주소록 프로젝트 C++변환#3

과제의 목적	디자인 패턴 및 UML 배우기
깃 주소	https://github.com/YCK1204/Bootcamp-Subjects/tree/main/Subject%233

목차

과제 개요

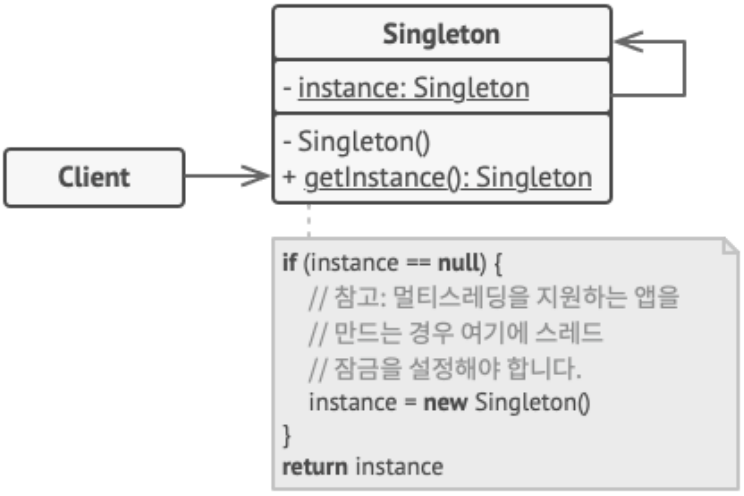
1. 이전 과제(C언어 주소록) C++로 변환
2. 디자인 패턴 3가지 이상 적용(singleton, observer, 그외 선택)
3. UML 다이어그램 작성
4. UI, DATA, CONTROL 각 기능에 맞는 클래스 구현
5. UI - observer / data, control - singleton 패턴 적용

디자인 패턴 정리

1. 싱글톤 패턴(Singleton)
2. 옵저버 패턴(Observer)
3. 상태 패턴(State)

1. 싱글톤 패턴(Singleton)

- 클래스에 인스턴스가 하나만 있도록 하면서 이 인스턴스에 대한 전역 접근(액세스) 지점을 제공하는 생성 디자인 패턴
- 문제점
  - 클래스에 인스턴스가 하나만 있도록 한다.
  - 해당 인스턴스에 대한 전역 접근 지점을 제공
- 해결책
  - 다른 객체들이 싱글턴 클래스와 함께 `new` 연산자를 사용하지 못하도록 디폴트 생성자를 비공개로 설정
  - 생성자 역할을 하는 정적 생성 메서드를 설정



2. 옵저버 패턴(Observer)

- Listener 패턴, 관찰자 패턴이라고도 불린다.
- 일부 객체들이 다른 객체들에게 자신의 상태 변경에 대해 알릴 수 있는 행동 디자인 패턴
- 구독자 인터페이스를 구현하는 모든 객체에 대한 이러한 이벤트들을 구독 및 구독 취소하는 방법을 제공
- 장점
  - OCP(개방-폐쇄 원칙), 기존 코드를 변경하지 않고 새로운 주제나 옵저버를 추가할 수 있다.
  - 주제와 옵저버의 관계가 느슨해서 주제가 변경되어도 옵저버에 영향을 주지 않는다.**(느슨한 결합, Loose Coupling)**

