K-CUBE 예약 및 조회 프로그램

설계 문서 수정 1 판

[A 반 분반 06 팀]

변경 내역

[1.개요] Notificaion.txt 파일 불러오기, 출력 과정 추가

[6-3-3] 삭제된 학번 행 출력 과정 삭제

[6-3-4] 공지 내용 수정 후 공지 수정 확인 Y/N 입력 과정 추가

[6-3-4] NotificationFile -> 공지 파일(Notification.txt) 내용 수정

* 노란색 하이라이트: 수정, 추가 내역, 취소선: 삭제 내역

202011293 박성근

202011360 장현준

202011381 최우헌

202011291 박경범

202211388 최정우

202211348 이창근

목차

1. 개요

(Room(n=1,2,3)_Information.txt 는 Room1_Information.txt, Room2_Information.txt, Room3_Information.txt 을 모두 포함하는 의미로 사용됩니다.)

- **K-CUBE 예약 및 조회 프로그램**은 학번, 이름, 비밀번호를 등록하여 신규 사용자가 회원가입 후 로그인하면 K-CUBE 예약 서비스를 이용할 수 있습니다. 기존에 등록된 사용자는 로그인을 통해 K-CUBE 예약 서비스를 이용할 수 있습니다. K-CUBE 예약 시스템 내에는 예약 신청, 예약 취소, 예약 조회 기능이 존재합니다. 관리자 로그인을 실행한 경우, 학생 정보를 수정할 수 있으며, 스터디룸 수정, 공지 출력을 할 수 있습니다.

- 프로그램은 다음과 같은 클래스로 구성됩니다:

  TestMain.java : **메인 메뉴**를 구성하는 클래스

  MemberInfo.java : **회원가입**과 **로그인**을 수행하는 클래스

  Reservation.java : **예약 메뉴**를 구성하는 클래스

  ManagerInfo.java : **관리자 메뉴**를 구성하는 클래스

- 프로그램에서 사용되는 텍스트 파일은 다음과 같습니다:

  StudentInfoFile.txt : **학생 정보**를 담는 파일

  Room1_Information.txt : 공학관의 예약 정보를 담는 파일

  Room2_Information.txt : **생명과학관의 예약 정보**를 담는 파일

  Room3_Information.txt : **상허연구관의 예약 정보**를 담는 파일

  Notification.txt : **공지 사항**을 저장하는 파일"

  .

- 프로그램 실행 후 TestMain.java 가 실행되고, StudentInfoFile.txt 가 프로그램으로 불러와집니다. 메인 메뉴 출력 전 Notification.txt 를 불러오고 파일의 문자열을 출력해서 공지사항을 출력합니다. 사용자가 회원가입 또는 로그인 기능을 선택하면 MemberInfo.java 클래스가 실행됩니다. 회원가입 시 StudentInfoFile.txt 가 수정되며, 로그인 시도 시 StudentInfoFile.txt 내의 학번과 비밀번호와 비교하여 로그인이 수행됩니다.

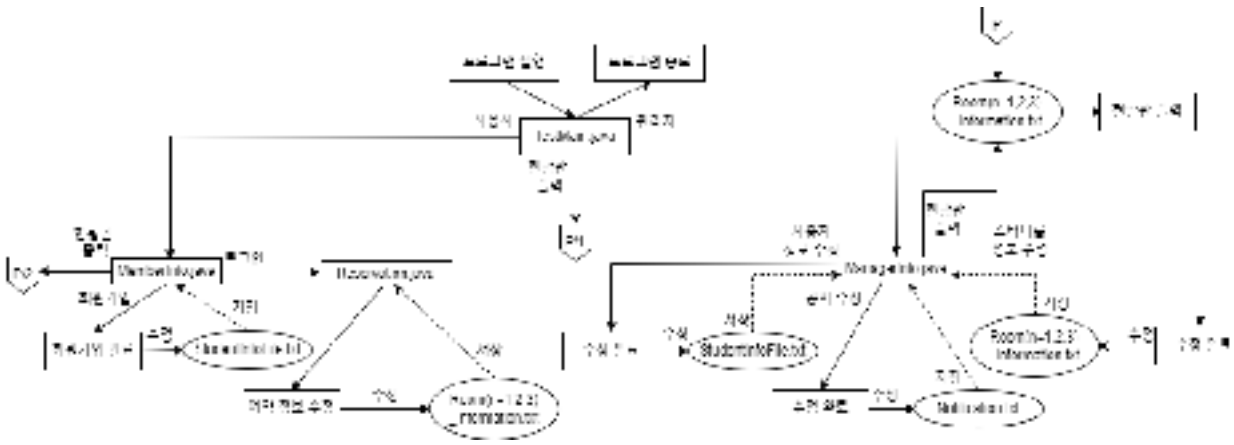
  사용자나 관리자가 건물을 선택하면 Room(n=1,2,3)_Information.txt 중 하나의 파일이 프로그램으로 불러옵니다. 로그인 후 사용자가 예약 메뉴에 접근하면 Reservation.java 클래스가 실행됩니다. 예약, 예약 취소, 예약 조회 작업이 ReservationInfo.txt 에 반영되고, 해당 작업에 따라 Room(n=1,2,3)_Information.txt 가 수정됩니다.

