Tài liệu Refactor Code: PersonalTaskManager

# 1. Mục tiêu của đoạn code

Đoạn mã xây dựng một ứng dụng quản lý nhiệm vụ cá nhân đơn giản (Personal Task Manager) bằng Java. Người dùng có thể:  
- Thêm task mới (với tiêu đề, mô tả, ngày hết hạn)  
- Hiển thị tất cả các task  
- Xoá task theo ID  
- Lưu dữ liệu task vào tệp JSON

# 2. Các vấn đề trong đoạn code ban đầu

|  |  |  |
| --- | --- | --- |
| Nguyên tắc | Vi phạm | Giải thích |
| S - SRP | Vi phạm | Lớp xử lý UI, dữ liệu và logic nghiệp vụ cùng lúc |
| O - OCP | Khó mở rộng | Thêm tính năng cần sửa mã gốc |
| L - LSP | Không áp dụng | Không có kế thừa/giao diện |
| I - ISP | Không áp dụng | Chưa có interface |
| D - DIP | Vi phạm | Phụ thuộc trực tiếp vào FileReader/FileWriter |
| KISS | Lặp lại logic | Chưa tách hàm riêng, logic lặp |

# 3. Các nguyên tắc đã áp dụng

• SRP: Tách các hàm loadTasks, saveTasks, createTask riêng biệt

• KISS: Giữ giao diện console đơn giản, tách logic xử lý

• UUID: Thay cho System.currentTimeMillis() để tạo ID duy nhất

• Xử lý lỗi: Bổ sung try-catch khi ghi file, parse JSON

# 4. Các bước refactor cụ thể

Bước 1: Tách logic tạo task bằng hàm createTask(...)

Bước 2: Tách hàm load/save JSON dùng chung

Bước 3: Thay System.currentTimeMillis bằng UUID

Bước 4: Hiển thị ID khi thêm task để người dùng dễ sử dụng

Bước 5: Giữ menu đơn giản, tránh thêm logic chưa cần thiết

# 5. So sánh trước và sau refactor

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Trước refactor | Sau refactor |
| Chức năng | Lẫn logic và giao diện | Tách chức năng rõ ràng |
| Tính mở rộng | Khó mở rộng | Dễ mở rộng bằng hàm riêng |
| Tái sử dụng | Không có | Tái sử dụng được hàm |
| Đơn giản (KISS) | Có lặp | Tối giản và rõ ràng |
| Quản lý lỗi | Ít kiểm tra | Có xử lý lỗi cơ bản |
| ID duy nhất | Có thể trùng | Sử dụng UUID bảo đảm |

# 6. Tổng kết

Việc refactor giúp mã dễ đọc, dễ bảo trì, và phù hợp hơn với nguyên tắc thiết kế phần mềm như KISS và SRP. Ngoài ra, nó cũng mở đường cho việc mở rộng thêm như cập nhật trạng thái task, lọc task, v.v.

**Refactor code**

import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

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.List;

import java.util.UUID;

public class PersonalTaskManagerRefactored {

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

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

// Load tasks from JSON file

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();

}

}

// Save tasks to JSON file

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());

}

}

// Check if the priority level is valid

private boolean isValidPriority(String priority) {

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

}

// Check for duplicate task based on title and due date

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;

}

// Create a new task object

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

JSONObject task = new JSONObject();

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

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;

}

// Add a new task with validation

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 với ID: " + task.get("id"));

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/Do-An-Cuoi-Mon-CNPM-G2