**게임개발자전문가과정**

**과 목 명 : 게임기초프로그래밍**

**능력단위 : C/C++ 프로그래밍**

**제출일자 : 2022년 07월 26일**

**포트폴리오 : 파일 입출력 및 템플릿을 적용한 학생관리 프로그램 제작**

**작 성 자 : 권성호**

**<제출내역>**

1. **학생 관리 프로그램 구현 프로젝트**
2. **학생 관리 프로그램 구현 분석 및 세부 문서**

|  |
| --- |
|  |

**< Contents >**

1. **프로젝트 소개 및 개요**
2. 프로젝트 소개
3. 프로젝트 주요 기술
4. **프로젝트 설계 및 다이어그램**
5. 클래스 다이어그램
   1. 애플리케이션 프레임워크 다이어그램
      1. 프레임워크 클래스 개요
   2. 직렬화 라이브러리 다이어그램

1-2-1) 직렬화 라이브러리 클래스 개요

* 1. 자료구조 다이어그램

1-3-1) 자료구조 클래스 개요

* 1. 학생관리 다이어그램

1-4-1) 학생관리 클래스 개요

1. 시퀀스 다이어그램

2-1) 앱기동

2-2) 저장

2-3) 로드

2-4) 삭제

2-5) 정렬

2-6) 검색

1. **최종 결과 및 추가 내용**
2. 범용성/유연성/확장성/간결성 고려하여 추가된 내용
   1. 현재 프로젝트 설계(구현 내용)
   2. 향후 개발 내용
3. 최종 결과
4. **프로젝트 소개 및 개요**
5. **프로젝트 소개**

본 프로젝트는 애플리케이션 개발 시 공통된 사항을 구조화 하여 제작하였다. 프로젝트를 개발하면서 최종적인 게임 제작에 사용될 프레임워크 개발의 기초 토대 제작을 목표를 두었다.

또한, 애플리케이션 제작에 사용될 기능을 일반화 및 추상화 하여 범용성 및 확장성을 가질 수 있게 하였다. 각 기능별 Scene을 모듈화 하여 확장성을 가질 수 있었다. 또한 파일 입출력의 기능을 일반화하여 직렬화 객체를 통해 파일 입출력을 쉽게 이용할 수 있게 제작하였다.

실제 위 프레임워크를 사용하여 학생 관리 프로그램을 제작하여 프레임워크의 검증을 하였다.

