스터디룸 예약 프로그램

설계 문서 수정 1 판

A04 팀

201910184 김민기
201911286 최민호
201911294 홍승택
202111294 박성준
202111309 손 본
202111332 유의진

변경 내역 요약

[2.1] "생성자 내부에서 하는 일" 에서 optimize 이후 생략된(빠진) 부분(유저 파일 관련) 추가
[2.1] 2 차 확장 때 추가된 부분에 대한 구현 방법 추가
[2.2] "메소드가 하는 일" 에서 오타 수정(pepleNum 에서 peopleNum 로 변경)

1. Optimizer: 스터디룸 최적화 알고리즘 구현

스터디룸 최적화 알고리즘을 수행하기 위해서 별도의 클래스를 사용합니다. 클래스 이름은 "Optimize"이며, 구현 파일(.cpp)와 헤더 파일(.h)을 생성합니다. 외부 클래스에서 해당 클래스의 메소드를 사용할 때는 사용 클래스 상단에 "#include "Optimize.h""를 선언한 뒤 public 메소드를 사용할 수 있습니다.

클래스 헤더파일(Optimize.h)

해당 클래스 헤더는 1 개의 public static 메소드와 2 개의 private 메소드를 갖습니다. 또한 외부 클래스에서 메소드의 인터페이스 및 세부 정보는 아래에서 설명합니다. 해당 클래스를 static class 처럼 사용하기 위해 public 의 기본 생성자를 delete 합니다.

1.1 optimize

인터페이스

```
vector<vector<string>> Optimize::optimize(string date, string startTime, string endTime, string roomIdStr)
```

메소드가 하는 일

optimize 클래스의 메소드 중 외부에서 호출 가능한(public static) 유일한 메소드입니다. 새롭게 예약할 정보를 인자로 받아 최적화 알고리즘 수행 후, 해당 예약이 가능하다면 그때의 스터디룸 예약 상태를 반환합니다.

인자

각 데이터요소의 표준형식을 따르며, 의미규칙을 만족함을 보장하는 4 개의 인자를 전달 받습니다.

1. string date: 예약하고자 하는 날짜를 의미합니다. "날짜"의 표준 형식을 따릅니다.
   표준 형식: <4 자리 숫자>-<2 자리 숫자>-<2 자리 숫자>로 이루어진 문자열
   이때, 앞 4 자리 숫자는 연(year), 중간 2 자리 숫자는 월(month), 끝 2 자리 숫자는 일(day)로 해석합니다.
   예를 들어, "2022-10-09"는 표준 형식으로 나타낸 날짜이며, 2022 년 10 월 09 일을 의미합니다.
2. string startTime: 예약 시작 시각으로, "시간"의 표준형식을 따르며, 시간 보정이 적용되어 있음을 보장합니다.
   표준형식 : <숫자 2 개>:<숫자 2 개>형태의 문자열
   시간보정 : 기존의 시간으로부터 가장 가까운 30 분 단위의 시간으로 변환합니다.
   따라서 인자의 형식은 "<숫자 2 개>:00" 또는 "<숫자 2 개>:30"입니다.
3. string endTime: 예약 종료 시각을 뜻합니다. StartTime 과 같이 "시간"의 표준 형식을 따르며, 시간 보정이 적용되어 있음을 보장합니다.
4. string roomIdStr: 예약할 스터디룸 방 번호입니다. "방번호"의 표준형식을 따릅니다.
   표준 형식: 숫자로만 이루어진 길이가 1 인 문자열

반환

최적화 알고리즘 수행 후,

1. 인자 정보대로 예약이 가능할 경우, 예약 날짜의 스터디룸 예약 상태를 반환합니다. 단, 인자로 주어진 예약 정보는 포함되어 있지 않습니다. 해당 인덱스에 다른 값(garbage value)가 있을 수 있습니다.
2. 예약을 할 수 없는 경우, 벡터 {{}}를 반환합니다.

구현

File 클래스에서 입력된 날짜의 스터디룸 예약 현황을 담은 vector<vector<string>> data 를 반환받아 저장합니다. 이 변수는 최적화 알고리즘을 수행하며 값이 변하게 되고, 최적화에 성공했다면 변화한 data 를 반환합니다.
내부 함수에서 용이하게 사용하기 위해, 인자로 받은 roomIdStr 를 stoi() 메소드를 이용해 int 타입의 roomId 으로 변환합니다. 또한 인자로 받은 startTime, endTime 을 convertTimeToIndex 메소드를 사용하여 data 의 인덱스로 변환하고, 각각 int 타입의 startTimeIndex, endTimeIndex 에 할당합니다.

