2019년 2학기 객체지향프로그래밍
-1차 프로그래밍 과제

좌석 예약 시스템

실습반 318호 / 남경진 조교

이름 신지수

학번 201621818

학과/학년 심리학과 / 4

Ⅰ. 서론

본 과제는 프로그래밍 언어 중 Java로 좌석 예약 프로그램을 구현하는 목표를 가진다. 좌석 예약 프로그램은 세 가지 기능을 수행한다. 좌석 예약하기, 예약 취소하기, 전체 예약 내역 보기가 그것이다.

첫 번째 기능인 좌석 예약은 유효한 날짜와 시간을 입력받는다. 만약 유효하지 않은 값을 받은 경우에는 오류 메시지를 출력하고 다시 입력 받는다. 유효한 값을 받은 경우에는 새로운 예약과 기존의 예약을 비교한다. 중복된 예약이라면 예약을 받지 않고 메뉴로 돌아간다. 중복이 아니라면 예약을 성공적으로 마친다.

두 번째 기능인 예약 취소는 좌석 값을 입력 받고 해당 좌석에 등록된 예약을 모두 출력한다. 사용자로부터 취소하고자 하는 항목을 입력 받아 해당 예약 내역을 취소한다.

세 번째 기능인 전체 예약 내역 보기는 좌석을 순서대로 출력하며, 해당 좌석에 등록된 모든 예약을 시간 순으로 출력한다.

기능을 구현하기 위해 네 가지의 클래스를 만들고 사용하였다. 첫째로, Time 클래스는 날짜와 시간을 인스턴스 변수들로 가지고, 그 변수들에 대해 출력, 반환하는 등의 메소드를 가지고 있다. 또한 시간, 예약정보가 유효한지 검사하는 메소드를 가진다. 둘째로, Seat 클래스는 예약 정보를 가지는 클래스이다. 어느 좌석이 갖는 예약 정보(시작 시간, 종료 시간)를 인스턴스 변수로 가진다. 또한 예약 정보의 중복 여부를 판단하는 메소드를 가진다. 셋째로, Manager 클래스는 프로그램의 세 기능을 구현하는 메소드들을 가진다. 예약, 예약 취소, 전체 예약 보기 기능과 메뉴 출력을 수행한다. 마지막으로, ReservationTest 클래스는 예약 과정을 진행하는 메인 함수를 가진다. 클래스에 관한 자세한 설명은 본론에서 이어가고자 한다.

Ⅱ. 본론

1. 프로그램 설명

본 프로그램은 도서관 열람실과 같은 곳에서의 좌석 예약을 관리하는 프로그램이다. 프로그램에서는 편의를 위해 행이 1부터 100까지, 열이 1부터 100까지인 100 x 100 행렬 형식의 좌석을 가정하였다. 각 좌석의 번호는 행렬의 표기와 같이 <행 번호, 열 번호>로 표시한다. 프로그램에는 종료 외에 세 가지 기능을 제시한다.

첫 번째 기능인 '좌석 예약하기'에서는 한 번에 하나의 좌석만을 예약할 수 있다. 예약을 할 때에는 순서대로 좌석 번호, 사용할 시간(시작 시간, 종료 시간)을 입력한다. 시간은 '연 월 일 시 분' 형식으로 띄어쓰기로 구분하여 입력한다. 이 때, 시간 정보는 현실에 존재하는 시간이어야 한다. 날짜의 경우, 2019년 10월 15일은 유효한 날짜이지만 2019년 13월 33일은 유효하지 않기 때문에 예약이 불가능하다. 시간의 경우, 25시 0분은 유효하지 않은 시간이기 때문에 예약이 불가능하다. 추가적으로, 연도에서 기원전은 유효하지 않은 시간으로 제약했으며, 분 단위는 30분 단위로 하기

