SRP & DRY

1. SRP (Single Responsibility Principle)

1.1.Định nghĩa:

SRP là nguyên tắc trong SOLID quy định rằng **một class chỉ nên có một lý do để thay đổi**. Nghĩa là một class chỉ nên đảm nhiệm một nhiệm vụ duy nhất.

1.2. Lợi ích của SRP:

- Code dễ bảo trì hơn do mỗi class chỉ có một nhiệm vụ.
- Dễ mở rộng vì thay đổi trong một chức năng không ảnh hưởng đến phần khác.
- Giảm sự phụ thuộc giữa các module, giúp code dễ hiểu hơn.

1.3. Dấu hiệu vi phạm SRP:

- Class có quá nhiều chức năng khác nhau (ví dụ: xử lý dữ liệu, giao diện, lưu trữ, tính toán trong cùng một class).
- Một class thay đổi vì nhiều lý do khác nhau.

1.4. Ví dụ vi phạm SRP và cách khắc phục trong đồ án:

Vi phạm:

class Student {

```
public:
    //setter
    //getter
    void display(); // Hiển thị thông tin sinh viên
    //Lưu, lấy dữ liệu từ file csv, json
    static void saveStudentToCSV(const string &filename);
```

```
    static void loadStudentFromCSV(const string &filename);
    static void exportStudentToJSON(const string &filename);
    static void importStudentFromJSON(const string &filename);
};
```

Vi phạm SRP vì một class Student thực hiện nhiều trách nhiệm.

Khắc phục: chia ra thành class Student và StudentExporter:

```
class Student {
public:
    //setter
    //getter
    void display(); // Hiển thị thông tin sinh viên
};
```

```
class StudentExporter {
public:
   static string toCSV(const Student &student);
   static Student fromCSV(const string &csvLine);
   static void saveStudentToCSV(const string &filename);
   static void loadStudentFromCSV(const string &filename);
   static void exportStudentToJSON(const string &filename);
   static void importStudentFromJSON(const string &filename);
```

};

2. SRP (Don’t repeat yourself)

2.1. Định nghĩa:

DRY quy định rằng **không nên lặp lại logic trong nhiều phần khác nhau của code**.

2.2. Lợi ích của DRY:

- Giảm sự dư thừa trong code, giúp code dễ đọc và dễ bảo trì hơn.
- Khi cần thay đổi logic, chỉ cần sửa ở một nơi duy nhất.
- Giúp tránh lỗi không đồng nhất khi chỉnh sửa code.

2.3. Dấu hiệu vi phạm DRY:

- Các đoạn code giống nhau hoặc rất giống nhau xuất hiện ở nhiều nơi.
- Hàm hoặc class thực hiện cùng một nhiệm vụ nhưng được viết lại nhiều lần.

2.4. Ví dụ vi phạm SRP và cách khắc phục trong đồ án và cách khắc phục:

Vi phạm khi tách các hàm checkDependencies ra thành 3 hàm con:

```
// Check department dependencies
bool hasDepartmentDependencies(const string& department) {
    return hasDependencies([&department](const Student& s) {
        return s.getDepartment() == department;
    });
}
//Check status dependencies
bool hasStatusDependencies(const string& status) {
```

```
    return hasDependencies([&status](const Student& s) {
        return s.getStatus() == status;
    });
}
//Check program dependencies
bool hasProgramDependencies(const string& program) {
    return hasDependencies([&program](const Student& s) {
        return s.getProgram() == program;
    });
}
```

3 Hàm trên có cùng logic nhưng bị lặp lại

Khắc phục: gộp 3 hàm lại thành một, đặt một biến type để kiểm tra loại dependency:

```
bool hasDependencies(const string& dependency, const string& type) {
    return any_of(students.begin(), students.end(), [&dependency, &type](const Student& s) {
        if (type == "program") {
            return s.getProgram() == dependency;
        } else if (type == "department") {
            return s.getDepartment() == dependency;
        } else if (type == "status") {
```

```
            return s.getStatus() == dependency;
        }
        return false;
    });
}
```