  관리자 메뉴에 접근하면 ManagerInfo.java 가 실행됩니다. 관리자가 스터디룸을 수정하면 선택된 건물의 정보가 Room(n=1,2,3)_Information.txt 에 수정되어 저장됩니다. 또한, 관리자가 공지 사항을 수정하면 해당 내용이 Notification.txt 에 반영되어 저장됩니다.

현황판 출력 기능 실행 시, 선택된 건물의 Room(n=1,2,3)_Information.txt 파일에서 정보를 가져와서 현황판을 출력합니다.

2. 순서도

- 프로그램 구성 순서도는 4 쪽의 '순서도' 그림에 해당합니다.
- 클래스, 파일의 명칭은 [1. 개요] 절에서 의미하는 내용과 동일합니다.
- 순서도는 **프로그램 시작, 종료, 프로그램 구성, 클래스 간의 관계, 파일 사용 시기**를 포함하고 있습니다.



<순서도>

3. 메인 메뉴 선택

수정 내역은 다음과 같습니다.

- 입력 받은 문자열을 정수로 변환한 변수인 choice 가 5 일 때의 분기를 추가합니다. 3 일 때의 분기를 '관리자 로그인'으로 변경하고, 4 일 때의 분기를 '현황판 출력'으로 변경하며, 5 일 때의 분기를 '종료'로 변경합니다.
- 메인 메뉴 선택 과정에서 사용된 while 문 내에서 **MemberInfo 객체**를 생성한 후, 해당 객체의 메소드로 **'회원가입'**과 **'로그인'**을 진행합니다.

4. 현황판 출력

수정 내역은 다음과 같습니다.

- 건물의 예약 정보를 담기 위해 2 차원 배열을 사용하지 않고, ArrayList 자료구조를 사용합니다. ArrayList 자료구조를 사용하는 변수의 이름은 build1, build2, build, tables 입니다.

  build1, build2, build3 의 자료형은 **ArrayList<ArrayList<String>>형식**이며 각각 **공학관, 생명과학관, 상허연구관의 예약 내역 정보**를 가지고 있습니다. 각 변수의 정보 저장 형식은 다음과 같습니다.

호실	한도 인원	1교시	2교시	3교시	4교시	5교시	6교시
1	4	202011291	0	0	X	0	0
2	9	0	0	X	0	0	X
3	3	202011394	0	201933485	X	0	0
4	6	202010332	0	0	X	X	0

build2, build3도 위 형식과 동일

<build1에 호실이 4개 있을 경우>

tables 의 자료형은 **ArrayList<ArrayList<ArrayList<String>>>** 형식이며, **build1, build2, build3 의 정보를 모두 가지고 있습니다**. 정보 저장 형태는 다음과 예시와 같습니다.

호실	한도 인원	1교시	2교시	3교시	4교시	5교시	6교시
1	4	X	X	0	X	0	X
2	5	201711383	0	X	201510293	0	0
3	2	0	0	0	X	0	0
4	8	0	0	0	0	0	0

<tables 에서 **build2** 의 정보를 가져올 경우>

위에서 설명한 변수들은 모두 메인 메뉴의 클래스인 TestMain.java 뿐만 아니라 예약 메뉴 클래스인 Reservation.java, 관리자 메뉴의 클래스인 ManagerInfo.java 에서도 사용됩니다. 메인 메뉴 부분에서는 main 안에 생성하지

않고 그 밖인 TestMain 클래스 내에서 생성하고 static 변수로 생성합니다. 다른 클래스에서는 static 으로 생성하지 않습니다. class 안에만 생성하면 됩니다.