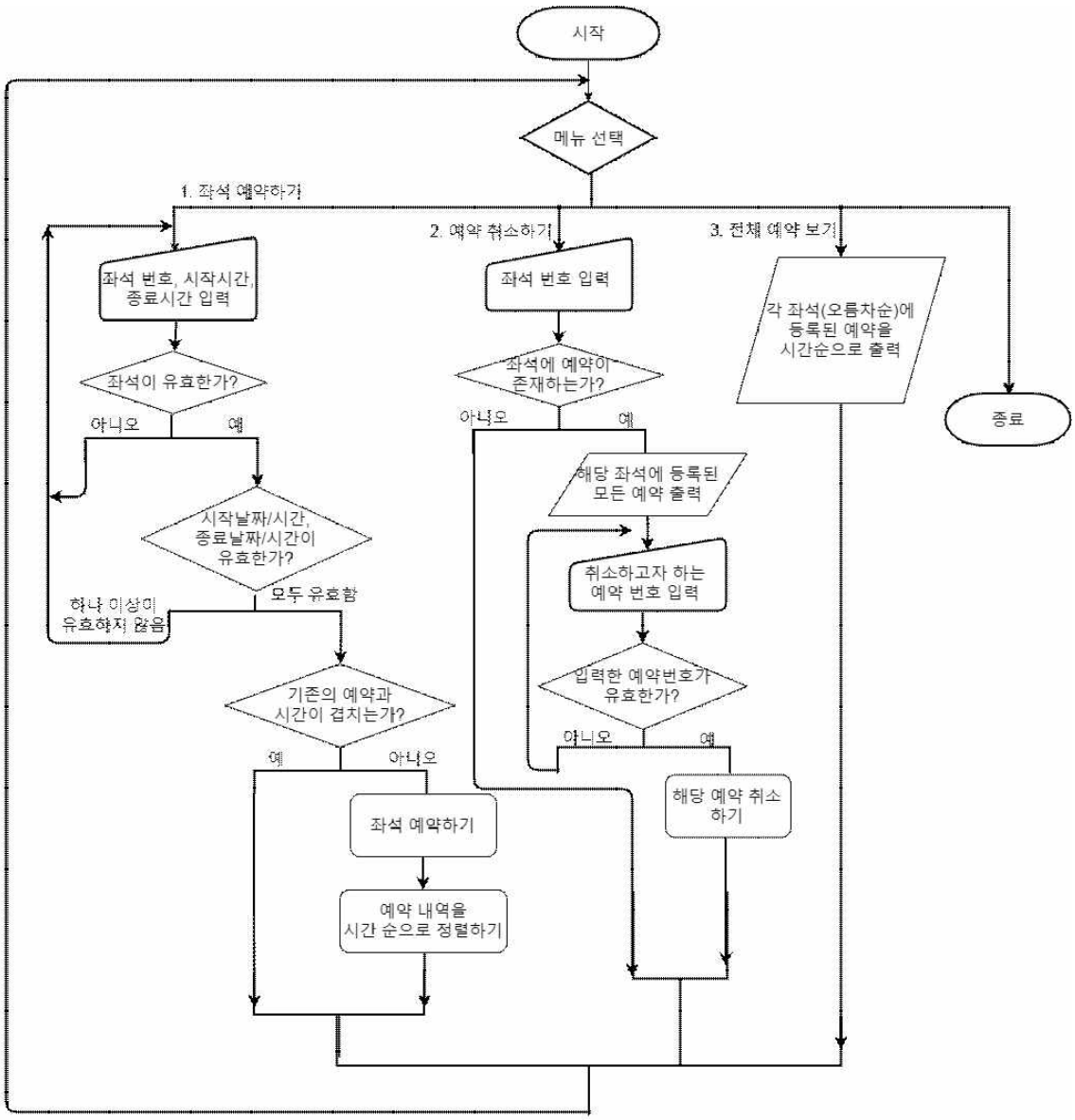

그림 1 좌석 예약 프로그램의 Flow Chart

로 결정하였다. 예를 들면, 11시 0분이나 11시 30분은 가능하나 11시 15분은 불가능하다. 유효하지 않은 경우가 또 존재하는데, 이는 시작 시간과 종료 시간의 관계에 있다. 만일 시작 날짜와 종료 날짜가 같지 않은 경우는 예약이 불가능하다. 또한 시작 시간보다 종료시간이 더 이를 경우 예약이 불가능하다. 제시된 여러 조건 중 하나라도 유효하지 않을 경우, 다시 세 가지 정보를 입력해야 한다. 유효한 예약 정보를 받았을 경우에는 기존의 예약과 중복 여부를 확인한다. 만약 중복될 경우 예약에 실패하고 메뉴 화면으로 돌아간다. 예약에 성공하였을 경우에는 예약에 성공했다는 메시지가 화면에 출력된다.

두 번째 기능인 '좌석 예약 취소하기'는 등록된 예약 중 하나를 취소하는 기능이다. 먼저 취소하고자 하는 좌석 번호를 입력받는다. 해당 좌석에 예약이 등록된 것이 없

다면 메뉴 화면으로 돌아간다. 등록된 예약이 존재할 경우, 예약을 시간 순으로 출력하여 취소할 항목을 사용자가 선택하게 한다. 사용자로부터 번호를 입력받으면 해당 순서의 예약은 취소된다.

세 번째 기능인 '전체 좌석 예약 보기'는 모든 좌석의 모든 예약을 출력한다. 예약이 되어있는 경우에만 좌석 번호와 예약 내역을 출력한다. 이 때 좌석 번호는 오름차순으로, 예약 내역은 시간 순으로 출력한다.

