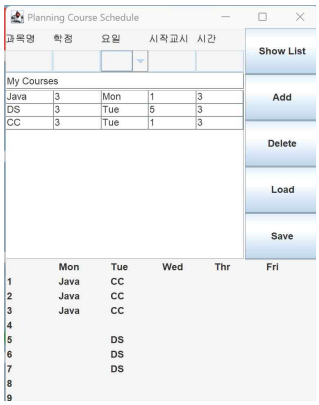
2025 소프트웨어프로젝트 프로젝트 3

# 강의 시간표 프로그램 (Course Schedule)

1. 내용

학기 초에 수강 신청을 하고 그 결과를 주간 강의 시간표로 작성하는 프로그램을 작성하는 것이 이번 학기 프로젝트의 주제이다. (강의에 관해서는 강의 시간만을 다루며, 강의실 등 다른 강의 속성은 고려하지 않는다.) 프로젝트 3에서는 수강 신청을 한 과목들이 파일에 기록되어 있다고 가정하고, 1) 파일의 내용을 입력해서 이를 해석하고, 2) 수강한 강의들의 정보를 내부 데이터로 관리하며, 3) 이를 콘솔에 시간표 형태로 출력하는 프로그램을 작성한다.

```
    Mon   Tue   Wed   Thr   Fri
---------------------------------
1   Java    CC
2   Java    CC
3   Java    CC
4
5           DS
6           DS
7           DS
8
9
```




향후, 과제 5, 6에서, 강의 목록에서 과목을 선택하여 ADD하고, 신청한 강의 리스트에서 강의를 DELETE하며, 강의 리스트를 파일에 LOAD/SAVE하는 기능을 제공하는 GUI 버전을 만들게 된다.

따라서, **단순히 입출력으로 강의 정보를 보여주는 프로그램이 아니고**, 강의 정보를 입력, **메모리에서 체계적으로 관리**, 출력하는 프로그램을 작성하는 것임. 향후 강의 정보의 추가/수정/삭제 기능의 추가에 대비해서 **객체 지향 개념으로 작성**하여야 한다. 따라서 다음 요구 사항에 맞게 프로그램을 작성하여야 한다.

**주의)**

**1. 이번 학기를 좌우하는 과제.** 이 과제를 대충해서 넘어가면, 이어지는 내용은 제대로 소화할 수 없음. 결국, 시간과 노력만 낭비하고 Java는 어렵다는 인식만 남게 됨. **제대로 하는 것이 매우 중요.**

**2. 중간시험과도 직결되는 내용.** (과제를 스스로 했는지 재확인하는 문제 출제)

**3. 문제가 복잡하기 때문에, 설명도 복잡할 수밖에 없음.** 복잡한 실제 생활 문제를 해결하는 것이 컴퓨터공학의 본질. 문제 설명이 복잡하다고 포기하고 거부하면, 팀장이 짜라는 대로 짜는 저급 프로그래머가 될 수밖에 없다. **이 설명서를 읽고 또 읽어서 스스로 문제를 이해하여야 한다.**

**4. 제대로 프로그램을 작성하려면, 고려해야 할 사항이 많기 때문에, 일찍 시작하는 것이 중요.**

1) Course class

- 강의 시간을 중심으로 기본 강의 정보를 하나의 객체로 모아서 나타냄.
- 다음 정보들은 반드시 포함하여야 한다.
  과목명, 학점, 강의 요일, 강의 시작 시간, 강의 시간
  (강의는 일주일에 한번 연속 강의로만 이루어진다고 가정)
- 정보들을 다루는 각종 메소드들 포함 (향후, 계속 추가되어야 함)
- 하나의 강의에 대한 정보를 콘솔로 출력하여 주는 public void print() 메소드는 **필수**
- 세부 정보들을 서술한 스트링 라인에서 객체를 생성하는 Course(String infoLine)
- 또는, 분리된 세부 정보들에서 객체를 생성하는 Course(String[] infoFields)