1. **프로젝트 주요 기술**

* **애플리케이션 구조화**

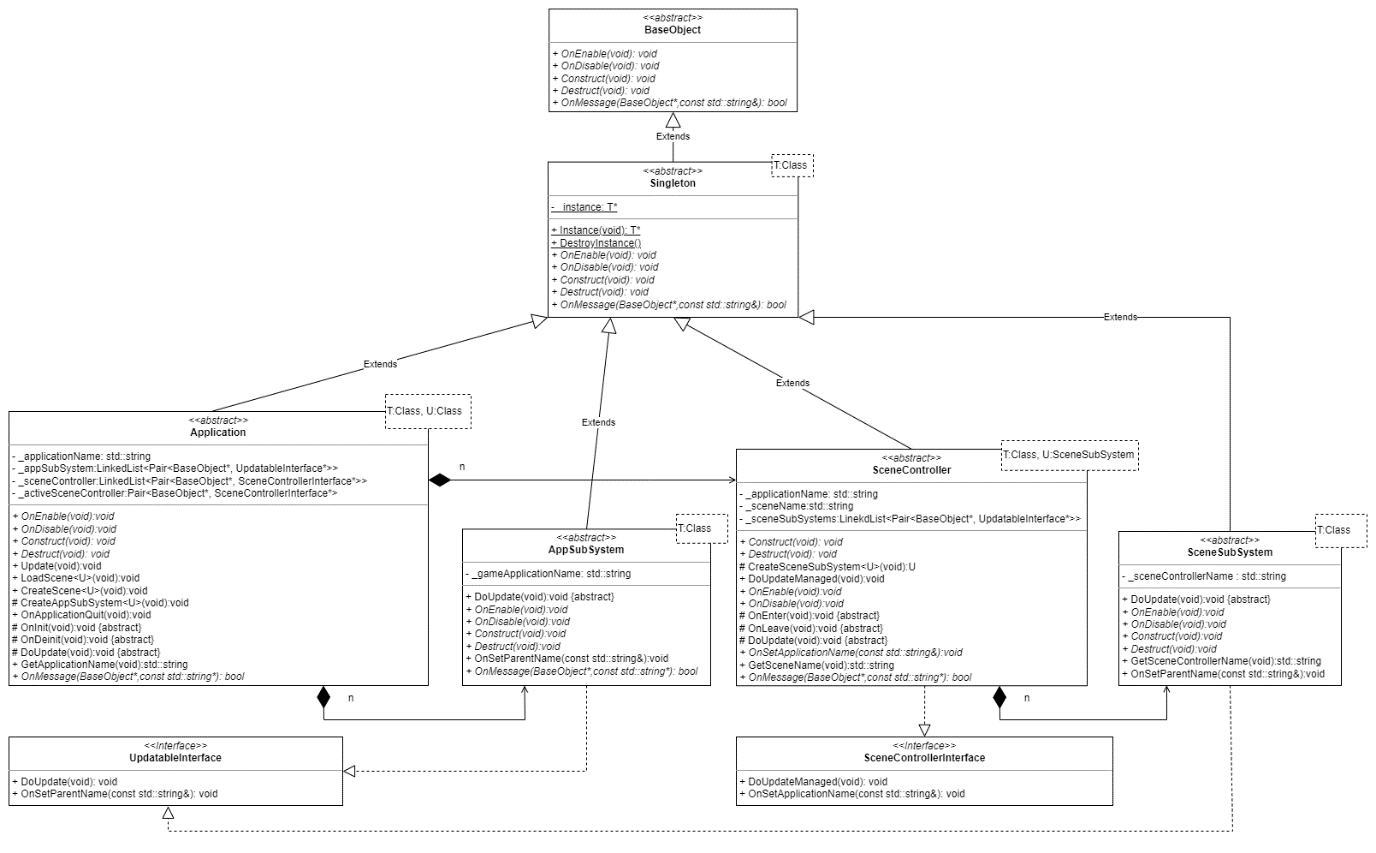
모든 애플리케이션은 공통 화된 구조를 갖는다. 모든 애플리케이션은 각각의 출력(씬), 입력(키보드) , 데이터(파일, 메모리)를 가진다. 이러한 점에 착안해 최종적인 게임 개발에 사용될 프레임워크 제작을 목표로 하여, 씬의 계층화, 데이터 입출력의 지원, 향후 입력 지원이 가능할 수 있는 구조의 뼈대를 제작하였다.

* **직렬화 라이브러리**

C++에서 프로그램 파일 입출력 라이브러리를 한 번 더 일반화하여 개발자들이 편이 사용할 수 있는 직렬화 라이브러리를 만들었다. 파일 입출력 사용할 객체는 해당 직렬화 객체를 상속받아 C++ 파일 입출력 할 때 해야 하는 작업을 간소화하여 손쉽게 파일 입출력을 가능하게 하였다.

* **템플릿 자료구조**

STL의 List와 Pair, Iterator를 모방하여 템플릿 자료구조를 구현하였다. 양방향 연결 리스트를 구현하였으며 삽입, 삭제, 검색을 지원하여, Iterator를 통해 노드의 이동을 랩핑하여 유저들이 손쉽게 연결 리스트를 사용할 수 있게 하였다. 추가로 Pair라는 자료구조를 통해 2개의 데이터를 관리할 수 있게 하였다.

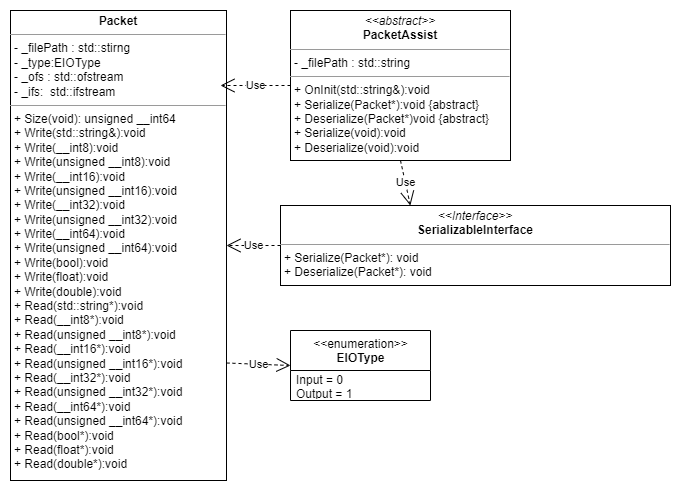
1. **프로젝트 설계 및 다이어그램**
2. **클래스 다이어그램**
   1. **애플리케이션 프레임워크 다이어그램**

