm validateInput(title, dueDateStr, priorityLevel).

**Bước 2: Tách hàm kiểm tra trùng lặp**

* Tạo hàm isDuplicateTask(tasks, title, dueDate).

**Bước 3: Tách hàm tạo đối tượng task**

* Tạo hàm createTask(...) để sinh JSONObject của nhiệm vụ.

**Bước 4: Lược bỏ các thành phần YAGNI**

* Loại bỏ is\_recurring, recurrence\_pattern, UUID (nếu không cần thiết).

Lý do cho từng thay đổi

|  |  |
| --- | --- |
| **Thay đổi** | **Lý do** |
| Tách hàm kiểm tra dữ liệu | Tăng tính tái sử dụng và giúp addTask() ngắn gọn |
| Loại bỏ UUID | Chưa cần đến UUID, có thể dùng chỉ số tự tăng hoặc ID đơn giản |
| Loại bỏ is\_recurring | Chưa có xử lý cho nhiệm vụ lặp lại, thêm vào gây dư thừa |
| Đưa load và save ra riêng | Tránh trùng lặp code (DRY) |

**So sánh trước và sau refactor**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Trước refactor** | **Sau refactor** |
| Độ dài hàm | Dài, trên 50 dòng | Ngắn gọn, <20 dòng |
| Mức độ rõ ràng | Khó đọc, nhiều nhiệm vụ trong 1 hàm | Mỗi hàm một nhiệm vụ duy nhất |
| Khả năng tái sử dụng | Kém | Cao hơn nhờ hàm chuyên biệt |
| Tuân thủ nguyên tắc OOP | Kém | Tốt hơn, hướng đến Single Responsibility |

**Refactor code**

public class PersonalTaskManagerRefactored {

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

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

private JSONArray loadTasks() {

JSONParser parser = new JSONParser();

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

Object obj = parser.parse(reader);

return (JSONArray) obj;

} catch (Exception e) {

return new JSONArray();

}

}

private void saveTasks(JSONArray tasks) {

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

file.write(tasks.toJSONString());

} catch (IOException e) {

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

}

}

private boolean isValidPriority(String priority) {

return List.of("Thấp", "Trung bình", "Cao").contains(priority);

}

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

for (Object obj : tasks) {

JSONObject task = (JSONObject) obj;

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

task.get("due\_date").equals(dueDate.format(DATE\_FORMATTER))) {

return true;

}

}

return false;

}

private JSONObject createTask(String title, String description, LocalDate dueDate, String priority) {

JSONObject task = new JSONObject();

task.put("id", String.valueOf(System.currentTimeMillis()));

task.put("title", title);

task.put("description", description);

task.put("due\_date", dueDate.format(DATE\_FORMATTER));

task.put("priority", priority);

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

String now = LocalDateTime.now().format(DateTimeFormatter.ISO\_DATE\_TIME);

task.put("created\_at", now);

task.put("last\_updated\_at", now);

return task;

}

public JSONObject addTask(String title, String description, String dueDateStr, String priority) {

if (title == null || title.isBlank()) {

System.out.println("Lỗi: Tiêu đề không hợp lệ.");

return null;

}

LocalDate dueDate;

try {

dueDate = LocalDate.parse(dueDateStr, DATE\_FORMATTER);

} catch (DateTimeParseException e) {

System.out.println("Lỗi: Ngày đến hạn sai định dạng.");

return null;

}

if (!isValidPriority(priority)) {

System.out.println("Lỗi: Mức độ ưu tiên không hợp lệ.");

return null;

}

JSONArray tasks = loadTasks();

if (isDuplicateTask(tasks, title, dueDate)) {

System.out.println("Lỗi: Nhiệm vụ đã tồn tại.");

return null;

}

JSONObject task = createTask(title, description, dueDate, priority);

tasks.add(task);

saveTasks(tasks);

System.out.println("Nhiệm vụ được thêm thành công!");

return task;

}

public static void main(String[] args) {

PersonalTaskManagerRefactored manager = new PersonalTaskManagerRefactored();

manager.addTask("Mua sách", "Sách học", "2025-07-20", "Cao");

}

}

https://github.com/NotPeterDang/refactor-code/tree/main