아래에서 설명할 함수들도 위 변수와 동일하게 세 클래스에서 모두 사용됩니다. 생성 방식은 변수와 동일합니다.

- **Recent (int buildnum)**

이 함수는 입력 받은 정수에 따라 해당 건물의 현황판을 출력합니다.

[입력 매개변수 1]

buildnum 이라는 정수형 자료를 입력 받습니다. buildnum 은 '건물 번호'를 의미하며 1,2,3 의 정수만 입력 받습니다.

[출력]

처음 받은 buildnum 의 값이 **1 일 경우 공학관**, **2 일 경우 생명공학관**, **3 일 경우 상허연구관**의 **현황판**을 출력합니다.

[동작]

1. buildnum 값에 따라 공학관, 생명공학관, 상허연구관 중 하나의 현황판을 우선 출력합니다.
2. building 변수를 생성하고, buildnum 에서 1 을 뺀 값을 대입합니다. building 변수는 건물 인덱스를 판단하기 위해 사용됩니다
3. tables 에는 세 건물의 예약 정보가 들어있고, 인덱스는 0 부터 2 까지입니다. 따라서 building 변수에 1 을 뺀 값을 대입하여 해당 건물의 인덱스를 얻습니다.
4. 현황판 출력을 위해 이중 반복문을 사용합니다.
5. 외부 반복문은 존재하는 호실의 수만큼 수행됩니다. 외부 반복문의 카운터 변수 i 는 0 부터 tables.get(building).size-1 까지 증가합니다.
6. 내부 반복문은 8 회 수행됩니다. 내부 반복문의 카운터 변수 j 는 0 부터 7 까지 증가합니다.

7. **j 가 2 이상일 때, "X"일 때의 조건이 추가됩니다. "X"는 해당 호실의 그 시간대에 예약이 불가능한 상태임을 의미합니다. 이 때 "F"를 출력합니다.**
8. tables 의 특정 행과 열 정보에 접근하기 위해 tables.get(building).get(i).get(j)을 반복문 내에서 사용합니다.
9. 반복문이 종료된 후, "F"는 예약이 불가능한 상태를 나타내는 문자열로 추가해 출력합니다.

위 과정을 모두 수행한 이후 Recent 함수는 종료됩니다.

[예외처리]

이 함수에서는 특별한 예외 처리가 필요하지 않습니다.

- **file_check (String filename)**

이 함수는 주어진 파일의 형식을 체크하고 파일의 요소를 변수에 대입하여 반환하는 함수입니다. 반환형과 동일한 자료형인 ArrayList<ArrayList<String>>을 반환합니다.

[입력 매개변수 1]

filename 이라는 문자열 자료를 입력받습니다. filename 은 '파일 이름'를 의미합니다.

예를 들어, build1 이라면 1 번 건물(공학관)의 파일 이름을 매개변수로 넣어서 수행합니다.

[반환]

파일요소를 담은 ArrayList<ArrayList<String>>자료형 변수를 반환합니다.

[동작]

1. 함수가 시작되면 **반환형과 동일한 tempbuild 변수를 생성**합니다.

2. 파일을 읽어오는 과정은 기존의 방법과 동일합니다.
3. 파일을 읽어오면서 형식을 체크합니다.
4. 파일 형식에 이상이 없는 경우, temp 라는 ArrayList<String> 변수를 생성합니다.
5. tokens 배열에 저장된 문자열을 temp 에 추가합니다.
6. temp 는 파일 형식과 동일하게 인덱스 0 에 호실, 1 에 한도 인원, 2 부터 7 까지 각 교시의 예약 정보를 갖고 있습니다.
7. temp 를 tempbuild 에 추가합니다.
8. check 를 1 증가시키고, while 반복문의 조건 확인으로 돌아갑니다.
9. 반복문이 종료된 후, tempbuild 를 반환합니다

[예외처리]

파일 형식 체크 과정에서 기존 코드에 변화가 있습니다.

- check 변수는 호실이 체크 될 때마다 1 씩 늘어납니다. 기존에는 호실이 8 로 고정되었기 때문에 check 가 8 이 된다면 파일 형식 오류로 프로그램을 종료했습니다. 프로그램 수정 후 호실이 0 개 이상 9 개 이하로 가변적으로 변했습니다. check 의 값이 최대 호실 개수인 9 임에도 읽어야 할 줄이 더 있을 경우 프로그램을 강제 종료합니다.