예약 시간을 30 분 단위로 잘라 알고리즘을 적용하기 위해, [startTimeIndex, endTimeIndex) 범위를 for 문을 이용해 아래 내용을 반복 실행합니다. TimeIndex 가 i 일 때, data[roomId][i]에 다른 예약이 있다(즉, "0"이 아니라)면, 클래스 메소드인 go(...)를 호출합니다. go 메소드로부터 반환되는 bool 값이 false 라면, 인자 정보대로 예약이 불가능한 경우이므로 {{}} 반환하고 메소드를 종료합니다.

For 문 실행 도중에 반환되어 메소드가 종료되지 않고 성공적으로 수행했다면, 처음에 File 클래스에서 받은 data 변수의 값이 최적화된 스터디룸 상태로 업데이트 되어 있습니다. 따라서 data 를 반환합니다.

1.2 go

인터페이스

```
bool Optimize::go(vector<vector<string>> &data, int reserveId, int targetRoomId,
int peopleNum, int timeIndex)
```

메소드가 하는 일

실제 최적화 알고리즘을 수행하는 메소드로, 클래스 내부(private)에서만 호출할 수 있습니다.

재귀적으로 호출되면서 최적화 알고리즘을 수행합니다. 인자로 받은 예약(이하 **target**)을 같은 시간, 다른 스터디룸(이하 **dest**)으로 이동시킬 수 있는지 여부를 반환합니다.

인자

5 개의 인자를 전달 받습니다.

1. vector<vector<string>>& data: 업데이트 된(또는 초기) 스터디룸 예약 상태를 저장하고 있는 인자입니다. 9 행 22 열의 크기를 가지고 있습니다.
2. int reserveId: target 의 예약 번호로, 1 이상의 정수입니다.
3. int targetRoomId: target 의 스터디룸 방 번호(행 인덱스)를 뜻합니다. 1 부터 9 까지의 한 자리 정수만이 들어올 수 있습니다.
4. int peopleNum: target 의 예약 인원 수를 뜻하며, 길이가 2 이하인 정수입니다.
5. int timeIndex: data 에서 해당하는 시간(열 인덱스)을 뜻합니다. 0 부터 21 까지의 정수 값을 가집니다.

반환

target 을 옮길 수 있는 dest 가

1. 존재하고
   1-1. 다른 예약이 없는 빈 스터디룸이라면: true 반환
   1-2. 다른 예약이 존재한다면: dest 를 인자로 넣은 go(...)를 재귀 호출한 결과를 반환
2. 존재하지 않다면, false 반환

구현

int 타입 3 개인 튜플을 갖는 벡터 possibleStudyRoom 를 선언합니다. 이 벡터는 target 이 이동할 수 있는 스터디룸을 담은 벡터입니다. 튜플 순서대로 (예약 번호, 스터디룸 방 번호, 스터디룸 제한 인원 – target 의 인원 수)를 의미합니다.

스터디룸의 방 번호를 id 라 할 때, 모든 스터디룸([1, 10))에 대해 for 문을 수행합니다.

현재 target 의 스터디룸 방 번호와 id 가 같다면 **continue.**
File 클래스에서 id 의 제한 인원을 int 타입의 limitNum 에 대입합니다.
target 의 인원 수가 limitNum 보다 크다면 **continue.**

id 에 다른 예약이 없다면(즉, data[id][timeIndex]=="0")이라면, possibleStudyRoom 에 (0, id, limitNum - peopleNum) 를 추가합니다.
id 에 다른 예약이 있다면, File 에서 dest 의 예약 인원 수를 받아와서 target 의 인원 수와

비교합니다. target 의 인원 수가 크다면, possibleStudyRoom 에 (data[id][timeIndex], id, limitNum - peopleNum) 를 추가합니다.

