『Perfect』 Time Table

Implementation Details

– 학사 관리 시스템 –

Team 8

200611458 김영승

200611478 성두훈

200611494 원스타

200611518 조민경

1. Introduction

이 문서는 프로젝트와 관련되어 선행되어 작성되었던 SRS와 SDD를 기반으로, 실제 구현된 TimeTable의 Implementation Detail에 대해 기술합니다.
"Perfect"한 타임테이블을 개발하기 위해 요구사항 분석과 디자인의 각 단계에서 도출된 오퍼레이션 항목들이 어떤 형태로 실제 구현에 적용되었는지 설명합니다.

2. Implementation Environment

Programming Language로는 객체 지향의 제약 사항과, GUI 어플리케이션이어야 한다는 제약 사항을 충족시키기 위하여 C++/CLI를 이용해 시스템을 구현하였습니다. C++/CLI는 Managed C++이라고도 하며, 객체 지향을 지원하는 언어로서 C++를 기반으로 .Net 위에서 확장된 언어입니다. 기존의 C++은 물론 C#, VB.Net 등 다른 .Net 언어들과도 호환성이 높아 GUI 어플리케이션을 개발하기에 알맞습니다.
GUI 기반으로 프로그램을 작성하기 위해서 Windows Form을 사용하였고, 개발 플랫폼으로는 Microsoft .NET Framework v3.5을 사용하였습니다.

2. UML Tool

UML 툴로는 StarUML을 사용했습니다.
StarUML은 오픈소스 기반의 국산 UML 툴로 각종 UML 다이어그램을 그릴 수 있으며, 그려진 다이어그램을 기반으로 C/C++/Java로 코드 제너레이션할 수 있는 우수한 툴입니다.
하지만 C++/CLI (Managed C++)로의 코드 제너레이션을 지원하지 않는다는 단점이 있습니다. 이를 극복하기 위해, C++로 코드 제너레이션 후 생성된 소스 코드를 Managed C++의 문법에 맞게 수정하는 작업이 필요했습니다. 이 작업으로 인하여 기존의 Class Diagram에 명시되었던 오퍼레이션 이름이나, 파라미터가 일부 변경된 사항이 존재하고, 이 부분에 대해선 뒷 부분에 좀 더 자세히 설명하도록 하겠습니다.

3. Operations

요구사항 분석 단계에서 도입되고 디자인 단계에서 구체화한 각각의 오퍼레이션을, Requirement - Design - Implementation 순서로 연계해 기술합니다.

Operation: 사용자의 타입과 신원을 구분
Requirements: SRS 3.1.1; 사용자별로 각기 다른 인터페이스를 제공해야 한다.
SRS 3.2.1; 프로그램 실행시 로그인 화면이 실행. 아이디, 패스워드 입력을 통해 로그인
Design: SDD 6.2.3 Sequence Diagram "로그인" 참조

Implementation: LoginForm::LoginProc()

Operation: 회원 가입 프로세스
Requirements: SRS 2.2.a: 사용자 별로 ID를 신청할 수 있다.
SRS 3.2.1.c 참조
Design: SDD 6.2.1 Sequence Diagram "가입신청" 참조
Implementation: class RequestIDForm 전체

Operation: 개인정보 변경 및 적용
Requirements: SRS 3.2.2 "개인정보" 참조
Design: SDD 6.2.4 Sequence Diagram "개인정보변경" 참조
Implementation: class PersonalInfoForm, class ProfessorPersonalInfoForm 전체

Operation: 학생 -> 강의 수강
Requirements: SRS 3.2.3.a "학생 -> 수업관리" 참조
Design: SDD 6.2.6 Sequence Diagram "수강신청" 참조
Implementation: class StudentAttendForm, class LectureListForm 전체

Operation: 수강한 강의 제거
Requirements: SRS 3.2.3.c "학생 -> 수업관리" 참조
Design: SDD 6.2.7 Sequence Diagram "수강제거" 참조
Implementation: class StudentAttendForm 전체

Operation: 학생 -> 시간표 보기
Requirements: SRS 3.1.4.b "학생" 참조
SRS 3.2.3.b "학생 -> 수업관리" 참조
Design: SDD 6.2.8 Sequence Diagram "시간표 보기" 참조
Implementation: class LectureTableForm 전체