[그림 1-1] 프레임워크 다이어그램

1-1-1) 프레임워크 클래스 개요

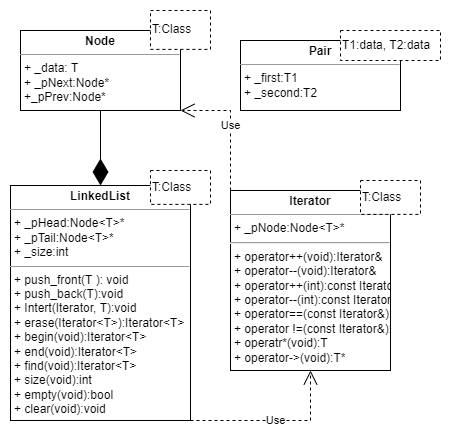
|  |  |  |
| --- | --- | --- |
| **부모클래스/인터페이스** | **클래스** | **내용** |
|  | **BaseObject** | 해당 프레임워크의 기본 클래스  해당 클래스를 상속받는 객체는 App의 상태에 따라 계층적으로 제어를 받게 된다.  - OnEnable : 객체가 활성화 될 때(예 : Scene Enter)  - OnDisable : 객체가 비활성화 될 때 (예 : Scene Leave)  - Construct : 객체의 초기화 값 정의 - Destruct : 객체의 릴리즈 값 정의  - OnMessage : 계층적으로 메시지를 송, 수신 |
| **BaseObject** | **SingleTon** | 유일객체 값 정의  앱에서 전역 적 로 하나만 존재해야 하는 경우 해당 클래스를 상속받는다. |
| **SingleTon** | **Application** | 앱을 총괄하는 객체  해당 객체를 통해 앱의 모든 객체들은 제어받는다  - DoUpdate : 계층적으로 생성된 객체들을 순회 하여 Update |
| **SingleTon/ UpdatableInterface** | **AppSubSytem** | 앱에서 전역적으로 사용되는 객체  시스템적 인(Input, Render, Physics..)등은 Application에 생성 등록 해야 된다. |
| **SingleTon/ SceneControllerInterface** | **SceneController** | 씬을 구성하기 위한 클래스  - OnEnter : 씬 진입 시 호출  - OnLeave : 씬 탈출 시 호출 |
| **SingleTon/**  **UpdatableInterface** | **SceneSubSystem** | 씬에 종속적인 시스템 정의를 위한 클래스 |
|  | **UpdatableInterface** | 일반 업데이트 객체의 인터페이스 |
|  | **SceneControllerInterface** | 씬 관련 컨트롤 객체의 인터페이스 |