프로그램은 사용자가 종료하기 전까지 계속 동작한다. 입력에 따른 어떤 기능을 마치고 다시 메뉴로 돌아가 사용자의 입력을 받는다. 전체적인 과정에 대한 플로우 차트를 그림 1로 제시하였다.

2. 자료구조

프로그램을 구현하기 위하여 연결 리스트(linked list)를 사용하였다. 예약 취소 시, 원하는 순서의 예약을 삭제해야 했기에 삭제 순서가 정해져 있는 큐와 스택은 자료구조로 적절하지 않았다. 또 다른 자료구조인 3차원 배열로 프로그램을 구현할 수도 있었다. 그러나 한 좌석에 예약이 얼마나 저장될지 모르기에 배열을 사용한다면 저장 공간을 넉넉하게 잡아야 한다. 만약 한 좌석에 저장할 수 있는 최대 예약 건수가 100건이라고 가정한다면 배열을 100*100*100칸을 잡아야 한다. 이 경우에는 메모리 낭비가 심하기 때문에 적절하지 않다. 따라서 예약을 하면 동적으로 메모리를 잡고 이를 해당 좌석에 연결해주는 연결 리스트로 프로그램을 구현하였다.

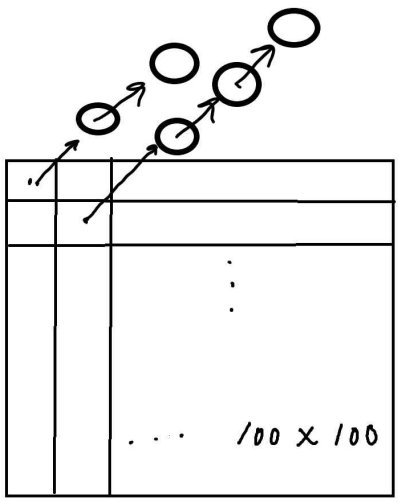

그림 2 연결 리스트

그림 2는 좌석 예약 자료구조를 시각화한 것이다. 그림에서 사각형은 좌석을 의미하고 원은 예약을 나타낸다. 화살표는 좌석에 예약이 연결되어 있음을 의미한다. 100x100의 좌석을 마련하여 하나의 좌석이 리스트의 head로서 예약 내역을 가리키는 구조이다. 또한 여러 개의 예약은 다음 예약을 가리켜서 연결되어있다. 연결 리스트로 구현하기에 메모리를 필요한 만큼만 사용할 수 있고, 데이터 삽입, 삭제, 정렬,

검색이 자유롭다.