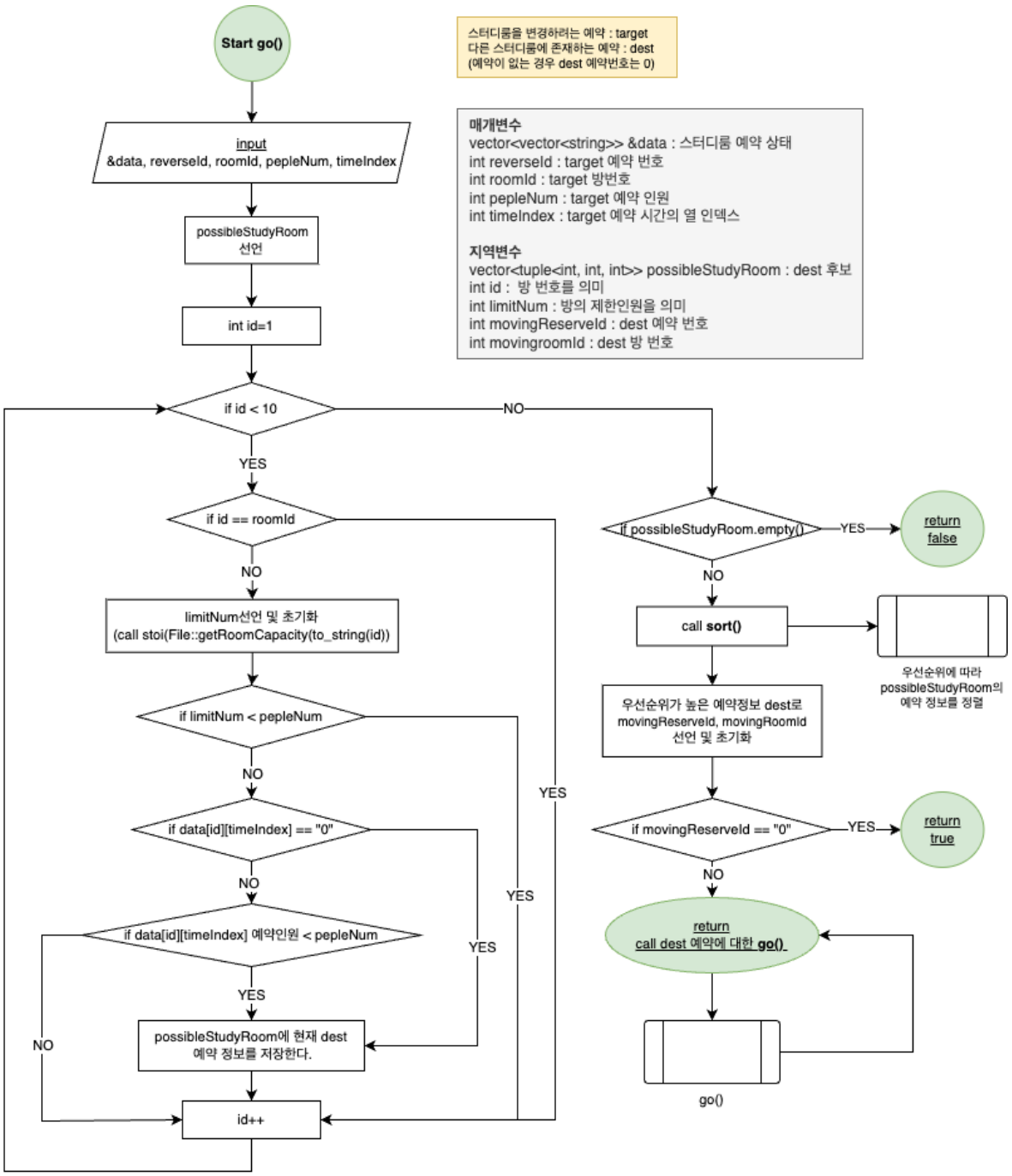
for 문 종료 후 possibleStudyRoom 가 비어있다면, target 은 그 어떤 스터디룸으로도 옮길 수 없으므로 false 를 반환합니다.

우선순위에 맞게 possibleStudyRoom 을 정렬합니다. 정렬 우선순위는 아래와 같습니다.

1. 현재 예약이 되어있지 않은 방
2. (스터디룸의 제한 인원 – target 의 인원 수)가 작은 방
3. 스터디룸의 방 번호가 작은 방

정렬 후의 possibleStudyRoom 의 0 번 인덱스 원소는 target 이 옮길 수 있는 스터디룸 중 가장 우선 순위가 높은 것입니다. possibleStudyRoom[0]의 튜플 중 첫 번째 원소를 int 타입의 destReserveId 에 넣고, 두 번째 원소를 int 타입의 destRoomId 에 넣습니다. 그리고 data[destRoomId][timeIndex]에 target 의 예약 번호를 넣습니다.

만약 destReserveId 가 0 이라면, dest 는 어떤 예약도 있지 않는 빈 자리이므로 true 를 리턴합니다. 0 이 아니라면 다른 예약이 존재하는 자리이므로, go(...) 메소드 재귀 호출 후 결과를 반환합니다. 이때 재귀 호출하는 go()의 인자로는 dest 의 정보를 넣습니다. 특히 4 번째 인자인 예약 인원 수는 File 클래스의 getReserNum() 메소드를 이용해 받아옵니다.



