객체지향 프로그래밍 1차 프로젝트

과제번호: 01

실습반: 318호

실습실명/실습조교명: 남경진 조교

이름: 안창희

학번: 201723272

학과: 소프트웨어학과

학년: 3학년

– 서론 (본 과제에 대하여 전체적으로 소개한다. 과제가 무엇인지, 어떻게 해결하였는지 접근방법, 구현하지 못한 부분 등을 설명한다.)

본 과제의 요구사항은 입출력 요구에 맞는 좌석 예약 프로그램을 구현하는 것 이다. 프로그램의 기능에 대한 요구사항은 4가지가 있으며 다음과 같다.

=〉 1.좌석 예약 / 2.예약 취소 / 3.전체 예약 확인 / 4.프로그램 종료

과제를 이해하면서 가장 처음으로 해야 할 일은 클래스를 구상하는 일이었다. 과제의 요구사항에서 최소 3개 이상의 클래스를 구현하라고 했고, 강의에서 배운 내용을 토대로 각 클래스는 프로그램을 기능별로 나눈 분류가 아니라 데이터 분류로 나눈 데이터 중심으로 설계되어야 하기 때문에 요구사항 중 어떤 부분이 핵심 도메인이 되고 분류가 어떻게 되어야 할지 정립할 필요가 있었다.

내가 이해한 요구사항의 핵심은 프로그램의 각 기능이 모두 “예약”이라는 데이터를 중심으로 실행되는 기능이란 것이었다.

따라서 프로그램의 기능을 다음과 같이 내가 구현할 방식대로 이해하기로 했다.

1. “예약” 객체의 생성
2. 지정된 “예약”객체를 삭제처리
3. 전체 “예약” 객체 정보 일람
4. 프로그램 종료

하지만 큰 기능을 정립한 후에도 입출력 요구에 맞는 세부사항을 확실하게 파악하고 넘어갈 필요가 있었다.

가장 큰 의문은 프로그램의 사용자가 1일 이상의 예약을 가능할 수 있도록 구현해야 하는가? 이었다.
이는 조교님에게 질문을 통해 해결할 수 있었고,
사용자는 하루 단위의 예약, 즉 시작시간과 종료시간의 Day가 반드시 동일하게 설정해야 하는 것으로 방향을 정했다.

그리고 이 부분의 구현은 예약 당시 시작시간과 종료시간의 년, 월, 일이 동일하지 않으면 에러를 검출하는 기능을 추가하는 것으로 결론지었다.

이후 각 클래스에 대한 명확한 정립을 하였다.
(클래스에 대한 자세한 설명은 본문에서 다루도록 한다.)

클래스들을 정립한 후 각 기능이 어느 정도 동작하는 것을 확인하였을 때에는, 과제에서 요구하는 겹치는 시간의 예약이 있었을 때 오류를 출력하고 메뉴로 돌아가는 기능을 추가했으며,

이는 시작시간과 종료시간을 입력받을 당시에 기존에 존재하는 모든 예약과 비교대조하여 시작시간이 다른 예약의 구간에 있으면,
"시작시간이 다른 예약구간의 중간에 있습니다!! 메뉴로 이동합니다." 라는 문장을 출력하도록, 하였다. (종료시간 또한 같은 기능을 지닌다.)

클래스와 각 기능에 대한 세부 설명은 본문에서 다루기로 하며,
문제를 발견했으나 해결하지 못했거나, 설계는 할 수 있었지만 구현하지 못한 기능은 다음과 같았다.

1. 입력 숫자의 종류와 범위는 오류 검출 기능을 통해 제한할 수 있었지만, 입력 숫자가 여러 개가 들어올 경우 프로그램이 오작동 했다.

예를 들어 좌석 객체의 생성을 위해 받는 입력은 반드시 행과 열에 해당하는 두 개 숫자만 입력받아야 하지만, 세 개 이상의 숫자를 입력해버릴 경우 프로그램이 오작동 하는 것을 피하지 못했다.