Operation: 학생 -> 성적 확인
Requirements: SRS 3.2.4.a "학생 -> 성적 및 강의 평가" 참조
Design: SDD 6.2.10 Sequence Diagram "학점 보기" 참조
Implementation: class LectureGradeForm 전체

Operation: 교수 -> 담당 강의 관리
Requirements: SRS 3.1.5.b "교수" 참조
SRS 3.2.5.a~c "교수 -> 담당 강의관리" 참조

Design: SDD에 누락됨
Implementation: class ProfessorLectureListForm 전체

Operation: 교수 -> 성적 주기
Requirements: SRS 3.2.5.e~g "교수 -> 담당 강의관리" 참조
Design: SDD 6.2.9 Sequence Diagram "학점 주기" 참조
Implementation: class ProfessorGiveGradeForm 전체

Operation: 교수 -> 강의등록 신청
Requirements: SRS 3.2.6 "교수 -> 강의등록 신청" 참조
Design: SDD 6.2.5 Sequence Diagram "강의등록_추가" 참조
Implementation: class ProfessorAddLectureForm 전체

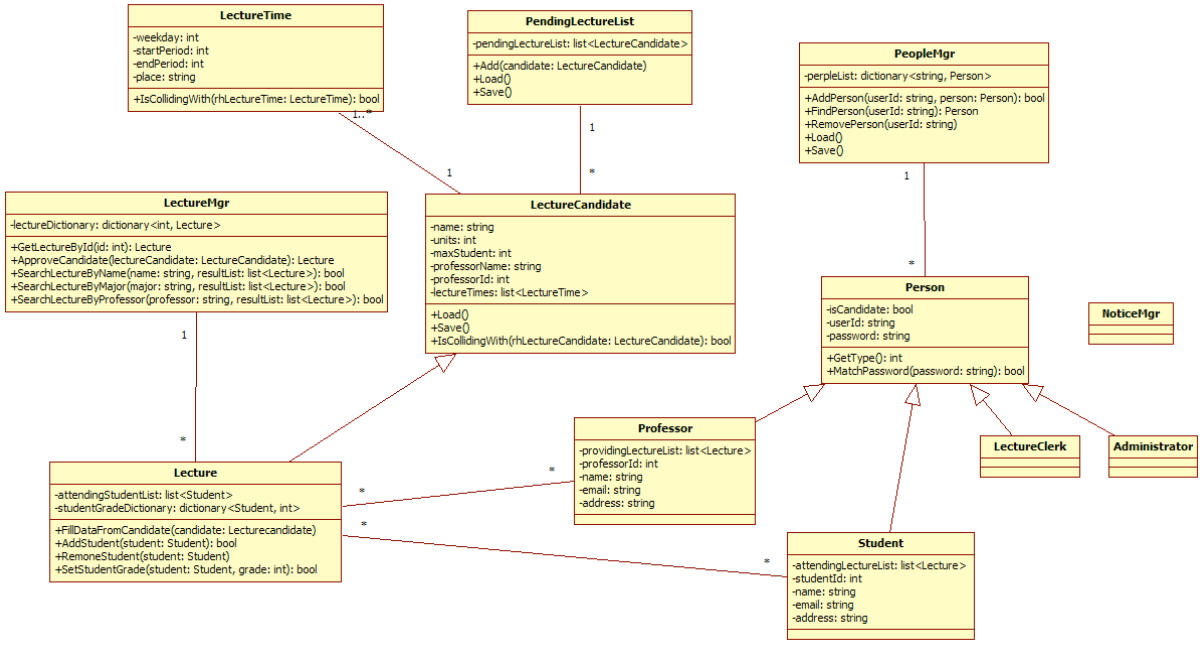
Operation: 수업관리자 -> 강의 조회 및 승인
Requirements: SRS 3.1.6.a~b "수업관리자" 참조
SRS 3.2.7.a~b "수업관리자" 참조
Design: SDD 6.2.5 Sequence Diagram "강의등록_추가" 참조
Implementation: class ApproveLectureForm 전체

Operation: 학사관리자 -> 인원 관리
Requirements: SRS 3.1.7.b "학사관리자" 참조
SRS 3.2.8 "학사관리자" 참조
Design: SDD 6.2.2 Sequence Diagram "가입승인" 참조
Implementation: class ManagePeopleForm 전체