<go(…) 메소드의 순서도>

Side-Effect

첫 번째 인자인 data 를 vector<vector<string>>&로 받게 됩니다. 따라서 해당 메소드 내부에서 data 의 값이 변경되면, 이 함수를 호출한 곳의 인자도 데이터도 변경되게 됩니다.

1.3. convertTimeToIndex

인터페이스

```
int Optimize::convertTimeToIndex(string &time)
```

메소드가 하는 일

시간(time)을 File 클래스에서 예약 날짜의 스터디룸 현황을 담은 vector<vector<string>> data 의 행 인덱스로 변환합니다.

인자

1. string time: 인덱스로 변환시킬 시간으로, "시간"의 표준형식을 따르며, 시간보정이 적용되어 있음을 보장합니다.
   표준형식 : <숫자 2 개>:<숫자 2 개>형태의 문자열
   시간보정 : 기존의 시간으로부터 가장 가까운 30 분 단위의 시간으로 변환합니다.
   따라서 인자의 형식은 "<숫자 2 개>:00" 또는 "<숫자 2 개>:30" 입니다.

반환

인자로 주어진 시간을 인덱스로 변환시킨 결과가 반환됩니다. 이때 반환 결과는 0 이상, 21 이하의 int 타입의 숫자입니다.

구현

09:00 는 0, 10:00 는 2, 11:00 는 4, ... ,19:00 는 20 으로 변환되어야 합니다.
time 의 앞 2 자리(시, hour)를 int 형으로 변환 후, 9 를 뺀 후 2 를 곱한 값을 int 타입의 ret 변수에 저장합니다.
time 의 인덱스 3(분, minute)에 해당하는 숫자가 '0'이라면 00 분이므로 ret 을 반환하고, 아니라면 30 분 단위이므로 ret 에 1 을 더한 후 반환합니다.

2. Book: 최적화 알고리즘을 도입한 예약기능

스터디룸 예약을 수행하기 위해서 별도의 클래스를 사용합니다. 클래스 이름은 "Book"이며, 구현 파일(.cpp)와 헤더 파일(.h)을 생성합니다. 외부 클래스에서 해당 클래스의 메소드를 사용할 때는 사용 클래스 상단에 "#include "Book.h""를 선언한 뒤 public 메소드를 사용할 수 있습니다.

1 차에서 구현했던 기능에 "Optimize"를 이용한 최적화 기능을 추가하였습니다.

클래스 헤더파일(Book.h)

해당 클래스 헤더는 1 개의 public 메소드와 7 개의 private 메소드, 1 개의 생성자를 갖습니다. 메소드의 인터페이스 및 세부 정보는 아래에서 설명합니다. 1 차 구현과 큰 변경사항이 존재하지 않는 메소드는 설명을 생략합니다.

2.1 Book 생성자

인터페이스(public)

```
Book(string sdate, string sRoomNumber, string sUseStartTime, string sUseEndTime,
string userId, string sPeopleNum)
```

생성자 내부에서 하는 일

- 인자로 전달받은 각각의 데이터 요소에 대하여 Book 클래스의 private 메소드 validTime(), validDate(), validRoomNumber(), validPeopleNumber() 메소드를 호출하여 올바른 의미규칙을 가진 인자인지 검증합니다.
- 올바른 의미규칙을 가졌으며, 예약 인원수가 예약하고자 하는 방의 인원수보다 작은 경우에는 optimize pubic 메소드를 호출하여 해당 예약 정보에 대한 최적화를 수행하고 반환한 예약 날짜의 스터디룸 예약 상태를 vector<vector<string>>타입의 bookFileData 변수에 저장합니다.
- Optimize 로 인해 바뀐 다른 사용자의 데이터를 File 클래스의 메소드를 이용해서 변경 내용을 반영해줍니다.

- 예약하고자하는 사용자의 사용자 번호에 해당하는 예약 내역 정보를 불러와 vector<vector<string>> 타입의 userData 변수에 저장합니다.

인자

표준형식을 따르는 6 개의 인자를 전달 받으며, 각 인자의 "의미 규칙"에 대한 검증은 되어있지 않은 상태입니다.