3. 클래스

설명한 프로그램을 구현하기 위해 네 가지의 클래스를 구현하였다. 네 가지 클래스를 간단하게 정리한 다이어그램을 아래에 제시하였다.

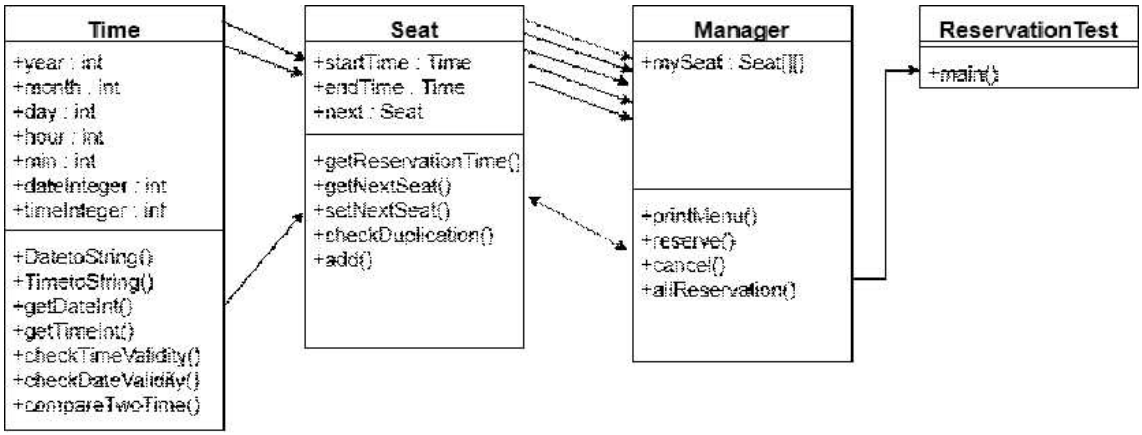

그림 3 클래스 구성도

1) Time

Time 클래스는 시간에 관련된 정보를 필드로 가지며, 이를 다루는 메소드, 그리고 시간과 관련된 여러 static 메소드를 가진다. 인스턴스 변수와 메소드에 관한 자세한 설명은 아래의 표로 제시하였다.

표 1 Time 클래스 인스턴스 변수: 흰색 배경 / 메소드: 회색 배경

이름	역할 / 기능	특징	
year	연도	기원전은 배제하며 0년 이후부터 유효함	
month	월	1부터 12까지의 값만 유효함	
day	일	2월	윤년이면 1~29일 유효
			윤년이 아니면 1~28일 유효
			윤년 판단을 위해 GregorialCalander 클래스의 isLeapYear()메소드를 사용
		1,3,5,7, 8,10,12월	1~30일 유효
		4,6,9,11월	1~31일 유효
hour	시	24시간 체제를 사용하기로 하며, 0~23시의 값이 유효함	
min	분	30분 단위이기에 0과 30만 유효함	
dateInteger	날짜 비교의 편의성을 위해 yyyymmdd형식의 정수로 변환하여 저장	예) 2019년 10월 15일 -> (int)20191015	

timeInteger	시간 비교의 편의성을 위해 hhmm 형식의 정수로 변환하여 저장	예) 11시 30분 -> (int)1130
DatetoString()	날짜/시간을 String으로 반환함	예) 2019년 10월 15일 -> 2019-10-15
TimetoString()		예) 11시 30분 -> 11:30
getDateInt()	dateInteger를 반환함	
getTimeInt()	timeInteger를 반환함	
checkTimeValidity()	사용자가 입력한 예약 정보 중 시간 정보가 유효한 값인지 확인함	static 함수 입력한 시간이 연, 월, 일, 시, 분의 유효조건에 부합하는지 판단함
checkDateValidity()	사용자가 입력한 시작시간과 종료시간이 유효한지 확인함	static 함수 시작날짜와 종료날짜가 동일한지 판단함
compareTwoTime()		static 함수 시작시간에 비교하여 종료시간이 이르거나 같다면 유효하지 않음

dateInteger와 timeInteger는 객체 생성 및 생성자 호출 시 year, month, day, hour, min 변수를 초기화 하면서 함께 계산하여 저장한다.