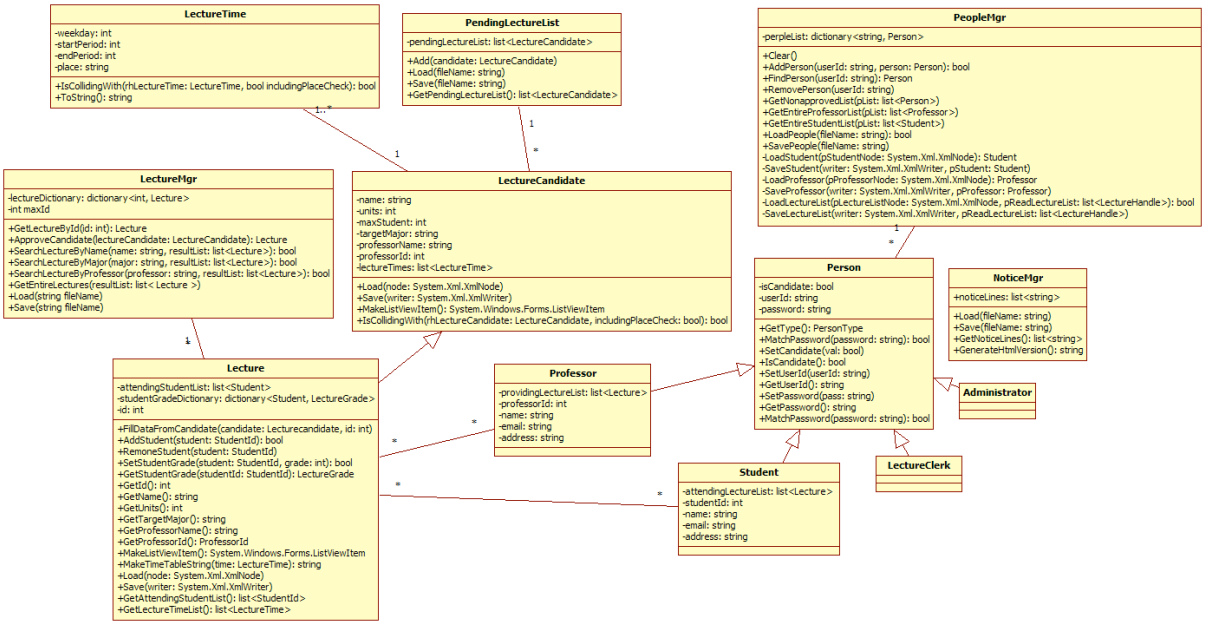
Operation: 공지사항 보기
Requirements: SRS 3.1.2 참조
Design: SDD 6.2.11 Sequence Diagram "공지보기" 참조
Implementation: class MainNoticeForm, class NoticeMgr 전체

Operation: 공지사항 수정
Requirements: SRS 3.2.9 "공지사항" 참조
Design: SDD 6.2.12 Sequence Diagram "공지추가" 참조
SDD 6.2.13 Sequence Diagram "공지삭제" 참조
Implementation: class NoticeManagementForm, class NoticeMgr 전체

4. 클래스 변경점



구현 전 클래스 다이어그램



구현 후 클래스 다이어그램

디자인 단계에서 도출된 클래스 다이어그램을 기반으로 GUI를 도입하여 구현 작업을 수행한 후 구현이 완료된 코드에 기반한 클래스 다이어그램을 비교하였습니다.

기존 디자인 단계에서 작성된 클래스 다이어그램의 구조에서 큰 변경은 없었습니다. 위 두 클래스 다이어그램을 보면 알 수 있듯이 각 클래스간의 구조적 변경은 없었고, 내부 오퍼레이션의 차이가 많다는 것을 알 수 있습니다. 이 오퍼레이션 부분은 대부분이 기존의 내용 변경이 아닌 새로운 오퍼레이션의 추가입니다. 기존의 클래스 다이어그램은 디자이너와 상의 하에 GUI 구현을 위하여 배제하였던 오퍼레이션들이 구체화된 것입니다. 변경된 사항에 대한 자세한 설명은 다음과 같습니다.