* 1. **직렬화 라이브러리 다이어그램**

****[그림 1-2] 직렬화 라이브러리 다이어그램

1-2-1) 직렬화 라이브러리 클래스 개요

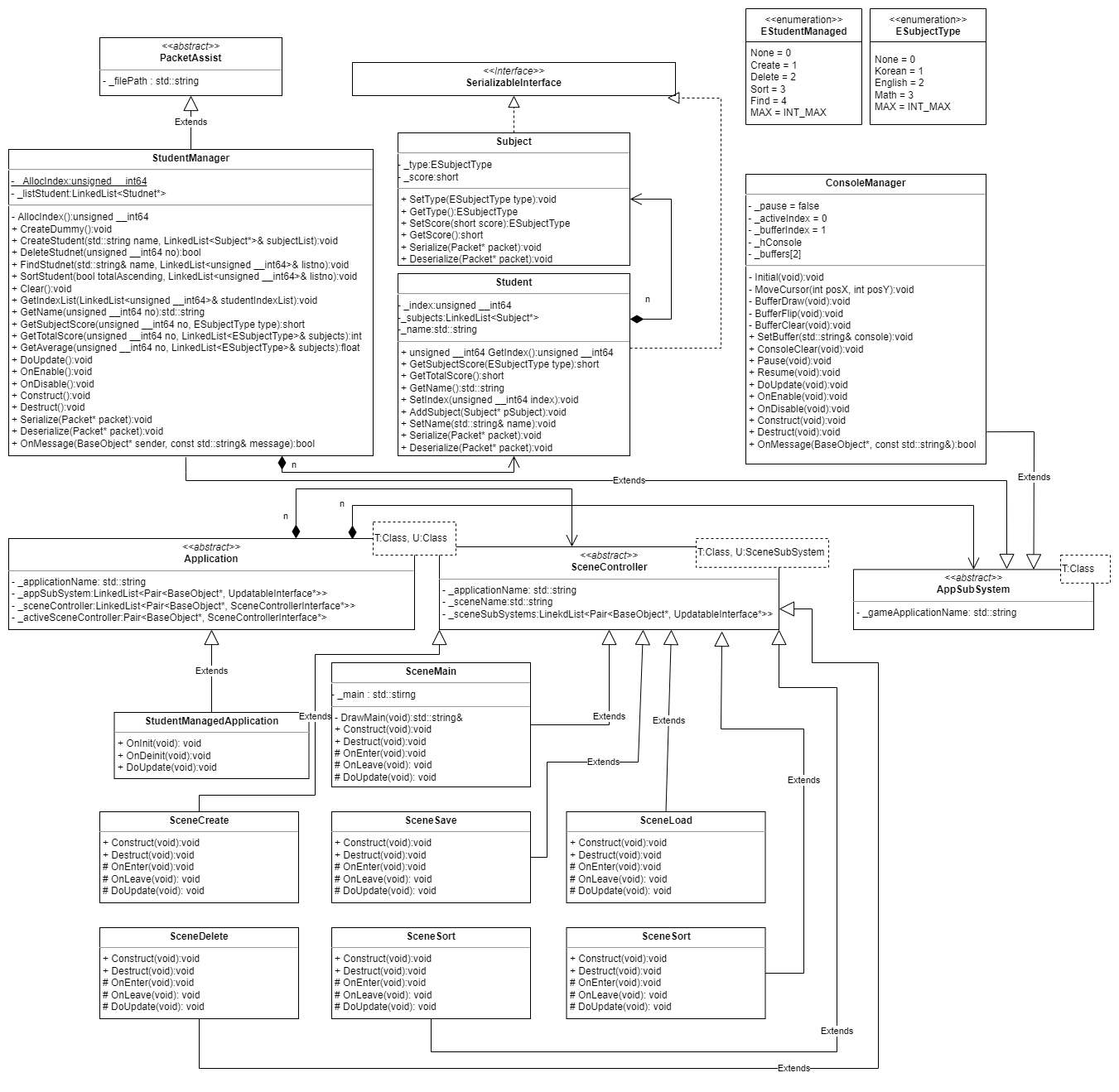
|  |  |  |
| --- | --- | --- |
| **부모클래스/인터페이스** | **클래스** | **내용** |
|  | **Packet** | 파일의 입출력을 일반화 한 클래스  C++의 값 타입에 따라 Write, Read를 오버로딩하여  Write, Read를 순차적으로 호출 시 파일의 입출력을 순서에  맞게 저장, 출력 해준다. |
|  | **PacketAssist** | 해당 클래스를 상속 받는 객체는 파일 입출력을 지원받을 수 있게 된다.  - Serialize(void): 저장하고 싶을 때 호출  - Deserialize(void): 로드하고 싶을 때 호출  - Serialize(Packet\*): 저장 해야 할 사항 정의  - Deserialize(Packet\*): 로드해야 할 사항 정의 |
|  | **SerializableInterface** | 직렬화 라이브러리의 지원 인터페이스 |

* 1. **자료구조 다이어그램**

[그림 1-3] 자료구조 다이어그램

1-3-1) 프레임워크 클래스 개요

|  |  |  |
| --- | --- | --- |
| **부모클래스/인터페이스** | **클래스** | **내용** |
|  | **Node** | LinkedList에서 사용하는 전이중 노드 |
|  | **Iterator** | LinekdList의 순회동작에 도움을 주는 순회자 클래스 |
|  | **LinkedList** | 템플릿 링크드리스트  일반적인 삽입, 삭제를 지원 |
|  | **Pair** | 템플릿 Pair 자료구조  데이터를 2개를 관리하고 싶을 때 이용 |

