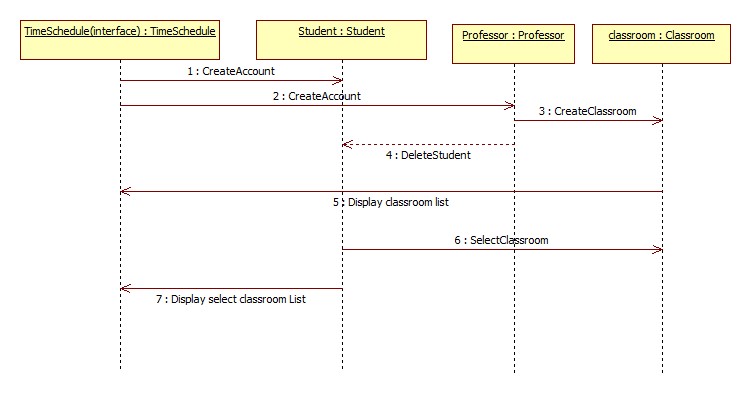
1. Introduction

저희 팀이 이번에 개발한 프로그램은 “The Simplest Timetable System” 입니다.

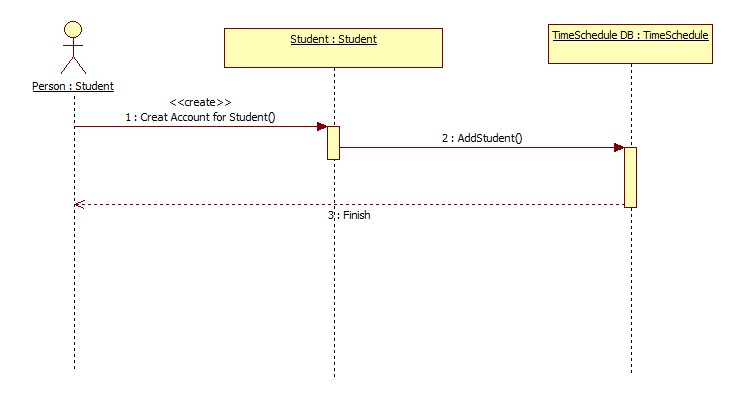
본교의 수강신청 프로그램을 지금까지 이용하면서 겪었던 불편 했던 점은 종합강의 시간표와 수강신청 기능이 독립적인 화면에서 수행 되는 것이었습니다. 그 때문에 시간표를 계획 한 후에 종이든 컴퓨터든 어딘가에 저장을 해두었다가 수강신청을 해야 할 때에는 저장해 두었던 것을 봐가면서 수강 신청을 해야 했습니다. 스피드가 생명인 수강신청에서 그러한 불편한 점을 해소 하고자, 최대한 간편하게 최소의 동작만으로 기능이 동작이 되는 최대한 편의를 위한 시스템 구축을 계획하게 되었습니다.