checkTimeValidity(), checkDateValidity(), compareTwoTime()은 static 메소드로 선언하였다. static 메소드는 non-static과는 다르게 특정 객체가 아닌 클래스에 의존한다. 따라서 객체를 생성하지 않아도 Time.checkTimeValidity()와 같이 메소드를 실행할 수 있다. 또한 static 메소드는 객체마다 각각 존재하는 것이 아닌 클래스가 로딩될 때 딱 하나만 메모리에 올라간다. 이 좌석 예약 프로그램에서 Time 객체가 생성될 때는 예약이 성공적일 때이다. 좌석 예약 시 사용자의 모든 입력 값에 대하여 Time 객체를 생성하고 객체의 메소드로 유효성을 판단하였더라면, 예약 실패로 쓸모가 없어지는 객체들이 많았을 것이다. 따라서 메모리가 낭비된다. 그래서 이 프로그램에서는 먼저 입력 값에 대하여 static 메소드로 그 유효성을 판단하였고, 그 이후에 객체를 생성하였다.

2) Seat

Seat 클래스는 좌석에 저장된 예약 내역을 정보로 가진다. Seat 객체는 예약의 시작시간과 종료시간을 인스턴스 변수로 가진다. 또한 자료구조를 예약내역끼리 서로 가리키는 연결 리스트로 만들었기 때문에 Seat 클래스가 연결 리스트와 관련된 변수와 메소드를 가지고 있다. 다음 예약 내역(Seat)을 가리키는 레퍼런스 변수, 연결 리스트 내 정보를 추가하거나 검색하는 메소드 등이 그러하다. Seat 클래스의 인스턴스 변수와 메소드를 다음의 표에 제시하였다.

표 2 Seat 클래스 인스턴스 변수: 흰색 배경 / 메소드: 회색 배경

이름	역할 / 기능	특징
startTime	예약 시작시간	Time 객체
endTime	예약 종료시간	Time 객체

next	다음 예약(Seat)을 가리키는 레퍼런스 변수	Seat 타입
getReservationTime()	예약 정보를 String으로 반환함	예) 2019-10-15 7:0 13:30 startTime, endTime 객체의 DatetoString(), TimetoString() 호출
getNextSeat()	Seat 객체가 가리키는 다음 Seat 객체를 반환함	
setNextSeat()	Seat 객체가 다른 Seat 객체를 가리키도록 연결함	
checkDuplication()	해당 좌석에 예약된 내역들 중 새로운 예약과 겹치는 내역이 있는지 확인함	비교를 위해 getDateInt(), getTimeInt() 메소드를 호출함 예약이 겹친다면 false 반환
add()	새로운 예약(Seat)을 시간 순서에 맞게 연결 리스트에 삽입	삽입 정렬을 사용

checkDuplication() 메소드는 새로운 입력 값이 기존의 예약과 겹치는지 판단한다. 특정 좌석에 등록된 모든 예약에 반복문으로 접근하여 조건에 부합하는지 확인한다. 먼저 해당 좌석에 아무런 예약도 연결되어 있지 않는다면 메소드는 중복이 없다는 의미의 true를 반환한다. 연결된 기존의 예약이 있다면 기존의 예약과 새로운 예약의 날짜가 일치하는지 확인한다. 날짜가 일치하지 않는다면 중복 여부를 판단할 필요가 없다. 날짜가 일치하는 경우에는 시간의 중복에 대하여 다음의 그림과 같이 판단한다.