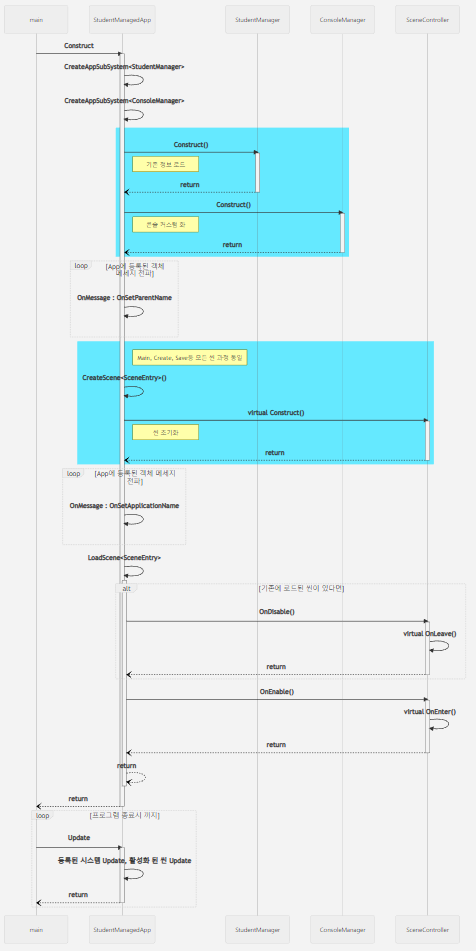
* 1. **학생관리 다이어그램**

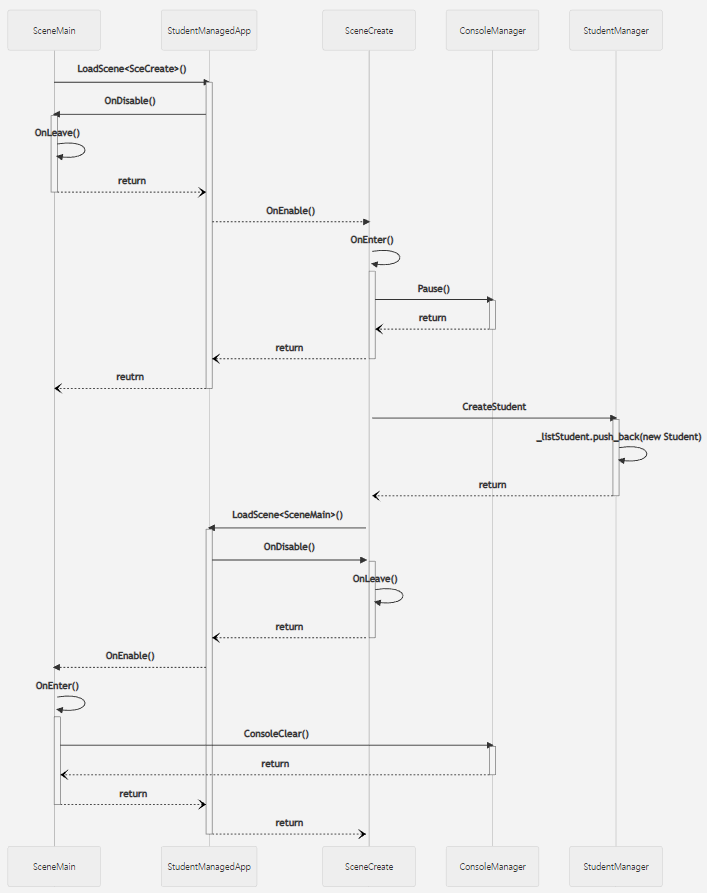
[그림 1-4] 학생관리 다이어그램

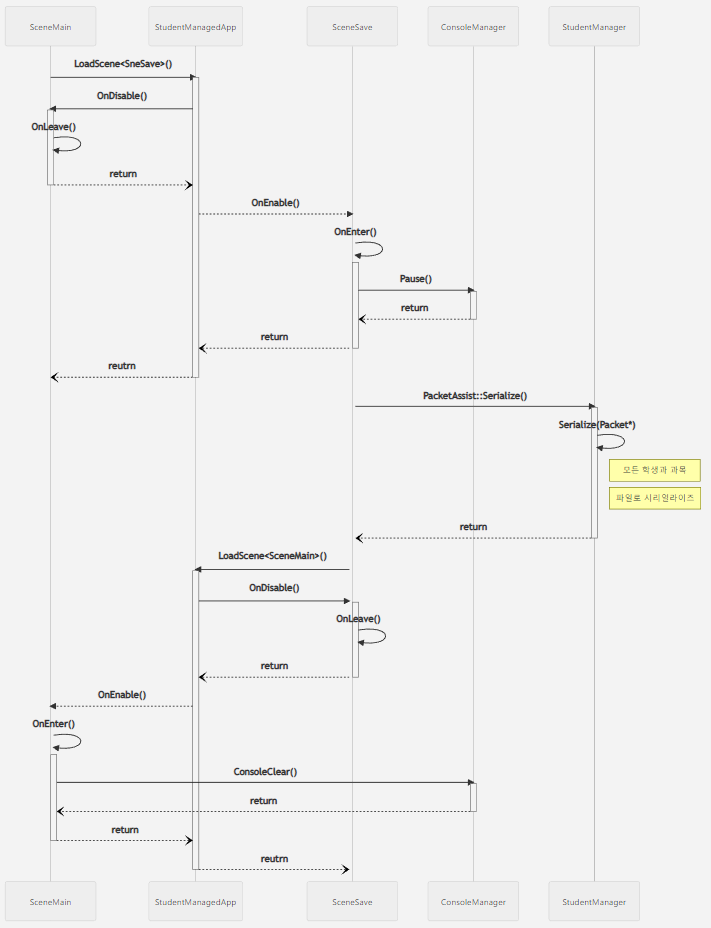
1-4-1) 학생관리 클래스 개요

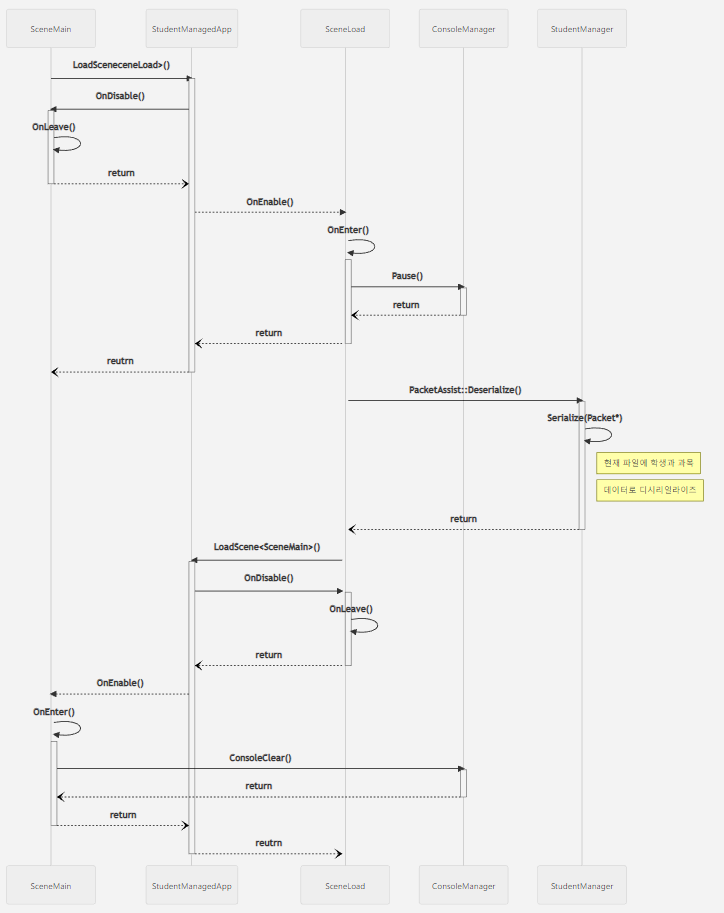
|  |  |  |
| --- | --- | --- |
| **부모클래스/인터페이스** | **클래스** | **내용** |
| **Application** | **StudentManagedApplication** | 학생관리 프로그램의 관리 객체 |
| **AppSubSystem, PacketAssist** | **StudentManager** | 학생을 관리하는 객체, 파일 입출력을 주도적으로 처리 |
| **AppSubSystem** | **ConsoleManager** | 콘솔 출력을 제어하기 위한 객체, 이중 버퍼링으로 구현 |
| **-/SerializableInterface** | **Student** | 학생 객체 |
| **-/SerializableInterface** | **Subject** | 과목 객체 |
| **SceneController** | **SceneEntry** | 첫 화면  아무키 클릭시 SceneMain으로 전환 |
| **SceneController** | **SceneMain** | 계속 적으로 학생목록을 출력  명령에 따라 씬전환을 함 |
| **SceneController** | **SceneCreate** | 학생 생성 씬 |
| **SceneController** | **SceneSave** | 파일 생성 씬 |
| **SceneController** | **SceneLoad** | 파일 로드 씬 |
| **SceneController** | **SceneDelete** | 학생 삭제 씬 |
| **SceneController** | **SceneSort** | 학생 정렬 목록 출력 씬 |
| **SceneController** | **SceneFind** | 학생 검색 목록 출력 씬 |