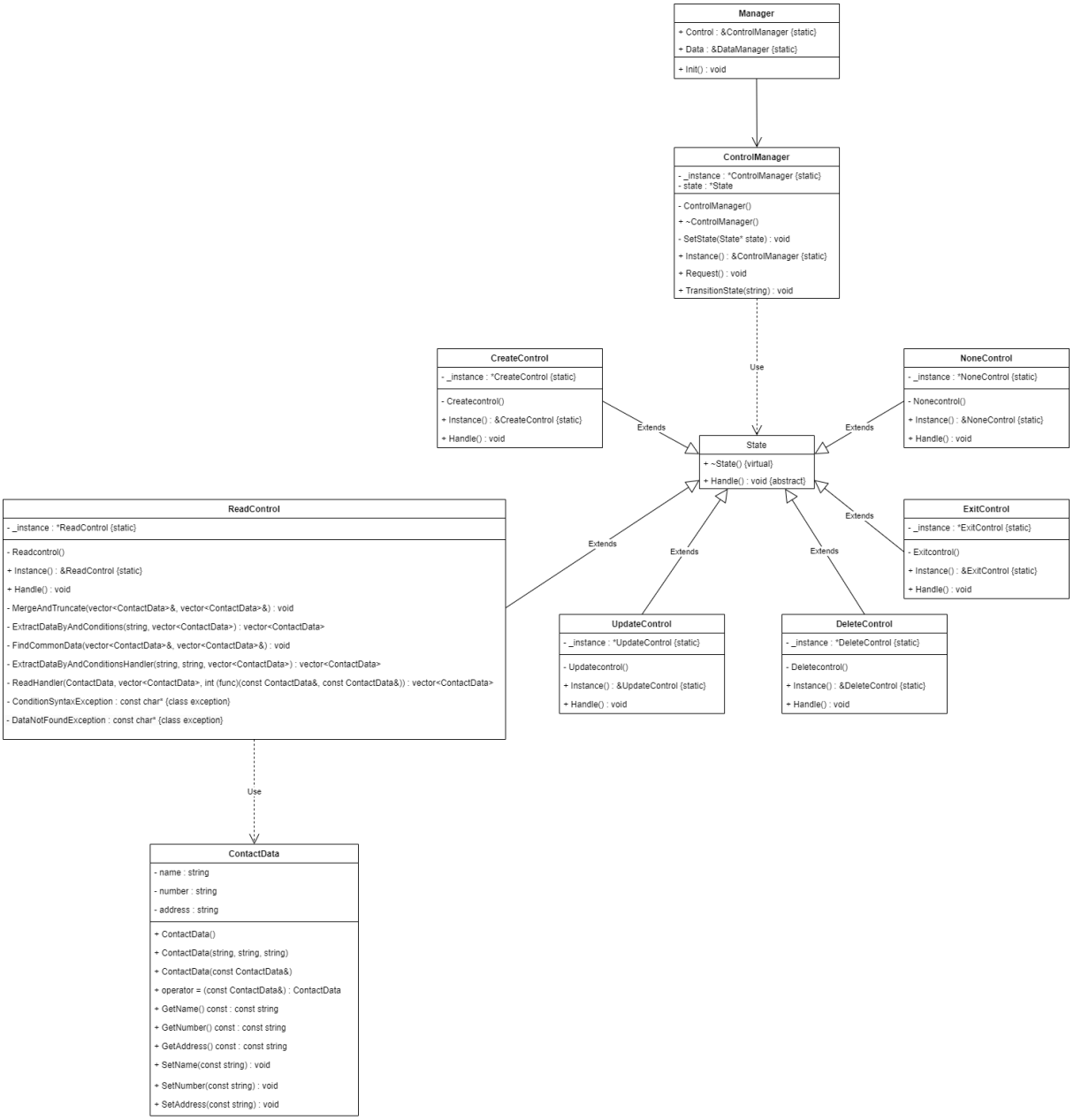
    - 런타임에 새로운 옵저버를 추가하거나 기존 옵저버를 제거할 수 있다. **(확장성, Scalability)**
    - 주제와 옵저버는 독립적인 모듈로 구성된다. **(Modularity)**
    - 주제는 옵저버가 특정 인터페이스를 구현하는 것만 알고, 옵저버를 추가한다고 해서 주제를 변경할 필요가 없다.
- 단점
    - 주제 객체의 상태가 변경될 때마다 모든 옵저버에게 알림을 보내야 하기에 처리 속도가 떨어질 수 있다.
    - 주제와 옵저버 간의 순환 참조가 발생할 수 있다.

3. 상태 패턴(State)

- 객체의 내부 상태에 따라 스스로 행동을 변경할 수 있게 허가하는 패턴
- 마치 자신의 클래스를 바꾸는 것처럼 행동할 수 있음
- 상태 패턴은 일반적으로 C++에서 대규모 `switch` 기반 상태 머신들을 객체들로 변환하는 데 사용