(1) UI 연동을 위해

UI는 디자인 단계에서 고려하지 않은 부분입니다. 이 내용에 관해서는 SDD 작성 단계에서 이미 디자이너와 상의를 통하여 배제하였던 부분이기 때문에, 디자인 단계에서는 각각의 요구사항에 의해 필요한 오퍼레이션에 대한 내용들만 클래스 다이어그램에 작성하였습니다. 따라서, 구현 단계에서는 UI 연동을 위한 다른 여러가지 함수가 추가되었습니다.

- Getter, Setter
    - 대표적으로 추가된 오퍼레이션 중의 공통부분으로 Getter와 Setter가 있습니다. 이 오퍼레이션은 클래스의 데이터를 가져오거나 설정하는 함수로, 디자인 단계에서 의도적으로 배제하였던 대표적 오퍼레이션입니다. 이 오퍼레이션은 UI를 구현함에 있어 구현 모듈에 의해 정해지기 때문에, 디자인 단계에서 UI 도구를 확정하지 않을 것이라면 지정하지 않도록 요구하였고, 그 요구에 의해 디자인 단계에서 배제되었던 사항입니다.

- LectureTime::ToString()
    - 이 오퍼레이션은 강의 시간에 대한 정보를 UI에 표시하기 위해 알맞는 문자열 형식으로 변환하는 메서드입니다. UI상에 강의 정보를 출력하기 위해 추가되었습니다.

- MakeListViewItem()
    - UI 구현에 있어 추가된 오퍼레이션으로, Windows Forms의 리스트 박스에 각 클래스의 데이터를 표시하기 위해 System.Windows.Forms.ListViewItem 클래스의 인스턴스를 생성하는 함수입니다. GUI 추가에 따라 구현 단계에서 추가되었습니다.

- MakeTimeTableString()
    - 시간표 보기 기능을 위해 추가된 오퍼레이션입니다. 강의에 대한 정보를 시간표 UI 표시에 맞는 형식으로 출력하기 위해 만들어졌습니다.

- NoticeMgr class
    - 해당 클래스의 내용은 기존 디자인 단계에서 전체적인 내용을 배제했던 부분입니다. 이 Notice 클래스는 공지사항에 관련된 부분으로, 이에 대한 데이터는 UI로 해결하기로 하였기

때문에 어떤 오퍼레이션도 존재하지 않았습니다. 이 클래스는 공지사항을 저장하고 있는 List와 xml 파일 간에 데이터를 저장하거나 불러오는 오퍼레이션으로 이루어져 있습니다.

(2) typedef의 사용

변경 전의 클래스 다이어그램에서는 사용자 타입, 학번, 강의 등을 구분하는 키로 int형 변수를 사용했으나, int라는 추상적인 자료형을 그대로 사용하기보다는 더욱 명확한 이름으로 자료형을 구분하는 것이 좋다고 생각해 아래와 같이 세부적으로 재정의하였습니다. 이 재정의는 구현상 구분의 편의성을 위해 정의한 것으로, 전체적인 구조에는 영향을 미치지 않습니다.

- int -> PersonType
- int -> ProfessorId
- int -> StudentId
- int -> LectureHandle

(3) 안정성 고려

변경 전에는 수강 강의 리스트 등에서 실제 객체의 포인터를 가지고 있었지만, 이 경우 객체를 삭제하는 등 무효화된 포인터로 인해 크래시가 발생할 가능성이 큽니다. 그래서 포인터 대신 딕셔너리에 대한 키인 핸들을 사용하도록 변경하였습니다.