이를 해결하기 위해선 Scanner 클래스의 nextint() 메소드의 작동을 관제하여 2번 이상의 띄어쓰기가 있을 경우 입력을 제한하고 다음으로 넘어가는 등의 기능을 넣어야 하나 구현하지 못했다.

2. 예약 객체의 자료를 관리하는 과정에서 최적화된 자료구조를 구현하지 못했다.

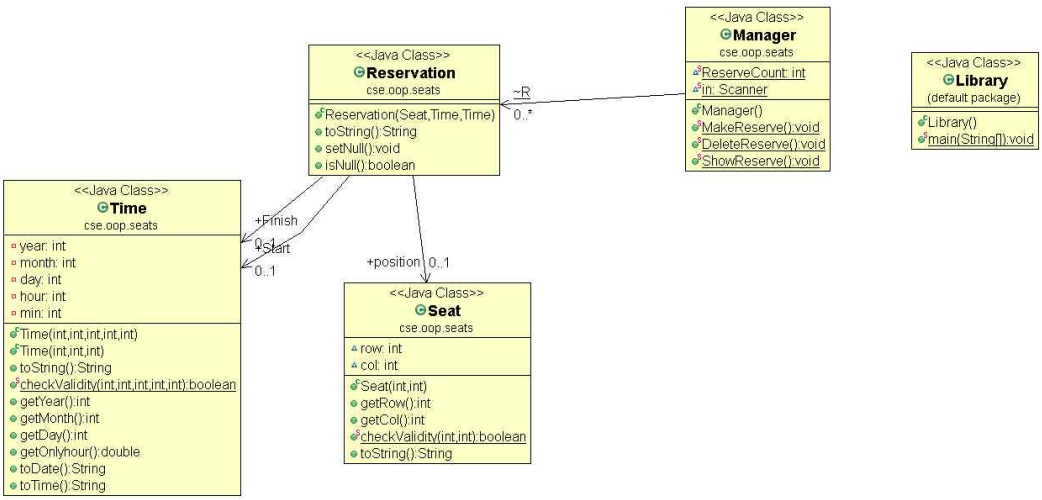
본인은 현재 자료구조 과목을 수강하는 중이고 스택, 큐, 연결리스트 등의 개념을 학습했지만,

Java 언어를 통해 이를 구현한 적이 없어 다양한 자료구조의 개념을 이번 과제를 수행하면서 적용시킬 수 없었다.

핵심 기능들이 모두 “예약” 객체를 통해 삽입, 삭제, 열람 등으로 구성되므로 연결리스트의 자료구조가 가장 적합함을 알고 있었으나,
과제의 구현에서는 “예약” 객체의 배열을 통해서만 구현하였다.

배열로 구현할 경우 데이터의 입력과 열람 기능만 구현하는 데는 문제가 없으나 “삭제” 기능이 추가 될 경우 해당 인덱스의 배열은 이미 할당된 공간이기 때문에 없앨 수가 없으므로 없는 “취급”을 하는 것으로 대신하여야 하는데, 본인은 삭제한 인덱스의 데이터를 모두 null 처리하고, 데이터의 열람(출력)시 무시하는 것으로 알고리즘을 설계했다.

– 분석/설계 (flowchart, 클래스 다이아그램 등을 이용하여 설명한다. 그림만 그리면 안되고 반드시 설명이 있어야 한다. 구조와 동작 흐름에 대해 설명한다)



제출한 프로그램의 클래스 다이아그램

위는 본인 프로그램의 대략적인 클래스 관계를 보이는 다이아그램이다.

Time 클래스와 Seat클래스는 Reservation 클래스에게 “집약관계” 이며

최종적인 프로그램의 수행은 이 Reservation 객체의 생성, 삭제, 출력을 통해 이루어진다.
Manager 클래스에는 Library 클래스의 main에서 사용될 static 함수들을 모두 정의해 놓았다.

다음은 각 클래스에 대한 세부적인 정의를 설명한다.

1. Seat 클래스

Seat 클래스는 int형 row와 col 두 개의 변수만 가진다.