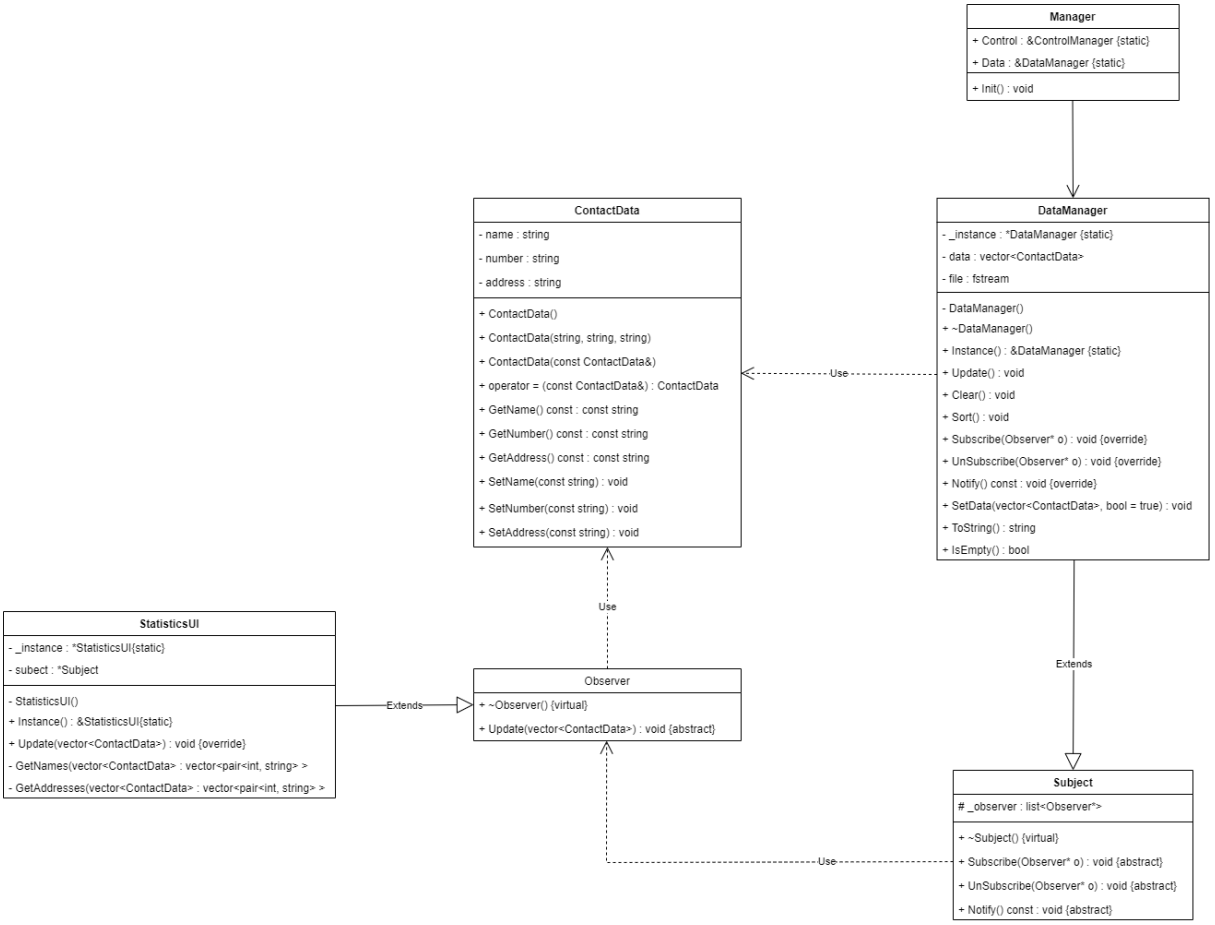
UML 다이어그램과 로직 설명

1. ControlManager Class Diagram



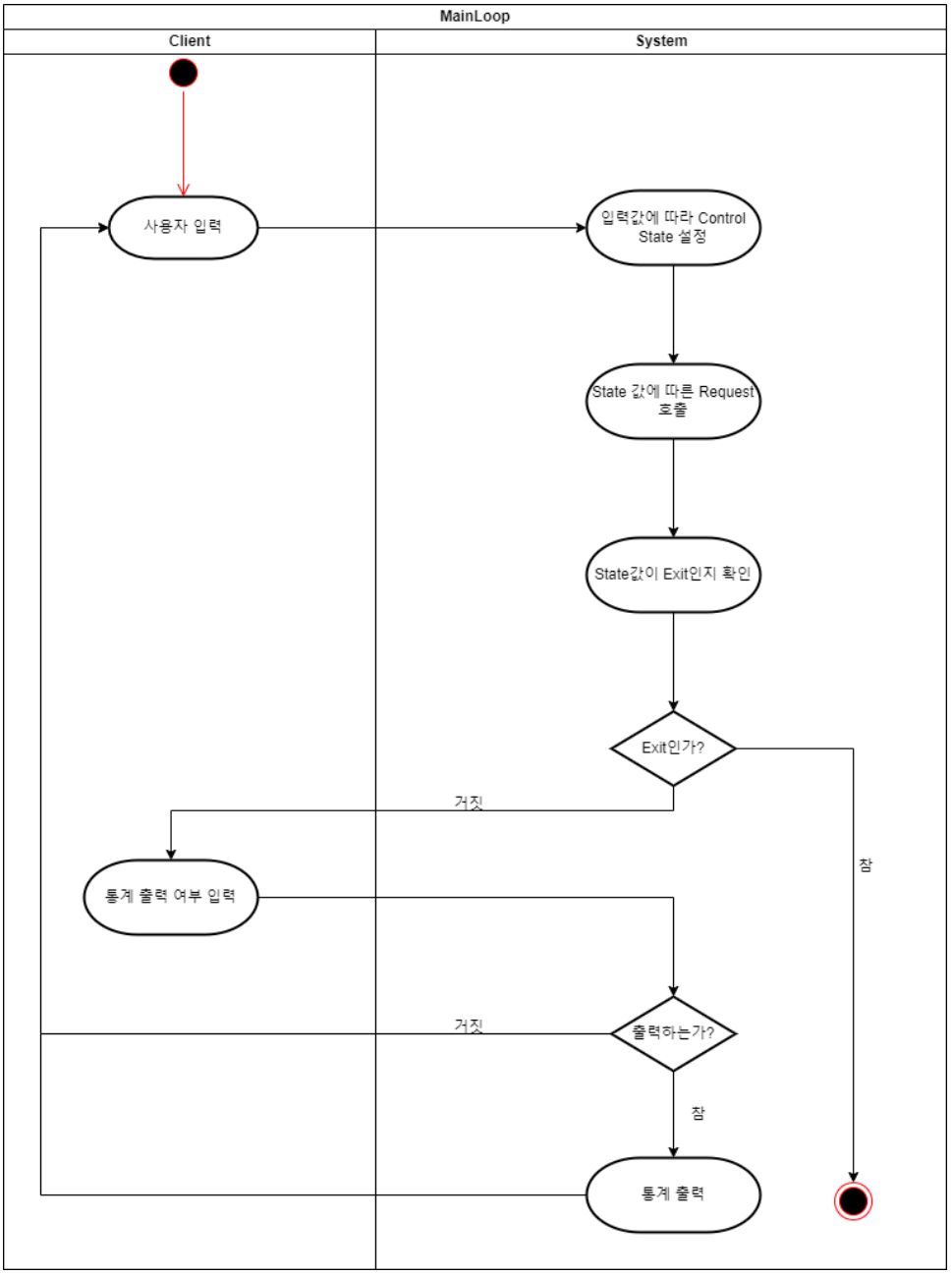
- 기존 함수 포인터로 사용자 입력값에 따라 행동하는것을 ControlManager를 통해 제어
- 클래스 사용처가 한곳이기에 Singleton 패턴 적용
- 각 행동 클래스들을 State 패턴을 적용해 각 행동들을 클래스별로 분리해서 정의
- 주요 클래스인 ControlManager를 Manager 클래스에서 관리(코드 조직화를 위함)

2. DataManager Class Diagram



- 기존 FileSystem 관련된 함수들을 DataManager 클래스를 통해 제어
- DataManager에 Observer 패턴을 적용해 CUD(Create, Update, Delete) 행동일 때 알림을 받아
  구독자인 StaticUI 클래스를 자동 갱신 (통계 출력)
- 클래스 사용처가 한곳이기에 Singleton 패턴 적용
- 주요 클래스인 DataManager를 Manager 클래스에서 관리(코드 조직화를 위함)

3. MainLoop Activity Diagram



1. 사용자 입력에 따라 ControlManager의 State 변경 후 State 클래스 함수의 Handle 함수 호출
2. Handle함수 호출 후 특정 행동에 관해서 DataManager가 알림 착신
3. DataManager에 속해있는 Observer들 갱신 (통계 출력)

4. 그 외 로직