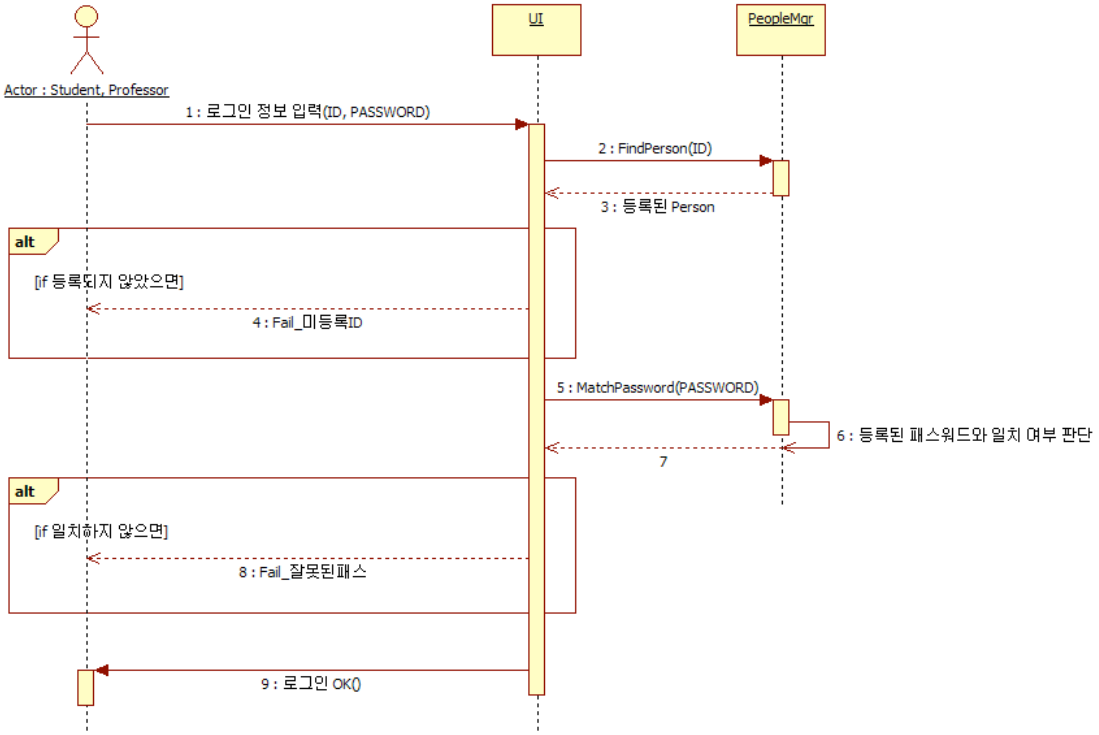
- Lecture
  attendingStudentList: List<Student> -> attendingStudentList: List<StudentId>
- Professor
  providingLectureList: List<Lecture> -> providingLectureList: List<LectureHandle>
- Student
  attendingLectureList : List<Lecture> -> attendingLectureList : List<LectureHandle>

5. Sequence -> Code

전체 코드는 크기 때문에 일부 다이어그램만 기술합니다.

(1) 로그인 (SDD 6.2.3)