Seat 클래스의 인스턴스화는 “1.Reservation 객체의 생성” 당시에만 이루어지며, 생성 당시에 각 row(행)과 col(열) 값을 10으로 제한시켜주는 checkValidity 메소드를 통해 10x10의 좌석을 가정한다.

이후 “3. Reservation 객체 출력” 기능의 구현인 Manager 클래스의 showReserve 메소드의 실행과정 중에서 좌석을 출력하기 위한 toString 메소드가 있고, 각 행과 열값을 얻기 위한 getRow와 getCol 메소드가 정의되어 있다.

2. Time 클래스

Time 클래스는 5주차 실습과제에서 구현한 코드를 많이 참조하였으며 int형 변수 Year, Month, Day, time, min (년 월 일 시 분)을 가진다.

객체를 생성할 때 잘못된 입력 값을 검출하기 위한 checkValidity 똑같이 가지며, “겹치는 예약시간의 오류검출” 이나 “하루단위 예약만 가능하도록 제한하는 기능” 의 구현을 위한 클래스 메소드인 getYear, getMonth, getDay, getOnlyhour 등의 메소드를 가진다.

마찬가지로 “3. Reservation 객체 출력” 기능의 구현을 위한 toString 메소드를 지닌다.

추가로 “2. Reservation 객체의 삭제처리” 기능의 구현도중 입력받은 좌석에 대한 예약 목록을 출력할 표현을 위한 toDate와 toTime 메소드도 정의되어 있다.

3. Reservation 클래스

여러 가지 시행착오 끝에 Reservation 클래스는 Seat클래스 하나와 Time 클래스 두 개를 aggregation 하도록 구성했다.
이 aggregation에 대한 정의는 UML 클래스 다이어그램 작성법에 대한 글인 다음의 글을 참조했으며

(https://gmlwjd9405.github.io/2018/07/04/class-diagram.html)

관련 대목은 다음과 같다.

1. 집약 관계(aggregation)
   - 한 객체가 다른 객체를 포함하는 것
     - '부분'을 나타내는 객체를 다른 객체와 공유할 수 있다.
   - '전체'를 가리키는 클래스 방향에 빈 마름모로 표시
   - 전체 객체의 라이프타임과 부분 객체의 **라이프 타임은 독립적이다.**
     - 전체 객체가 메모리에서 사라진다 해도 부분 객체는 사라지지 않는다.
   - 예시
     - 생성자에서 참조값을 인자로 받아 필드를 세팅한다.

```
public class Computer {
 private MainBoard mb;
 private CPU c;
 // 생성자
 public Computer(MainBoard mb, CPU c) {
 this.mb = mb;
 this.c = c;
 }
}
```

따라서 Reservation 클래스는고유한 변수는 지니지 않고,
좌석 정보와 시작시간, 종료시간 에 대응하는 Seat 클래스 하나와 Time 클래스 두 개의 집합으로 이루어진 클래스라고 할 수 있겠다.

이 클래스에는 "3. Reservation 객체 출력" 기능의 Reservation 단위 구현인 toString 메소드가 있으며, Seat와 Time 클래스의 toString 메소드를 이용한다. 그리고 Manager 클래스의 static 메소드인

showReserve 메소드에서 최종적으로 이용된다.

그리고 setNull 메소드는 “2. Reservation 객체의 삭제처리”의 구현에서 핵심적인 기능을 하는 메소드인데, 해당 Reservation 객체의 내부 변수값을 모두 null로 바꾸어 주는 메소드이다. 이 기능은 이후 isNull 메소드와 연계하여 모든 Reservation 객체를 보고 내부 변수값이 null이 아닌 경우에만 출력하는 알고리즘을 통해 “삭제처리”의 구현을 만들어낸다. (isNull 메소드는 “삭제처리” 된 객체를 발견하면 true를 반환하고 이외의 경우에는 false를 반환하는 메소드이다)

4. Manager 클래스

Manager 클래스는 위에서 설명한 “1, 2, 3” 기능을 구현하는 메소드 들이 모두 static 메소드로 정의되어 있는 클래스이다.