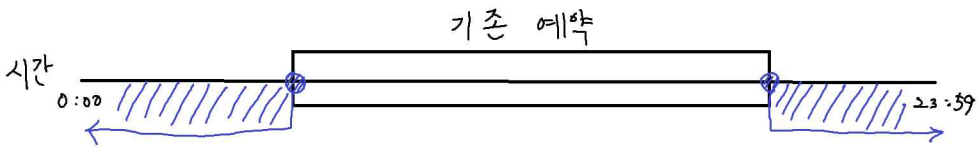

그림 4 빗금 친 부분이 중복이 되지 않는 구간이다.

㉠ 새로운 예약의 종료시간이 기존예약의 시작시간보다 이르거나 같은 경우
㉡ 새로운 예약의 시작시간이 기존예약의 종료시간보다 느리거나 같은 경우

둘 중 하나의 조건을 만족하면 두 예약은 겹치지 않는다. 그러나 이를 둘 다 만족시키지 않는다면 예약이 중복되기에 false를 반환한다. 이 비교에서는 연, 월, 일, 시, 분을 각각 비교하는 것이 아닌 이를 하나로 합친 dateInteger과 timeInteger로 값을 비교하는 방법을 사용하였다. 예를 들어, 11시와 15시 30분을 비교하는 경우, 11과 15를 비교하고 0과 30을 비교하여 판단하는 것보다 1100이 1530보다 작기에 11시가 더 이르다고 판단하는 것이 더 편리하다.

add() 메소드는 새로운 예약을 저장할 때 사용한다. 예약 취소와 전체 예약보기 기능에서 예약을 시간 순으로 출력해야 하기 때문에, 삽입 정렬을 통해 시간 순으로 저

장되게 하였다. 새로운 예약을 저장할 때, 네 가지 상황을 고려할 수 있다.
㉠ 특정 좌석에 대한 연결 리스트가 비어있을 경우
- 새 예약을 리스트의 첫 번째 자리에 저장한다.
㉡ 연결 리스트의 첫 번째 예약보다 더 빠를 경우 (날짜가 이르거나 날짜는 동일하나 시간이 이른 경우)
- 새 예약을 리스트의 첫 번째 자리에 저장한다.
㉢ 새 예약이 다음 예약보다 이를 경우 (새 예약이 중간에 낄 경우)
㉣ 새 예약이 가장 마지막일 경우
ㄱ과 ㄴ의 경우에는 새 예약이 앞에 저장되기 때문에 연결 리스트의 head를 새 예약으로 바꿔주는 작업 또한 필요하다.

3) Manager

표 3 Manager 클래스 인스턴스 변수: 흰색 배경 / 메소드: 회색 배경

이름	역할 / 기능	특징
mySeat	좌석을 의미함	Seat[][] 타입 Seat 객체를 참조하는 변수를 100*100 배열로 선언함
printMenu()	메뉴를 출력하고 사용자가 원하는 기능을 입력 받음	static 함수
reserve()	좌석 예약하기 기능	
cancel()	좌석 취소하기 기능	
allReservation()	전체 예약 보기 기능	

Manager 클래스는 프로그램 예약 프로그램을 담당하는 역할을 한다. mySeat은 Manager 객체 하나가 관리해야 하는 좌석들을 의미한다. 좌석들에 예약정보인 Seat 객체를 연결해야 하기에 Seat[100][100] 배열로 선언하였다. 이 프로그램에서는 Manager 객체를 하나만 선언하여 한 장소에 대해서만 예약이 가능하게 구현했다. 만약 우리 학교 도서관의 C 열람실, D 열람실처럼 장소가 여러 곳이라면 객체를 여러 개 생성하여 좌석을 각각 따로 관리하면 된다.

printMenu()는 모든 Manager 객체들이 같은 함수를 동일하게 사용할 것이기 때문에 메모리를 아끼고자 static 함수로 선언하였다. 이는 메모리에 한 번만 올라가고 수차례 호출될 것이다.