1. string sdate : 예약하고자 하는 날짜를 뜻합니다. "날짜"의 표준형식을 따릅니다.
   표준 형식: <4 자리 숫자>-<2 자리 숫자>-<2 자리 숫자>로 이루어진 문자열
   이때, 앞 4 자리 숫자는 연(year), 중간 2 자리 숫자는 월(month), 끝 2 자리 숫자는 일(day)로 해석합니다.
   예를 들어, "2022-10-09"는 표준 형식으로 나타낸 날짜이며, 2022 년 10 월 09 일을 의미합니다.
2. string sRoomNumber : 예약하고자하는 방번호를 뜻합니다. "방번호"의 표준형식을 따릅니다.
   표준형식 : 숫자로만 이루어진 길이가 1 인 문자열
3. string sUseStartTime : 예약하고자 하는 시작 시간을 뜻합니다. "시간"의 표준형식을 따르며, 시간 보정이 적용되어 있음을 보장합니다.
   표준형식 : <숫자 2 개>:<숫자 2 개>형태의 문자열
   시간보정 : 기존의 시간으로부터 가장 가까운 30 분 단위의 시간으로 변환합니다.
   따라서 인자의 형식은 "<숫자 2 개>:00" 또는 "<숫자 2 개>:30" 입니다.
4. string sUseEndTime : 예약하고자 하는 종료 시간을 뜻합니다. sUseStartTime 와 같이 "시간"의 표준 형식을 따르며, 시간 보정이 적용되어 있음을 보장합니다.
5. string userId : 예약하고자 하는 사용자의 사용자 번호를 뜻합니다. "사용자 번호"의 표준형식을 따릅니다.
   표준 형식: 숫자로만 이루어진 길이 1 이상, 4 이하의 문자열
6. string sPeopleNum : 예약하고자 하는 인원 수를 뜻합니다. "예약 인원"의 표준형식을 따릅니다.
   표준 형식: <숫자 1 개> 또는 <숫자 2 개>형태의 문자열

반환

생성자는 별도의 반환값을 가지지 않습니다.

구현

1. File::getBooking 메소드를 이용해서 예약 인원이 예약할려는 방의 예약 인원보다 큰지 확인합니다.

2. Optimize 메소드를 실행하기 전에 예약하려는 날의 data 예약 정보를 vector<vector<string>> 타입의 beforedata 변수에 담습니다.
3. Book 클래스의 private 변수이며 vector<vector<string>> 타입인 bookFileData 에 optimize 메소드의 반환값을 받습니다.
4. 기본값이 false 인 flag 를 선언해줍니다.
5. 예약 가능 여부를 판단하기 위해 book 의 checkReservation 메소드를 실행합니다.
6. 위 결과가 true 이면 예약 할려는 방과 해당하는 시간을 의미하는 bookFileData 의 값을 0 으로 변경해줍니다.
7. bookFileData 와 beforedate 를 비교하면서 일치하지 않는 부분의 값을 vector<string> 타입의 changed_id 변수에 담고 flag 값을 true 로 바꿔줍니다.
8. Flag 가 true 인 경우 vector<vector<string>> 타입의 changed 변수를 선언하고 bookFileData 의 모든 값에 대해 다음의 내용을 실행합니다. ('열'을 먼저 탐색하고 그 다음 '행'을 탐색)

   changed_id 에 bookFileData[i][j] 값이 있는 경우 changed 에 다음의 값을 추가합니다.
   아래의 timeIndex 는 bookFileData 에서 해당하는 '열' 번호를 말합니다.
   아래의 날짜는 표준형식의 날짜 형식에서 '-'를 뺀 숫자로 이뤄진 string 을 말합니다.

   - changed 의 크기가 0 인 경우
     - changed 에 예약번호, 날짜, 시작 timeIndex, 종료 timeIndex, 방 번호를 순서대로 저장하고 있는 vector<string> 타입의 변수를 추가해줍니다.
   - 그 외의 경우
     - changed 에 같은 예약 번호가 있는,경우 해당 예약 번호가 들어간 부분에 대해
       - changed 의 예약 종료 timeIndex 값이 추가하려는 시작 timeIndex 값이며, 같은 방인 경우 종료 timeIndex 만 추가하려는 종료 timeIndex 값으로 변경
     - 시작 timeIndex, 종료 timeIndex, 방 번호를 순서대로 push_back
   - changed 에 예약번호, 날짜, 시작 timeIndex, 종료 timeIndex, 방 번호를 순서대로 저장하고 있는 vector<string> 타입의 변수를 추가해줍니다.