또한 static 변수인 int형 ReservationCount를 가지며, 이는 MakeReserve 메소드를 통해 Reservation객체가 하나씩 생겨날 때 마다 1씩 증가하도록 된다. 차후에 전체 예약목록에 접근하는 기능을 수행하는 과정에서 for문의 조건문에 사용된다.

그리고 전체 예약목록의 역할을 하는 Reservation 형 배열 R을 가지며 또한 static 변수이고 100개의 크기를 지닌다. 이는 생성을 할 수 있는 예약의 최대 개수가 100개라는 의미를 가진다.
(자료구조와 알고리즘 상 한번 삽입된 공간은 재활용할 수 없으므로 예약의 생성과 삭제는 모두 최대 100회까지만 가능하다.)

MakeReserve 메소드는 좌석, 시작시간, 종료시간, 예약 순으로 각 객체에 대한 데이터를 받아들인 후 checkValidity 메소드에 의한 검증이 끝나면 constructor를 통해 Seat, Time, Time, Reservation 객체를 순서대로 만들어내는 메소드이다. 가장 마지막에 생성되는 Reservation형 객체는 앞에서 생성된 Seat, Time, Time형 객체를 매개변수로 constructor통해 만들어지게 되며 이 객체의 정보는 위에서 언급한 static 배열공간인 R의 ReserveCount번째 칸에 저장된다. 이후 ReserveCount를 1개 증가시키는 것으로 메소드는 끝이난다.

```
====================================
수행할 작업을 선택하세요 :

1. 좌석 예약하기

2. 예약 취소하기

3. 전체 예약보기

4. 종료
====================================
1
Enter Seat info! ex)row col .... 3 5 =>3행 5열
1 1
Enter start time info!
2019 1 1 1 0
Enter finish time info!
2019 1 1 1 30
====================================
수행할 작업을 선택하세요 :

1. 좌석 예약하기

2. 예약 취소하기

3. 전체 예약보기

4. 종료
====================================
```

showReserve 메소드는 for문을 통해 배열 R의 각 객체를 차례로 ReservationCount 번쨰 까지 훑으며 isNull이 false인 객체만 출력시킨다. 이때 출력은 Reservation 클래스의 toString 메소드로 구현된다.

```
Library [Java Application] C:₩openjdk-11.0.2_windows-x64_bin₩jdk-11.0.2₩bin₩javaw.exe (2019. 10. 13. 오후 4:48:40)
================================
수행할 작업을 선택하세요 :

1. 좌석 예약하기

2. 예약 취소하기

3. 전체 예약보기

4. 종료
================================
3
<1 1>
날짜                    시작시간                    종료시간
2019-1-1                1:0                     1:30
================================
수행할 작업을 선택하세요 :

1. 좌석 예약하기

2. 예약 취소하기

3. 전체 예약보기

4. 종료
================================
```

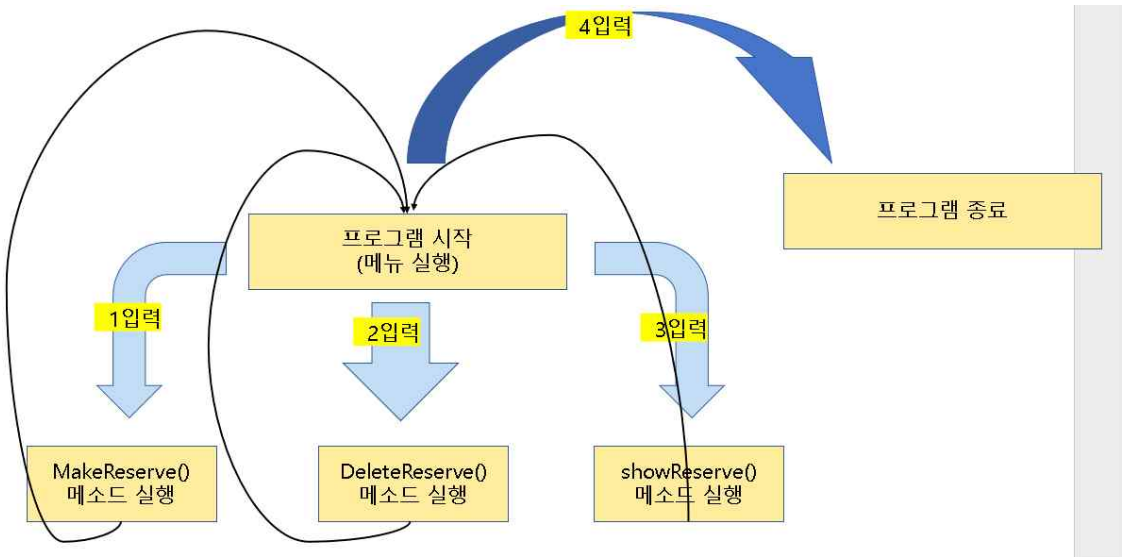
DeleteReserve 메소드는 사용자에게서 좌석번호를 받아들이고, 그 좌석번호에 대한 예약정보만 선택적으로 출력 후, 입력된 번호에 해당하는 예약 1개만 취소하는 메소드이다. 따라서 int형 변수인 row와 col을 Scanner 클래스의 nextInt 메소드를 통해 받아들이고, R에 있는 모든 Reservation형 객체를 for문을 통해 하나씩 대조하며 Reservation 속의 Seat형 객체의 row와 col과 동일한 예약 정보만 출력해서 사용자에게 보여주게 된다.