4) ReservationTest

마지막 클래스는 main()만 존재한다. 필요한 만큼의 Manager 객체를 생성하여(이 프로그램에서는 하나의 Manager 객체만 존재한다.) 사용자로부터 메뉴를 입력 받고, 이를 진행한다.

4. 기능

0) 메뉴 선택

메뉴를 출력하고 사용자로부터 원하는 기능을 입력 받는다.

```
메뉴를 선택하세요.
1. 좌석 예약하기
2. 예약 취소하기
3. 전체 예약보기
4. 종료
>
```

1) 좌석 예약하기

㉠ 좌석 번호, 시작시간, 종료시간 입력하기

```
메뉴를 선택하세요.
1. 좌석 예약하기
2. 예약 취소하기
3. 전체 예약보기
4. 종료
> 1

예약할 좌석을 선택하세요!
좌석번호: 1 1
시작시간: 2019 10 16 11 0
종료시간: 2019 10 16 12 0
```

㉡ 유효한 좌석인지 확인하기

```
예약할 좌석을 선택하세요!
좌석번호: 101 2
시작시간: 2019 10 16 14 0
종료시간: 2019 10 16 15 0
예약이 불가능합니다!
```

유효한 좌석인지 확인하여 유효하지 않을 경우 다시 예약정보를 입력하도록 한다. 만약 바로 돌아가지 않는다면, 예약 중복 여부를 확인하는 과정에서 mySeat 배열의 범위를 벗어난 곳을 참조하고자 하여 오류가 발생한다. 좌석 번호의 행과 열이 모두 1~100 내의 값이라면 다음 과정으로 넘어간다.

㉢ 시작시간, 종료시간이 유효한지 확인하기

```
예약할 좌석을 선택하세요!
좌석번호: 1 1
시작시간: 2019 13 20 12 0
종료시간: 2019 13 20 15 0
예약이 불가능합니다.
```

```
예약할 좌석을 선택하세요!
좌석번호: 1 1
시작시간: 2019 10 16 15 0
종료시간: 2019 10 17 9 0
예약이 불가능합니다.
```

checkTimeValidity(), checkDateValidity(), compareTwoTime()를 사용하여 유효

성을 판단한다. 유효하지 않은 값이 있다면 다시 예약정보를 입력하게 한다.

㉣ 기존의 예약과 시간이 겹치는지 확인하기

```
예약할 좌석을 선택하세요!
좌석번호: 1 1
시작시간: 2019 10 16 11 0
종료시간: 2019 10 16 13 0
해당 시간에 예약이 존재합니다. 예약이 불가능합니다.
```

checkDuplication()을 이용하여 예약시간이 겹친다면 오류 메시지를 출력하고 메뉴로 돌아간다.

㉤ 성공적인 예약

```
예약할 좌석을 선택하세요!
좌석번호: 1 1
시작시간: 2019 10 16 11 0
종료시간: 2019 10 16 12 0
좌석 <1,1>이 예약되었습니다.
```

add()를 이용하여 좌석의 연결 리스트에 추가한다.

2) 예약 취소하기

㉠ 좌석 번호 입력하기

```
메뉴를 선택하세요.
1. 좌석 예약하기
2. 예약 취소하기
3. 전체 예약보기
4. 종료
> 2

취소할 좌석번호를 입력하세요!
좌석번호:
1 1
```

㉡ 해당 좌석에 예약이 존재하는지 확인하기

```
취소할 좌석번호를 입력하세요!
좌석번호:
2 2
해당 좌석에 대한 예약이 없습니다.
```

만약 좌석 번호가 유효하지 않거나(0<행, 열<101이 아닌 경우) 해당 좌석에 예약이 존재하지 않는다면 오류 메시지를 출력하고 메뉴로 돌아간다.