Sequence Diagram



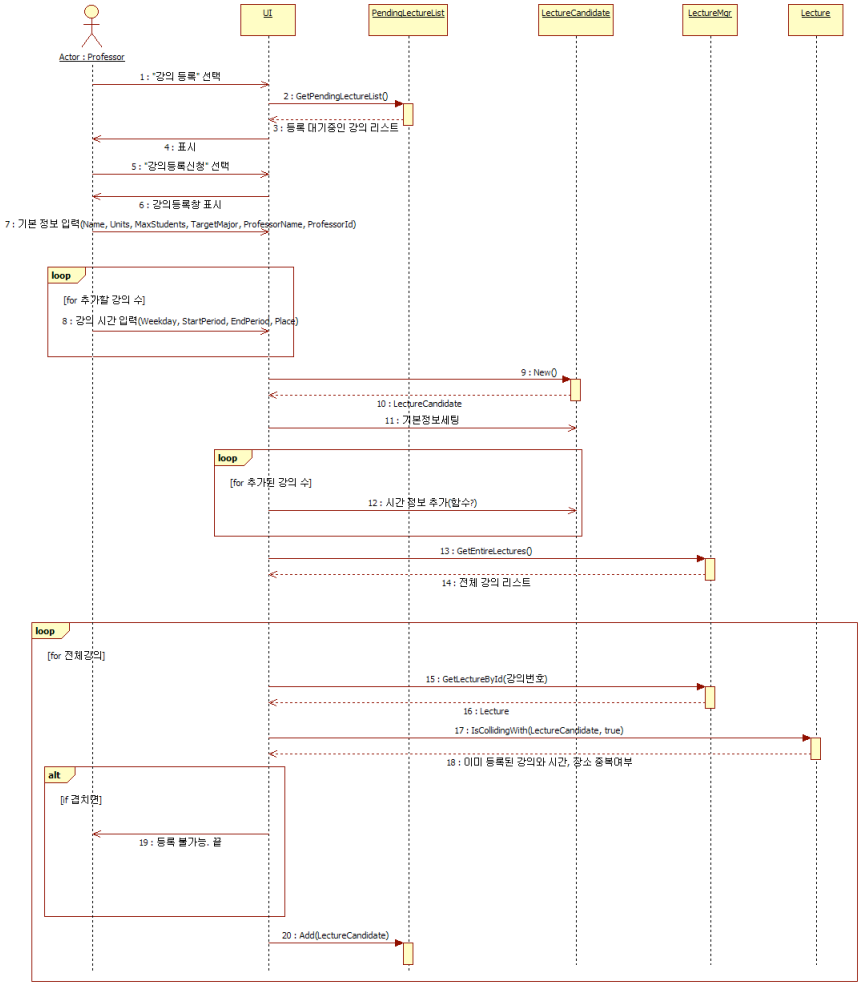
Source Code

TimeTable::LoginForm — LoginProc(System::String ^ userId, System::String ^ password)

```
82          //
83      }
84
85      LoginForm::LoginResult LoginForm::LoginProc(String^ userId, String^ password)
86      {
87          Person^ pFoundPerson = m_pPeopleMgr->FindPerson(userId);
88          if (!pFoundPerson)
89          {
90              return LoginResult::kLoginFail_UnknownId;
91          }
92
93          if (!pFoundPerson->MatchPassword(password))
94          {
95              return LoginResult::kLoginFail_IncorrectPassword;
96          }
97
98          m_pLoggedPerson = pFoundPerson;
99          // 로그인 처리
100         return LoginResult::kLoginOk;
101     }
```

(2) 교수 -> 강의등록 신청 (SDD 6.2.5)

Sequence Diagram

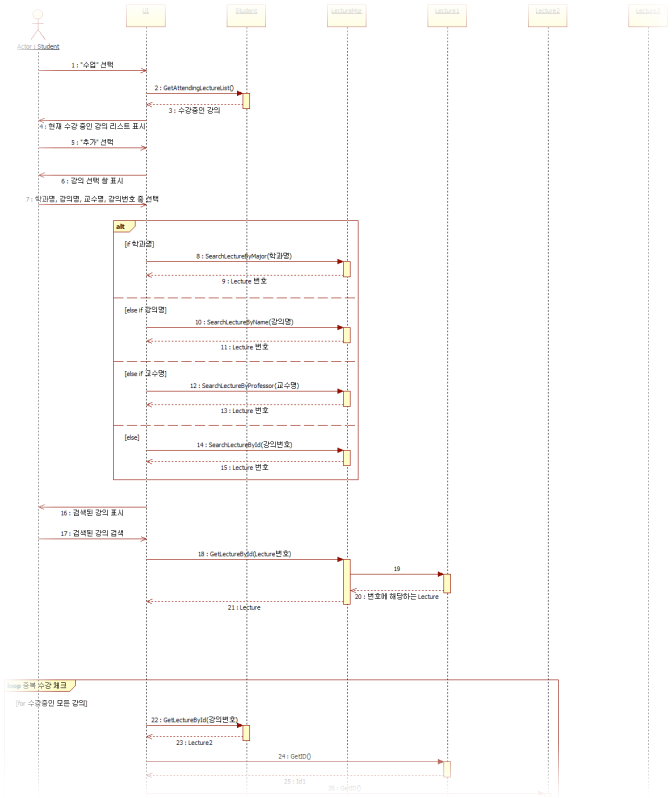


Source Code (일부)

```
System::Void ProfessorAddLectureForm::requestButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (m_timeList.Count == 0)
    {
        MessageBox::Show("강의 시간을 정해 주세요.");
        return;
    }

    LectureCandidate^ candidate = gcnew LectureCandidate;
    candidate->m_name = nameTextBox->Text;
    candidate->m_units = Convert::ToInt32(unitsNumericUpDown->Value);
    candidate->m_maxStudents = Convert::ToInt32(limitsNumericUpDown->Value);
    candidate->m_targetMajor = majorTextBox->Text;
    candidate->m_professorName = m_pProfessor->GetName();
    candidate->m_professorId = m_pProfessor->GetProfessorId();

    candidate->m_lectureTimes.Clear();

    for each (LectureTime^ lectureTime in m_timeList)
    {
        candidate->m_lectureTimes.Add(lectureTime);
    }

    try
    {
        // 이미 등록돼 있는 강의와 검사
        Generic::List< Lecture^ > lectureList;
        m_pLectureMgr->GetEntireLectures(%lectureList);
```