1. **시퀀스 다이어그램**

**2-1) 앱기동**

**2-2) 생성**

**2-3) 저장**

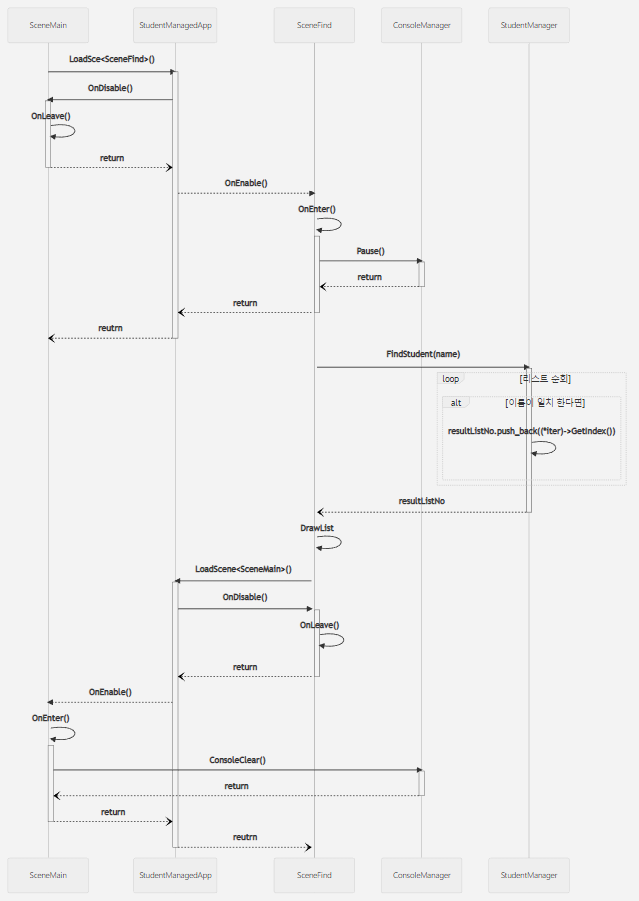
**2-4) 로드**

**텍스트, 하늘, 지도이(가) 표시된 사진

자동 생성된 설명2-5) 삭제**

**텍스트, 실내이(가) 표시된 사진

자동 생성된 설명2-6) 정렬**

**2-7) 검색**

**최종 결과 및 추가 내용**

1. **범용성/유연성/확장성/간결성 고려하여 추가된 내용**
   1. **현재 프로젝트 설계(구현 내용)**
2. 프레임워크 구현으로 프로그램의 확장과 유연함을 확보
3. 하나의 앱 객체에서 모든 객체를 관리가 되어 객체 간의 통신이 가능하게 구현
4. 앱 전체 혹은 씬 별로 계층적으로 객체를 관리하여 객체들의 활성화, 비활성화, 생성, 소멸을 사용자가 주도적으로 제어할 수 있게 구현
5. 파일 입출력을 라이브러리화 하여 다른 프로젝트에서도 사용할 수 있도록 제작
6. 학생관리 프로그램이 별도의 유니크한 키를 레코드별로 발급하여 값을 원자적으로 관리가 가능
   1. **향후 개발 내용**
7. 중복되는 씬 의 기능들이 많아 씬 을 더 추상화 하여 씬의 일반적인 기능별로 부모 클래스 구현
8. 현재 전체 데이터 쓰기, 읽기 구조로 되어 있어 레코드별로 파일에 offset을 저장하여 레코드 별로 값이 저장될 수 있게 구현
9. 특정 씬 의 Update에서 block함수 사용으로 Update가 멈추게 되어있음 이점 개선하여 InputManager와 ConsoleManager를 개선하여 실시간으로 사용자의 입력과 화면을 갱신할 수 있게 구현
10. 현재 각 싱글톤 객체에 접근하여 프로세스를 제어, 더욱 유저 편의적이게 프록시 객체를 생성하여 프록시 객체 만으로 애플리케이션을 제어할 수 있게 구현
11. **최종 결과**

텍스트이(가) 표시된 사진

자동 생성된 설명