2) CourseSchedule class
- 수강한 강의 객체들의 묶음을 나타냄.
- 즉, Course 인스턴스들을 모아서 관리하는 클래스.
- Excel로 수강 강의를 관리한다고 가정한다면, Course는 각 행, CourseSchedule는 테이블 전체.
- 자료구조는 array 사용 (향후, ArrayList로 갱신)
- 리스트의 i 번째 강의 정보를 알려주는 public Course getCourse(int i) 메소드는 **필수**
- 인자로 파일 이름을 주면 해당 파일에 기술된 강의 정보들로 리스트를 만드는 생성자, CourseSchedule(String infoFileName) **필수**
- 모든 강의 정보들을 주간 시간표 형태로 출력하는 public void print() **필수**
- 파일에서 강의 정보 내용을 한 줄씩 읽고 파싱해서, Course와 CourseSchedule 객체를 만들고,
- 반대로, CourseSchedule 객체를 파일에 저장하는 메소드 구현 추천
- Course 객체를 리스트에 add/delete하는 메소드들은 향후, ArrayList를 배우고 난 뒤에, 추가
- 그 외의 내부 구현 사항은 보여서는 안 됨.
- main()을 포함하면 안 됨.

3) 강의, 즉, Course 정보를 기술하는 파일은 다음 양식을 따른다.
- 하나의 줄(line)이 하나의 Course를 기술.
- ":"으로 분리된 5개 항목으로 구성
  * 과목명 (예, Data Struture, DS)
  * 학점 수 (예, 3)
  * 강의 요일 (예, Mon)
  * 강의 시작 시간 (예, 5 (교시))
  * 강의 시간 (예, 3 (시간))
- 예) DS:3:Tue:5:3
- 앞뒤의 공백문자는 무시
- 빈줄, 또는 처음을 //로 시작하면 comment 처리
- 정보 사항은 일단 모두 영어로. (한글 처리가 문제가 될 수도 있음)
- 강의 정보 파일의 예로 "myschedule-normal.data" 배포.
  사용하고 있는 개발 도구에서, 프로젝트에 copy&paste로 추가
- 수강 신청한 강의 수는 10개 이하로 가정한다.

4) 잘못된 파일명, 잘못된 규칙으로 기술된 파일 내용 등 오류에 대해 콘솔에 보고하여야 함.
  - 즉, 잘못된 입력 경우에 대처.
  - 비정상 입력 발견 뒤, 나머지 입력을 계속할지 여부는 개인이 결정. 즉, 평가 대상이 아님.

5) main()을 포함하는 별도의 Test 클래스를 만들어서 테스트. (뒤의 "필수 테스트" 참조)

2. 목적
- Java String 구조 사용법 연습
- File I/O 사용 연습
- 인터페이스 정의를 통한 프로그램 배분/검사 등 분업화된 프로그래밍 맛보기

3. 추진 방법

0) **제대로 하려면, 일찍 시작하여야 한다.**
  - 마감에 쫓기면 자기 프로그램을 뒤돌아볼 수 없다.
  - 먼저 하는 것이 절대로 손해 보는 것이 아니다. 자신이 직접 겪은 실행 착오는 '약'

1) 문제 정의에 입각해서 프로그램 구조 설계. 즉, 사용할 클래스들 정의.
  - Course, CourseSchedule

2) 강의 정보 기술방법 (“위 1-3)”)이해
3) String이 제공하는 method들 공부

--- 정상 입력 파일에 대한 동작 완성 ----------
4) 1단계로 파일에서 가장 간단한 형태 한 줄을 읽어서 정해진 필드를 분리하는 파싱 메소드를 작성하고, 그 결과에 해당하는 **하나의 강의 정보**(class Course) **인스턴스/객체를** 만드는 작업을 완성.
5) print()를 코딩하여 제대로 파싱을 하는지 확인
6) 여러 줄의 파일 입력, 즉, 여러 강의 정보들의 처리 코딩.
   - 즉, CourseSchedule 코딩
   - 필수 메소드 구현 및 테스트

--- 비정상 입력 파일에 대응하도록 확장 ----------
7) 강의 시간이 겹치는지 체크하고, 리스트에 추가
8) 각종 정상/비정상 입력에 대해 동작 점검 및 프로그램 수정
   - 비정상 입력 발견 뒤, 나머지 입력을 계속할지 여부는 개인이 결정. 즉, 평가 대상이 아님.
9) comment 라인을 처리하는 문장을 추가해서 파일 읽기 메소드를 완성