(3) 학생 -> 수강 신청 (SDD 6.2.6)

Sequence Diagram (일부)

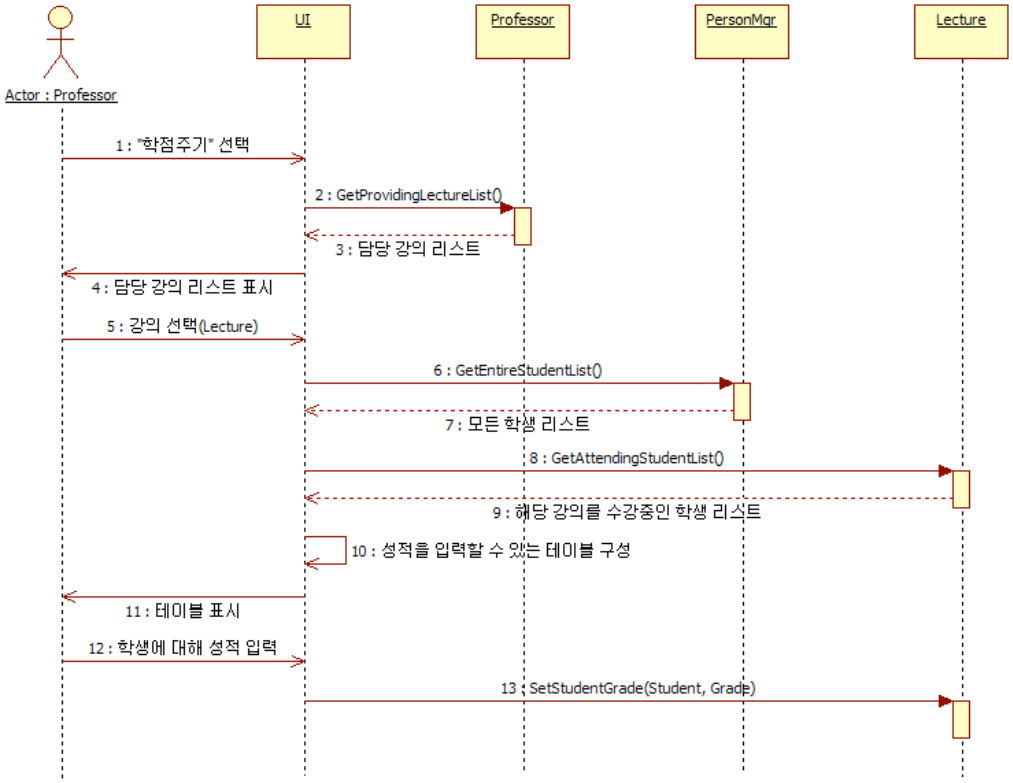


Source Code (일부)

```
meTable::StudentAttendForm                          addButton_Click(System::Object ^ sender, System::EventArgs ^ e)
35      System::Void StudentAttendForm::addButton_Click(System::Object^ sender, System::EventArgs^ e)
36      {
37          LectureListForm form(m_pLectureMgr);
38
39          form.ShowDialog(this);
40
41          if (!form.GetSelectedLecture())
42              return;
43
44          Lecture^ pSelectedLecture = m_pLectureMgr->GetLectureById(form.GetSelectedLecture());
45          if (pSelectedLecture)
46          {
47              // 중복 체크
48              for each (LectureHandle lectureHandle in m_pAttendingLectureList)
49              {
50                  Lecture^ pLecture = m_pLectureMgr->GetLectureById(lectureHandle);
51                  if (!pLecture)
52                      continue;
53
54                  if (pSelectedLecture->GetId() == pLecture->GetId() || pSelectedLecture->GetName()->ToLower()->Equa
55                  {
56                      MessageBox::Show("과목은 중복 수강할 수 없습니다.");
57                      return;
58                  }
59              }
60
61              // 선행 과목 체크
62
```