㉢ 취소하고자 하는 예약 번호가 유효한지 검사하기

```
취소할 좌석번호를 입력하세요!
좌석번호:
1 1
예약 현황입니다. 아래에서 취소할 예약 번호를 선택하세요!
1. 2019-3-13    14:0    17:0
2. 2019-5-5     14:0    17:0
3. 2019-10-16   11:0    12:0
취소할 예약번호:
4
해당하는 번호의 예약이 없습니다.
```

만약 취소하고자 하는 예약 번호가 존재하지 않는다면 다시 취소할 예약번호를 입력받는다. 사진에서는 1, 2, 3 외 다른 숫자를 입력할 시 재입력을 요구한다.

㉣ 성공적인 예약 취소

```
취소할 예약번호:
1
예약이 취소되었습니다!
```

3) 전체 예약 출력하기

```
메뉴를 선택하세요.
1. 좌석 예약하기
2. 예약 취소하기
3. 전체 예약보기
4. 종료
> 3

=== 전체 예약 목록 ===
< 1,1 >
 날짜              시작시간   종료시간
2019-5-5          14:0     17:0
2019-10-16        11:0     12:0
< 2,2 >
 날짜              시작시간   종료시간
2019-10-30        1:0      7:0
```

4) 종료하기

```
메뉴를 선택하세요.
1. 좌석 예약하기
2. 예약 취소하기
3. 전체 예약보기
4. 종료
> 4

+----------------------+
 좌석 예약 시스템이 종료됩니다.
+----------------------+
```

Ⅲ 결론

이번 과제를 통해 처음으로 여러 클래스로 하나의 프로그램을 구성하여 보았다. 객체지향 프로그래밍의 개념이 낯설어서 적절하게 클래스를 만들고 그 역할을 나누는 것이 어려웠다. 또한 각 인스턴스 변수와 메소드를 어느 클래스에 넣어야 하는지도 많은 고민을 요했다. 처음부터 완벽한 설계를 하고 코딩을 한 것이 아니다보니 코딩 도중에 코드를 이곳저곳으로 옮기기도 하고, 객체의 생성과 삭제를 번복하는 등 복잡한 과정을 거쳤다. 그러다보니 클래스 내부가 많이 복잡해져서 코드를 수정할 엄두가 나지 않았다. 프로그램을 먼저 정확히 이해하고 그에 따른 클래스의 정의와 기능을 정리하는 것이 중요함을 알게 되었다.

또한 링크드 리스트의 구현이 조금 까다로웠는데 이번 과제를 통해 링크드 리스트의 삽입, 삭제, 검색 시 고려해야 하는 부분을 복습하게 되었다. 추가적으로 C언어에서는 연결 리스트에서 포인터 개념이 중요한데, Java에서는 포인터 대신 레퍼런스 변수를 사용하였다. 포인터와 레퍼런스 변수 모두 가리키는 역할을 한다는 공통점이 있지만, 포인터가 가리키는 것의 값을 쓰려면 *을 붙여야 하지만 레퍼런스 변수는 그 자체로 가리키는 것의 값을 사용할 수 있다는 차이점이 있었다. 이 또한 링크드 리스트의 링크 변수를 다루면서 생각해볼 수 있었다.

아쉬운 점은 링크드 리스트의 삽입, 삭제, 검색의 코드를 다른 메소드로 분리해서 모듈화를 잘 하고 싶었는데, 코드가 복잡해져서 deletion() 메소드를 만들지 못했다는 점이다. 또한 클래스의 이름이 적절하지 못한 것 같아 아쉽다. 프로그램을 다 구현하고 보니 Seat 클래스가 예약정보를 담고 있기 때문에 Reservation으로 이름을 붙여야 하지 않을까 생각했다.

구현하지 못한 부분은 예약 정보를 입력 시, 입력해야 하는 숫자보다 더 많은 숫자를 입력했을 경우 Scanner에서 오류가 발생한다. 입력 스트림을 비우는 방법을 생각하여 구현하고 싶었으나 이해하지 못해서 구현하지 못했다.