2. Whole System Flow

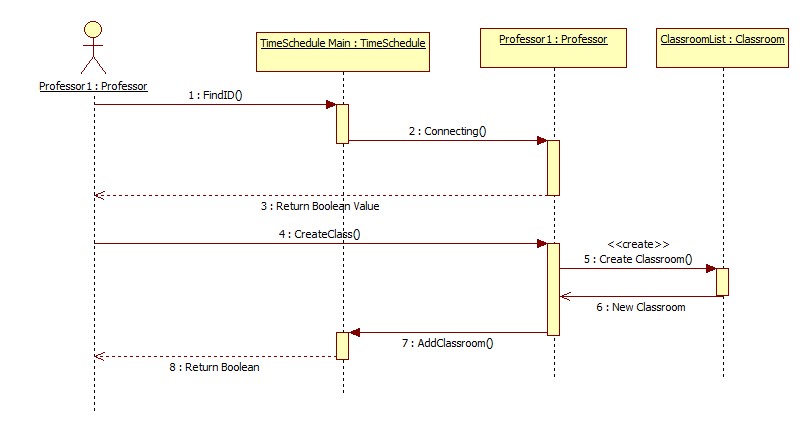


3. Sequence Diagrams

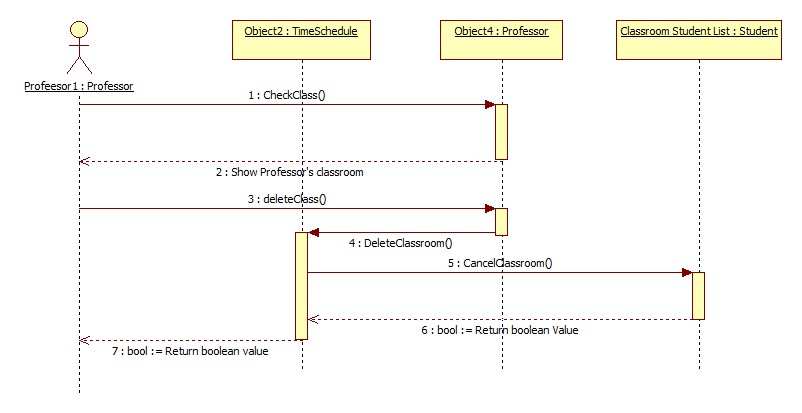
3-1.Create Account



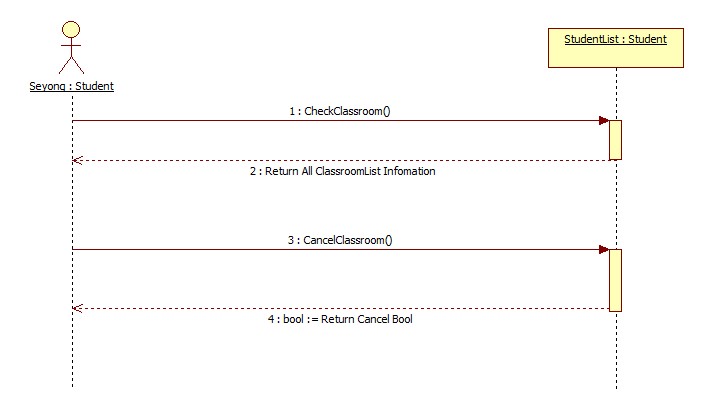
3-2.Create Classroom



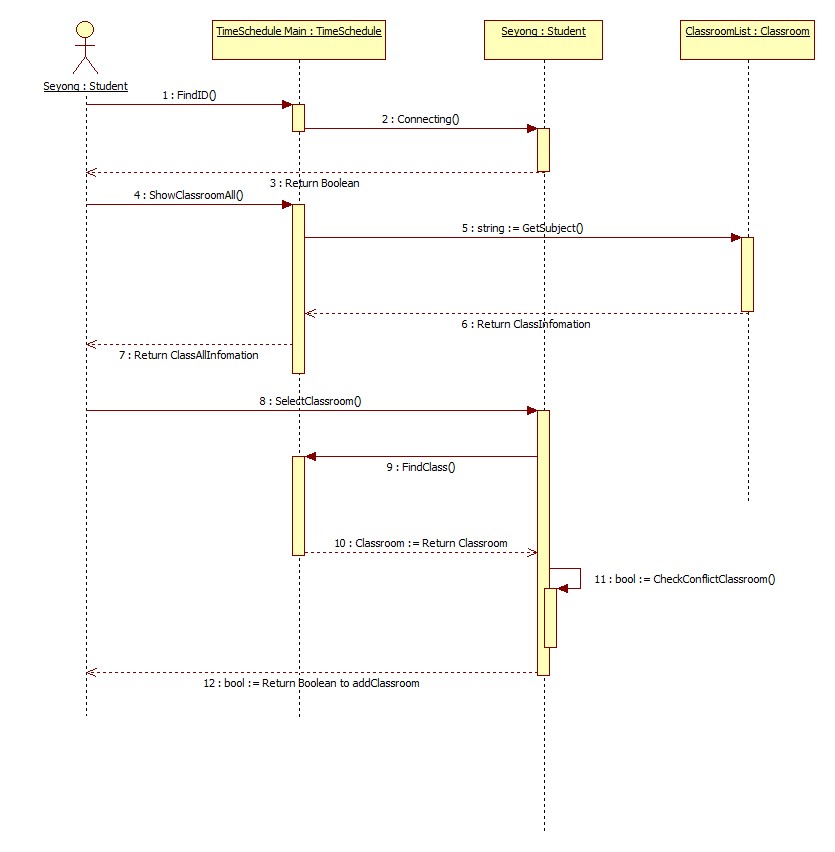
3-3.Delete Classroom



3-4.Cancle Classroom



3-5.Select Classroom



4. Change of Class Diagrams from

Design to Implementation









Professor 클래스에서의 변경 사항

CreateClass 메소드에서 받는 파라미터를 각각의 5개 변수에서

그것들을 묶은 클래스 형태로 받도록 하였습니다.



Classroom 클래스에서의 변경 사항

정수형의 멤버 변수 studentCount 를 추가 하였습니다.

이것은 해당 수업에 수강신청이 되어있는 학생들의 숫자를 나타냅니다.





TimeSchedule 클래스에서의 변경 사항

보다 심플한 구조를 위하여 AddStudent 와 AddProfessor 메소드를 AddPerson 메소드로 통합하였습니다.

교수 리스트를 출력 할 필요가 없기 때문에 ShowProfessorAll 메소드를 삭제 하였습니다.

각각의 클래스에서 초기화를 위하여 파라미터를 받는 생성자를 추가하였습니다.

Before Implementation



After Implementation



5. Conclusion

저희 팀의 “The Simplest Timetable System” 은

처음 시작 단계부터 simplicity 를 지향 했기 때문에, 프로그램의 규모를 작게 잡았습니다. 결과적으로 디자인 단계에서 구현 단계로 넘어가는 과정에서 시스템 동작의 큰 흐름은 거의 변경 사항이 없었고, 객체간의 관계, 데이터 흐름에서도 변경이 없었습니다.

다만, 소스코드 구조의 간결성을 위해 겹치는 메소드는 하나로 통합, 사용 되지 않은 메소드는 삭제 하였고, 여러 개의 파라미터를 주고 받던 메소드는 통합 된 하나의 클래스를 받도록 파라미터 부분을 바꾸었습니다.

이번 프로젝트를 진행하면서 느낀 점은, 분석과 설계 과정의 준비가 잘 되어 있을 수록 구현에 필요한 비용이 줄어든다는 것과, 개발 과정에서 어느 정도의 변경이 발생 하더라도, 수정이 용이 하다는 것을 직접 체험하며 느끼게 되었습니다.

그뿐만 아니라, SE 과정에서는 인적 자원 문제 등 예기치 못한 걸림돌이 자주 발생 한다는 것과 개발 과정에 상당한 타격이 된다는 것을 몸소 느끼게 되었습니다. 감사합니다.