이때 다시 사용자에게 입력을 받고 예약을 삭제 처리할 것을 대비해서, tempR 이라는 R과 동일한 형태를 지닌 배열을 하나 더 메소드 내에서 생성하고 temp[indexNum]이라는 각 배열 칸에 예약정보를 출력하기 전 차례로 해당 Reservation 객체의 정보를 복사하고 출력한다.
이후 출력에 따라 사용자가 입력한 정보는 indexNum 변수에 저장되고, temp[indexNum]에 해당하는 Reservation 객체와 동일한 정보를 가진 Reservation 객체를 배열 “R"에서 찾아 setNull 메소드로 ”삭제 처리“한다.

```
수행할 작업을 선택하세요 :

1. 좌석 예약하기

2. 예약 취소하기

3. 전체 예약보기

4. 종료
=====================================
2
취소할 좌석번호를 입력하세요!
1 1
예약 현황입니다 아래에서 취소할 예약 번호를 입력하세요!
1. 2019-1-1/1:0 ~ 2019-1-1/1:30

 취소할 예약번호:
1
예약이 취소되었습니다!
=====================================
수행할 작업을 선택하세요 :

1. 좌석 예약하기

2. 예약 취소하기

3. 전체 예약보기
```

5. Library 클래스

Library 클래스는 import cse.oop.seats.*; 라는 문장을 통해 Manager, Time, Reservation, Seat의 모든 클래스를 import하게 되고 프로그램의 최종적인 구현을 위한 main 메소드만 정의되어있다. 코드의 길이가 길지 않고 코드에 대한 설명보다 전체적인 흐름의 설명이 더 중요하므로 다음의 그림을 통해 설명하도록 한다.



전체적인 main 메소드는 switch 문의 case를 통해 1, 2 ,3, 4의 입력으로 각 기능을 구현하도록 설계되어 있으며 최종적으로 4를 입력하지 않는 한 continue 문을 통해 반복적으로 기능을 수행하도록 설계되어 있다.

다음은 각 에러상황에 대한 프로그램의 예외처리를 그림과 글을 통해 설명한다.

```
수행할 작업을 선택하세요 :

1. 좌석 예약하기

2. 예약 취소하기

3. 전체 예약보기

4. 종료
==================================
1
Enter Seat info! ex)row col .... 3 5 =>3행 5열
1 13
Error: enter Seat info again!
Enter Seat info! ex)row col .... 3 5 =>3행 5열
1 1
Enter start time info!
2019 33 33 33 0
Error: enter time info again!
Enter start time info!
2019 1 1 1 0
Enter finish time info!
2019 33 33 33 0
Error: enter time info again!
Enter finish time info!
2019 1 1 1 30
==================================
수행할 작업을 선택하세요 :
```