- 이름, 전화번호, 주소를 멤버 변수로 갖는 구조체를 Class화(ContactData)
- .csv 파일로 데이터 관리
- 데이터 검색에 이진 탐색 활용

- 특정 데이터 범위를 리스트가 아닌 이진 탐색을 활용해 인덱스 범위를 찾음
- 기존 qsort 활용 코드를 std::sort로 변경 (데이터 정렬)
- Read 행동할 때 오류 발생 시 std::exception.what() throw()
- 기존 정적 UI를 StaticUI 클래스로 관리

문제점 및 해결

- Create 할 때 파일이 일정 크기 이상인 경우 filestream buffering 문제로 인해 데이터가 짤리는 현상
  - file.flush()를 통해 버퍼에 남아있는 데이터 파일에 출력
- std::sort 세 번째 인자인 함수에 int 반환 함수를 넣었을 때 오류 발생
  - boolean 으로 반환형 교체 및 string 비교 시 부등호로 비교

참고 자료

- https://m.blog.naver.com/jovincicode/222584165619
- https://seulhee030.tistory.com/56
- https://m.blog.naver.com/ruvendix/222038862519
- https://refactoring.guru/ko/design-patterns/singleton
- https://boycoding.tistory.com/109
- https://refactoring.guru/ko/design-patterns/observer/cpp/example
- https://bloodstrawberry.tistory.com/1363
- https://boycoding.tistory.com/110