Trường Đại học Khoa học tự nhiên - ĐHQG TPHCM

Khoa Công Nghệ Thông Tin

BÁO CÁO ĐÁNH GIÁ CẢI TIẾN

Thiết kế phần mềm

Ngày 28 tháng 2 năm 2025

Lớp: 22CLC08

Sinh Viên

22127451 - Phan Thị Tường Vi

Giáo viên hướng dẫn

Thầy Trần Duy Thảo

Mục lục

1 Khó khăn khi test

1. Phụ thuộc quá nhiều vào Singleton và Instance Toàn Cục
   - Ví dụ: StudentRepository::getInstance() khiến việc test khó khăn vì không thể thay thế bằng giả lập (mock repository).
   - Khi chạy test, dữ liệu có thể bị ảnh hưởng bởi trạng thái trước đó, làm giảm tính ổn định của bài test.
2. Khó kiểm soát trạng thái của dữ liệu khi test
   - Các bài test có thể gặp lỗi do dữ liệu trong StudentRepository không được reset hoặc không thể kiểm soát dữ liệu đầu vào.
   - Hàm clearStudentRepository() được sử dụng để reset dữ liệu, nhưng cách này vẫn chưa tối ưu và phụ thuộc vào logic cụ thể của repository.
3. Không thể dễ dàng kiểm thử lỗi hoặc điều kiện biên
   - Các hàm như repo.addStudent(s) không trả về kết quả rõ ràng về lỗi (ví dụ: trùng MSSV) mà chỉ hoạt động âm thầm.
   - Không có cơ chế kiểm soát lỗi ngoại lệ trong các bài test.
4. Phụ thuộc vào file I/O làm chậm quá trình test
   - Các bài test như testRecordIO_CSV() và testRecordIO_JSON() cần thao tác với file, làm bài test chạy chậm hơn.
   - Việc đọc/ghi file có thể gây lỗi khi chạy test trên môi trường khác nhau

2 Cải thiện thiết kế

1. Thay thế Singleton bằng Dependency Injection (DI)
   - Thay vì gọi StudentRepository::getInstance(), truyền một đối tượng StudentRepository vào các lớp cần sử dụng.
   - Điều này giúp thay thế repository bằng mock khi test, tránh thay đổi trạng thái dữ liệu ngoài ý muốn.
2. Tách biệt logic kiểm tra dữ liệu và thao tác dữ liệu
   - Tạo một class StudentService để chứa các logic kiểm tra (ví dụ: kiểm tra trùng lặp MSSV), thay vì đặt logic này trong StudentRepository.
   - Điều này giúp test dễ dàng hơn, vì có thể mock repository mà không ảnh hưởng đến các kiểm tra logic.
3. Cải tiến phương thức thêm sinh viên để dễ kiểm thử
   - Hiện tại, repo.addStudent(s) không trả về giá trị. Nên cải tiến để trả về bool hoặc enum giúp test dễ xác định lỗi.

- Ví dụ:

```
enum class AddStudentResult { SUCCESS, DUPLICATE_ID, INVALID_DATA };
AddStudentResult StudentRepository::addStudent(const Student& s);
```

4. Giảm phụ thuộc vào File I/O trong test

- Tạo lớp MockRecordIO để test mà không cần ghi file thật.
- Ví dụ:

```
class MockRecordIO : public RecordIO {
public:
    bool exportToCSV(const std::string& filename, const std::vector<std::vector<std::s
        storedData = records;
        return true;
    }
    std::vector<std::vector<std::string>> importFromCSV(const std::string& filename) c
        return storedData;
    }
private:
    std::vector<std::vector<std::string>> storedData;
};
```