9. changed 에 들어가 있는 값들을 표준형식으로 바꿔줍니다.
10. File 클래스의 메소드 changedStudyRoom 에 changed 의 값을 인자로 넣어서 변경된 예약 내역을 파일에 반영해줍니다.

2.2 validPeopleNumber

인터페이스

```
void Book::validPeopleNumber()
```

메소드가 하는 일

Book 클래스의 멤버변수 peopleNum 에 대하여 "예약 인원" 데이터 요소의 의미 규칙을 만족하는지 검증합니다.

"예약 인원"의 의미 규칙은 다음과 같습니다.

- 예약 인원은 0 명이 될 수 없습니다.
- 스터디룸이 허용하는 최대 인원수보다 더 많은 인원은 예약인원이 될 수 없습니다.

인자

이 메소드는 별도의 인자를 전달받지 않습니다.

반환

이 메소드는 별도의 반환값을 가지지 않습니다.

2.3 updateBookFile

인터페이스

```
void Book::updateBookFile()
```

메소드가 하는 일

- 예약으로 인한 파일들의 변경사항을 업데이트 합니다.
- 여기서 변경되는 파일들은 user/[userid].txt, resource/resernum.txt, resource/meta.txt, book/[YYYYMMDD].txt 파일들입니다.

인자

이 메소드는 별도의 인자를 가지지 않습니다.

반환

이 메소드는 별도의 반환값을 가지지 않습니다.

구현

File 클래스의 메소드 setUserData(userId, userData), setBooking(sOriginDate, bookFileData), addReserNum(sPeopleNum, userId) 를 호출합니다.

3. List: 최적화 알고리즘을 도입한 예약기능

스터디룸 예약을 수행하기 위해서 별도의 클래스를 사용합니다. 클래스 이름은 "List"이며, 구현 파일(.cpp)와 헤더 파일(.h)을 생성합니다. 외부 클래스에서 해당 클래스의 메소드를 사용할 때는 사용 클래스 상단에 "#include "Book.h""를 선언한 뒤 public 메소드를 사용할 수 있습니다.
1 차에서 구현했던 기능에 "Optimize"를 이용한 최적화 기능을 도입하였습니다.

3.1 excuteList

인터페이스

```
List::excuteList(string peopleNum, string date)
```

메소드가 하는 일

- 인자로 전달받은 각각의 데이터 요소에 대하여 Book 클래스의 private 메소드 validDate(), validPeopleNumber() 메소드를 호출하여 올바른 의미규칙을 가진 인자인지를 검증합니다.
- 올바른 의미규칙을 가졌으며, PeopleNum 이 해당 방의 예약 가능 인원보다 작다면, optimize pubic 메소드를 호출하여 예약 가능한 자리인지 검사를 합니다. 예약이 가능하다면 "가능", 불가능하다면 "X"를 출력합니다.
- 별도의 반환 값 없이, 예약 가능 여부를 출력합니다.

인자

표준형식을 따르는 2 개의 인자를 전달 받으며, 각 인자의 "의미 규칙"에 대한 검증은 되어있지 않은 상태입니다.

1. string peopleNum : 예약하고자 하는 인원 수를 뜻하며, "예약인원"의 표준형식을 따릅니다.
   표준 형식: <숫자 1 개> 또는 <숫자 2 개>형태의 문자열
2. string sdate : 예약하고자하는 날짜를 뜻합니다. "날짜"의 표준형식을 따릅니다.
   표준 형식: <4 자리 숫자>-<2 자리 숫자>-<2 자리 숫자>로 이루어진 문자열
   이때, 앞 4 자리 숫자는 연(year), 중간 2 자리 숫자는 월(month), 끝 2 자리 숫자는 일(day)로 해석합니다.

반환

이 메소드는 별도의 반환값을 가지지 않습니다.

구현

1. VaildData() , vaildPeopleNumber() 메소드를 이용해 받은 인자의 의미규칙을 검증합니다.
2. File:getBooking()메소드를 통해서 date 에 맞는 fileData 를 읽어옵니다. 데이터는 fileData 변수에 저장합니다.
3. 사용자가 보기 편하도록 날짜와 시간, 사용가능한 인원수를 출력합니다.
4. 방의 예약 가능 여부는 Optimize::optimize() 메서드를이용해서 요청한 스터디룸이 최적화 가능할때 '가능' 출력을 합니다. 최적화가 불가능 하면 'X'라는 출력을 합니다.