10) **구조 > 기능 > 성능**
   구조를 맞추지 않고 스트링 처리만 하면, 이어지는 프로젝트에 적용할 수가 없음.
   제대로 구조를 완성하는 것이 무엇보다 중요하며, 이를 위해서는 문제의 이해가 선행되어야 함.
11) **스스로 충분히 디버깅을 수행한 후, “필수 검사”를 통해 최종 점검**
12) 리포트 작성

4. **“필수 검사”**를 이용한 구현 검증
   1) **주의:** 필수 검사는 보고서 제출/평가를 위한 **종합 테스트**
      - 프로그래밍 과정 중에서의 디버깅을 필수 검사로 하면, 진도를 나갈 수 없음.
      - 또한, 필수 검사는 프로그램 동작 점검의 일부분. 즉, 필수 검사를 통과했다고 해서 프로그램 개발이 완성되었다고 할 수 없음.
      - 자체적으로 디버깅을 충분히 하고 난 후, 평가 포인트들을 점검하는 목적으로 필수검사 사용
      - 필수 검사의 테스트 소스 프로그램을 공개하면, 필수 검사를 왜곡해서 사용 가능
      - 이상의 이유로 필수 검사 프로그램의 소스는 제공하지 않음
   2) 프로그램이 과제의 요구 사항을 만족하는지를, 모든 학생에게 **동일하고 규격화해서** 검사
      - 인터페이스 요구 사항, 즉, 앞의 1-1), 1-2) 규정을 준수
      - 기능적 요구 사항 만족. 즉, 앞의 1-3) 준수
   3) 검사 방법 (배포 자료 참조)
      a. (사용하고 있는 개발 도구에서) 필수 검사 용도의 **별도** 프로젝트 생성
      b. 작성한 소스 프로그램 추가 (주의: 자체 테스트용 main()이 포함된 class 소스는 제외)
      c. 배포된 필수검사 Kit, “required-test.zip”의 내용을 프로젝트에 추가
         c-1. data 파일들은 (*.data) 테스트 **프로젝트에 추가**
         c-2. bin 폴더 아래 있는 “Test.class”는 개발 도구의 해당 프로젝트 workspace 내에 있는 class 파일들 공간에 추가 (Eclipse 경우, “workspace/project/bin”)
      d. 개발 도구의 실행 구성 (Run Config)에서 Main class를 “Test”로 지정하고, Run

      g. 먼저, 정상 입력 case에 집중해서 필수 검사를 수행하여 완성한다.
      h. 비정상 모든 항목의 결과가 “Correct Answer”와 **유사하게** 나오면, 구현이 올바르게 된 것이므로 구현을 마치고, 그렇지 않은 경우 구현을 수정하여 다시 필수 검사를 수행한다.

5. 평가항목

- 프로그램의 동작 여부 (70%)
  * 동작 여부를 확연히 알아볼 수 있는 결과. 즉, “필수 검사” 결과 화면
  * 본인이 작성한 프로그램의 진위 여부를 판단하는데 도움 되는 인증 자료 포함
- 리포트 적정성 (30%)
  * 설계 및 구현 시 고민했던 사항과 결정 내용을 간략히 기술
    + 본인이 직접 프로그램을 했음을 자연스럽게 입증
  * 평가자 입장에서, 평가사항을 찾기 쉽게 작성하였는가?

6. 리포트 제출

1) 기한 : 04/16(수) 까지
- 최종 테스트 과정 화면, 설계 노트, 소스 프로그램, 자체평가표
- 하나로 묶은 리포트를 eClass로 제출

<참고 1> file 입력 및 try/catch 예

File file = new File(“myschedule-normal.data”);

```
try {
        input = new Scanner(file);  // file is an instance of File
}
catch(Exception e) {
        // print error message and return;
}

while (input.hasNext()) {
        String line = input.nextLine();
        // 강의 정보 파싱
}
```