예약생성시 입력 오류에 대한 각각의 예외처리

위는 차례대로 Seat객체 형 position, Time 객체 형 Start, Time 객체 형 Finish를 생성하는 과정에서의 MakeReserve 메소드가 지정된 값 이외의 값이 입력되었을 경우 에러문장을 출력하고 받아야되는 입력을 다시 요구하는 장면이다.
(Seat의 입력에서는 각 행과 열을 모두 10 이하의 자연수로 제한한다)

이는 모두 무한루프인 while을 통해 구현되었으며 지정된 조건을 만족할 때만 다음 단계의 무한루프 while문으로 넘어가도록 설계되었다.

```
==================================
1
Enter Seat info! ex)row col .... 3 5 =>3행 5열
1 1
Enter start time info!
2019 1 1 1 0
Enter finish time info!
2019 1 1 5 0
==================================
수행할 작업을 선택하세요 :

1. 좌석 예약하기

2. 예약 취소하기

3. 전체 예약보기

4. 종료
==================================
1
Enter Seat info! ex)row col .... 3 5 =>3행 5열
1 1
Enter start time info!
2019 1 1 2 0
Enter finish time info!
2019 1 1 3 0
시작시간이 다른 예약구간의 중간에 있습니다!! 메뉴로 이동합니다
==================================
```

시간이 겹치는 예약 생성에 대한 예외처리

위의 그림은 첫 예약이 2019년 1월 1일 1시~5시 의 예약이고
두 번째 예약이 2019년 1월 1일 2시~3시의 예약이다. 첫 번째 예약이 먼저 생성되었고 두 번째 예약은 첫 번째 예약의 시간과 명백히 겹치므로 MakeReserve 메소드에서 두 번째 예약의 시작시간인 2시가 첫 번째 예약의 1시부터 5시 사이의 구간에 있음을 감지하고 예약을 하지 못하도록 메소드를 종료시킨 후 다시 메뉴로 이동하도록 하는 그림이다.

– 결론 (이 프로그램에서 배운 점 등과 어려웠던 부분, 미구현 부분에 대해 해결책 등에 대해 기술한다.)

기능을 구현함에 있어서 입력과 출력만 요구사항이 있고 클래스의 정의나 알고리즘의 구현은 자유형식이어서 무엇보다 처음 클래스의 관계를 정의하는 것이 가장 힘들었다.

문제의 핵심을 파악하고 가장 간단하게 구현하기 위해서 Reservation 객체를 통해 예약생성, 삭제, 출력 기능을 구현하는 방법을 택했으나, 다른 방법도 있을 것이다.

예약객체 배열인 R을 구현함으로서 생기는 공간활용의 문제는 Reservation 객체를 data로 하는 연결리스트를 구현하고 그에 따른 Manager클래스의 static 메소드들의 알고리즘을 연결리스트 버전으로 수정한다면 구현할 수 있을 것이다.

입력 문자의 개수에 대한 문제점은 현재 배운 공부로서는 구현할 수 없지만 nextInt가 아닌 다른 메소드를 이용하여 지정된 문자 개수만 입력되도록 제한하는 방법으로 구현할 수 있을 것이다.

하나의 프로젝트를 진행하면서 정말 작은 프로젝트임에도 불구하고 코딩을 하는 과정보다 요구사항을 분석하고 그에 대한 설계를 하는 과정이 얼마나 중요한지를 실감하게 되었다.

현업에서 쓰이는 프로그래밍은 이번 과제와는 비교도 되지 않을 정도의 규모일 텐데 단순 코딩 능력이 아닌 그런 실무능력을 가지기 위한 분석과 설계를 얼른 전문적으로 배우고 싶다는 열망을 가지게 되었다.

이후 수강할 과목인 도메인 분석 설계 과목을 더 주도적으로 내가 원하는 지식을 얻기 위해 수강할 수 있겠다고 느꼈으며, 현재 수강중인 소프트웨어 공학 과목에서의 소프트웨어 개발을 위한 각 프로세스의 중요성을 다시금 깨닫게 되었다.

이상으로 이번 프로젝트 보고서를 마친다.