4. File : 예약 인원과 관련된 파일 Read/Write 수정, 추가/변경된 파일

파일을 read/write 하기 위해 별도의 클래스를 사용합니다. 클래스 이름은 "File"이며, 구현 파일(.cpp)와 헤더 파일(.h)을 생성합니다. 외부 클래스에서 해당 클래스의 메소드를 사용할 때는 사용 클래스 상단에 "#include "File.h""를 선언한 뒤 public 메소드를 사용할 수 있습니다.

1 차에서 이미 구현했던 기능에 예약 인원과 관련된 기능을 추가 및 수정했습니다.

추가된 파일(rootPath/resource/resernum.txt)
해당 파일은 새롭게 추가된 파일로 예약 번호에 해당하는 예약 인원과 예약자 id 를 공백(탭)을 기준으로 저장하고 있습니다.
N 행 첫번째 요소는 예약 번호 N-1 번의 예약 인원을 담고있으며, N 행 두번째 요소는 예약 번호 N-1 번의 예약자 id 를 담고 있습니다.
여기서 '행'은 1 번째부터 시작합니다. 그러므로 resernum 의 첫번째 행은 실제 프로그램의 동작에서 사용하지 않는 값입니다.(예약 번호 0 번은 없으므로)

변경된 파일(rootPath/resource/meta.txt)
해당 파일은 기존의 파일의 끝에 각 방에 해당하는 제한 인원과 개행이 추가됐습니다.
여기서 파일의 끝이란 std::ios::end 를 의미합니다.
각 방의 제한인원은 공백(tab)을 기준으로 구분됩니다.
추가된 라인의 N 번째 요소는 N 번방의 제한 인원을 의미합니다.

변경된 파일(rootPath/user/[userid.txt])
해당 파일은 중간에 방이 바뀌는 경우를 저장하도록 파일의 형식이 바뀌었습니다.
기존에는 "<예약 번호> <예약 날짜> <예약 시작 시간> <예약 종료 시간> <방번호>" 였지만, 현재는 "<예약 번호> <예약 날짜> (<예약 시작 시간> <예약 종료 시간> <방번호>)+"입니다.

여기서 <> 은 형식을 표현한 것이고, '(', ')','+' 는 파일에 쓰이지 않으며 이해를 돕기 위해 정규식으로 표현한 것입니다.

클래스 헤더 파일(File.h)

해당 클래스 헤더는 1 개의 private 메소드, 13 개의 public static 메소드와 1 개의 static 변수를 갖습니다. Static 변수의 이름은 rootPath 로 파일의 시작 위치를 저장하고 있습니다. 또한 외부 클래스에서 메소드의 인터페이스 및 세부 정보는 아래에서 설명합니다.

1 차 구현과 큰 변경사항이 존재하지 않는 메소드는 설명을 생략합니다.

4.1 getReserNum

인터페이스

String File::getReserNum(string reserveId)

메소드가 하는 일

인자로 전달받은 예약 번호를 가진 예약의 예약 인원을 반환합니다.

인자

표준형식을 따르는 1 개의 인자를 전달 받으며, 각 인자의 "의미 규칙"에 대한 검증은 되어있는 상태입니다.

1. string reserveId : 예약 인원을 조회할 예약 번호입니다. "예약 번호"의 표준형식을 따릅니다.
   표준 형식: <숫자 1 개> 형태의 문자열

반환

해당 예약 번호의 예약 인원을 반환

구현

1. rootPath + resource/resernum.txt 파일을 읽어서 각 요소를 2 중 벡터 vec 에 저장합니다.
   - 파일의 n 번째 행 n 번째 요소는 vec[n-1][n-1]에 저장됩니다.
2. 파일의 n 번째 라인의 첫번째 요소(예약 인원)를 리턴합니다.

4.2 getRoomCapacity

인터페이스

String File::getRoomCapacity(string roomNum)

메소드가 하는 일

rootPath/resource/meta.txt 를 읽고 인자로 전달받은 방 번호의 최대 수용 인원에 해당하는 값을 리턴합니다.

인자

표준형식을 따르는 1 개의 인자를 전달 받으며, 각 인자의 "의미 규칙"에 대한 검증은 되어있는 상태입니다.

1. string roomNum : 방 최대 수용 인원을 조회할 방 번호입니다. "방번호"의 표준형식을 따릅니다.
   표준 형식: <숫자 1 개> 형태의 문자열

반환

해당 예약 번호의 예약 인원을 반환합니다.

구현

File 클래스의 메소드 getMetaData()의 리턴 값에서 방 번호 roomNum 의 제한 인원을 의미하는 곳을 반환합니다.