(4) 교수 -> 학점 주기 (SDD 6.2.9)

Sequence Diagram

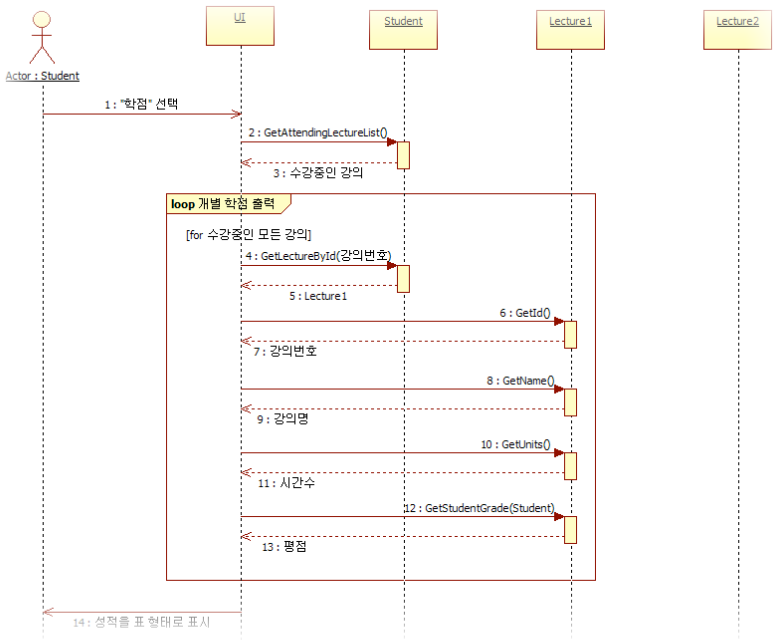


Source Code (일부)

```
94
95      System::Void ProfessorGiveGradeForm::listView1_SelectedIndexChanged(System::Object^ sender, System:
96      {
97          if (this->listView1->SelectedIndices->Count == 1)
98          {
99              LectureHandle lectureHandle = m_pProfessor->GetProvidingLectureList()[this->listView1->Sele
100             m_pSelectedLecture = m_pLectureMgr->GetLectureById(lectureHandle);
101         }
102
103         SyncStudentList();
104     }
105
106     System::Void ProfessorGiveGradeForm::dataGridView1_CellEndEdit(System::Object^ sender, System::Win
107     {
108         if (e->ColumnIndex == 2 && m_pSelectedLecture)
109         {
110             Generic::List< StudentId >^ attendingStudentList = m_pSelectedLecture->GetAttendingStudent
111             StudentId student = attendingStudentList[e->RowIndex];
112             System::Object^ valueObj = this->dataGridView1->Rows[e->RowIndex]->Cells[2]->Value;
113             int value = StringToGrade(safe_cast< System::String^ >(valueObj));
114
115             m_pSelectedLecture->SetStudentGrade(student, (LectureGrade)value);
116         }
```

(5) 학생 -> 성적 확인 (SDD 6.2.10)

Sequence Diagram (일부)



Source Code (일부)

```
eTable::StudentGradeForm                                    UpdateSummary()
    }
}

void TimeTable::StudentGradeForm::UpdateSummary()
{
    int totalUnits = 0;
    int approvedUnits = 0;
    double totalScore = 0.0;
    double avgScore = 0.0;

    for each (LectureHandle lectureHandle in m_pStudent->GetAttendingLectureList())
    {
        Lecture^ pLecture = m_pLectureMgr->GetLectureById(lectureHandle);
        int units = pLecture->GetUnits();
        LectureGrade grade = pLecture->GetStudentGrade(m_pStudent->GetStudentId());
        double score = GradeToScore((int)grade);
        totalUnits += units;
        if (grade != LectureGrade::Unidentified)    // 성적을 받은 경우만...
        {
            approvedUnits += units;
            totalScore += score * (double)units;
        }
    }

    avgScore = totalScore / (double)approvedUnits;

    this->summaryLabel->Text = String::Format("신청 학점: {0}        취득 학점: {1}
}
```