4.3 addReserNum

인터페이스

void File::addReserNum(string num, string userId)

메소드가 하는 일

인자로 전달받은 예약 인원과 유저 아이디를 resernum.txt 파일에 저장합니다.

인자

표준형식을 따르는 2 개의 인자를 전달 받으며, 각 인자의 "의미 규칙"에 대한 검증은 되어있는 상태입니다.

1. string num : 저장할 예약 예약 인원을 의미하며 예약하는 방의 제한 인원 이하인 정수입니다.
2. string userId : 예약을 한 유저의 id 를 의미합니다.

반환

이 메소드는 별도의 반환값을 가지지 않습니다.

구현

rootpath + resource/resernum.txt 파일의 끝부분(std::ios::end)에 num 과 userId 를 공백(탭)을 두고 write 하고 개행을 추가적으로 write 합니다.

4.4 changeStudyRoom

인터페이스

void File::changeStudyRoom(vector<string> changeDate)

메소드가 하는 일

유저 파일에서 인자에 저장되어있는 예약 번호와 일치하는 예약을 인자의 내용으로 바꿉니다.

인자

표준형식을 따르는 1 개의 인자를 전달 받으며, 각 인자의 "의미 규칙"에 대한 검증은 되어있는 상태입니다.

1. vector<string> reserveId : 예약번호 하나에 해당하는 정보를 담고 있는 인자입니다. Ex) 예약 번호, 날짜, 시작 시간, 종료 시간, 방 번호가 인자입니다.

이 순서는 [userId].txt 에 저장되는 인자의 순서와 같습니다.

즉 '(시작 시간, 종료 시간, 방 번호)+'입니다. ('(', ')', '+'는 앞에서 말한 정규 표현식을 의미합니다.)

반환

이 메소드는 별도의 반환값을 가지지 않습니다.

구현

1. 해당 예약을 가진 유저 id 를 메소드 내에서만 유효한 지역변수 user_id 에 저장합니다.
2. 1 에서 구한 유저 id 파일을 메소드 내에서만 유효한 지역 변수 data 에 저장합니다.
3. data 에서 인자로 들어온 예약 번호(changeData 의 첫번째 요소)와 일치하는 부분을 찾고 이 메소드 내에서만 유효한 지역 변수 target 에 그 값을 저장합니다.
4. data[target]에 changeDate 를 대입합니다.
5. File 클래스의 메소드 setUserData 에 user_Id 와 data 를 인자로 넣어서 파일을 변경해줍니다.

4.5 getId

인터페이스

String File::getId(string reserveId)

메소드가 하는 일

- 인자로 전달받은 예약 번호를 가진 사용자의 id 를 반환합니다.
- Private 로 선언된 메소드로 다른 file 메소드에서 사용됩니다.

인자

표준형식을 따르는 1 개의 인자를 전달 받으며, 각 인자의 "의미 규칙"에 대한 검증은 되어있는 상태입니다.

1. string reserveId : 예약 인원을 조회할 예약 번호입니다. "예약 번호"의 표준형식을 따릅니다.
   표준 형식: <숫자 1 개> 형태의 문자열

반환

해당 예약 번호의 사용자 id 를 반환

구현

1. rootPath + resource/resernum.txt 파일을 읽어서 각 요소를 2 중 벡터 vec 에 저장합니다. - 파일의 n 번째 행 n 번째 요소는 vec[n-1][n-1]에 저장됩니다.
2. 파일의 n 번째 라인의 두번째 요소(예약자 id)를 리턴합니다.

4.6 Start

인터페이스

```
void File::start()
```

메소드가 하는 일

- 프로그램이 시작할 때 필요한 기본적인 파일이 있는지 확인합니다.
- 파일 실행에 필요한 디렉토리들이 없다면 디렉토리를 생성합니다.
- 기존에 존재하는 meta.txt 와 resernum.txt 가 형식이 맞는지 검증합니다.

인자

이 메소드는 별도의 인자를 가지지 않습니다.

반환

이 메소드는 별도의 반환값을 가지지 않습니다.

구현

1. 프로그램 시작 시, 각 디렉토리를 생성하고 meta.txt 파일을 이동합니다.
2. ./resource에 resernum.txt이 없다면 resernum.txt 파일을 만들고 초기값을 넣어줍니다.
3. meta.txt 파일의 존재 여부를 확인하고 파일에 저장된 요소의 개수(정상이라면 13)를 확인합니다.
4. meta.txt 파일이 올바른 형식(이름/전화번호/회원 수/예약 수/방 수용 인원)을 만족하는지 확인합니다.