- 한도 인원 체크에서 변경된 점은 호실에 관계없이 한도 인원이 2 이상 9 이하의 범위 내에 속하지 않으면 파일 형식 오류로 간주합니다. (tokens[1]이 한도 인원을 나타내며, 2 미만 또는 9 초과일 경우 오류로 처리합니다.)

- 현황판 출력 방법

1. switch 문의 case 3 에 코드를 길게 작성해야 했지만, 이를 함수화하여 switch 문 내의 코드를 간결하게 유지했습니다.
2. build1, build2, build3 에 각 파일명에 맞게 file_check 함수를 실행하고 그 결과를 변수에 저장합니다.
3. 이후, tables 에 각 건물들을 차례대로 추가합니다.

4. 각 건물 번호에 해당하는 현황판을 출력하기 위해 Recent 함수를 건물 번호를 인자로 넣어 세 번 호출합니다.

5. 현황판 출력 메뉴를 실행할 때마다 tables 에 새로운 요소가 추가되므로, case 3 를 종료하기 전에 tables 을 초기화(clear)합니다

예약 메뉴와 관리자 메뉴에서 현황판 출력 방법은 해당 항목에서 설명합니다.

5. 예약 메뉴

수정 내역은 다음과 같습니다.

5-1. 변수 및 함수

- **'4. 현황판 출력' 항목**과 동일하게 build1, 2, 3, tables 변수를 생성하고 기존의 Recent() 함수를 수정된 버전으로 적용하고 file_check 함수를 생성합니다. 변수 및 함수 생성 방식은 '**4. 현황판 출력'항목**에 나와 있습니다.
- builds 라는 String 배열 변수를 생성합니다. 들어갈 문자열은 "공학관", "생명공학관", "상허연구관"으로 총 세 개입니다. 해당 변수는 건물명 출력을 위해 생성한 변수입니다.

5-2. Reservation 생성자

- 기존에는 사용자의 아이디를 받아 Reservation 생성자를 실행하면 파일 형식 체크와 파일의 내용을 배열에 저장하는 과정을 실행하는 코드가 모두 들어 있었으나 이 과정이 함수화 되어 코드가 간소화되었습니다.

1. build1, build2, build3 에 대해 파일 형식 체크를 수행하기 위해 해당 파일명에 맞게 file_check 함수를 실행하고 그 결과를 변수에 저장합니다.
2. 저장된 내용을 배열에 추가하기 위해 tables 에 각각 add 합니다.
3. 예약 메뉴를 실행합니다.
4. 예약 메뉴가 종료된 후, tables 를 초기화합니다 (clear).

5. 생성자가 종료됩니다.

5-3. 예약 신청

[추가 변수]

buildnum: 건물 번호 저장용, 정수형

rooms: 해당 건물 호실 개수 저장용, 정수형

filename: 파일명 저장용, 문자열

[임시 사용 변수]

choose1: 건물 선택 시 사용되는 임시 변수, 기존에 호실 인덱스로 사용하는 변수.

건물 선택 시에만 사용되고 호실 선택 시 기존 방식대로 호실 인덱스로 사용됩니다.

[동작]

1. 함수가 시작되면 건물 선택지를 출력하고 건물을 입력 받습니다. 건물은 공학관, 생명공학관, 상허연구관 세 개이고 각각에는 1 번부터 3 번까지 번호를 부여합니다. 입력은 문자열로 입력 받습니다.
2. 입력 받은 값을 choose1 에 정수로 변환한 후 저장합니다.
3. choose1 이 1 이상 3 이하 값이라면 buildnum 에 건물 번호를 저장합니다.
4. tables 의 해당 건물 인덱스(buildnum-1)의 size 는 호실 개수를 의미하니 이 값을 rooms 에 저장합니다.
5. 해당 건물 번호의 현황판을 출력하는 Recent 함수를 실행합니다. 매개 변수로 buildnum 을 넣어줍니다.
6. 호실, 교시, 인원은 기존의 입력 방식과 동일합니다.
7. 해당 호실의 예약 가능 유무를 확인합니다.
8. 삼중 반복문을 이용하여 현재 사용자가 예약한 내역(1 일 1 예약 초과)이 있는지 확인합니다.

9. 해당 호실 다른 예약 내역 존재, 해당 호실 적정 인원 불충족 여부를 확인합니다.

- 비교 방식은 기존과 동일하며 배열을 썼던 부분을 tables 로 수정합니다.

10. 예약 조건에 만족하면 예약 의사를 다시 묻습니다.

11. "Y", "N" 입력에 따라 해당 기능 수행 후 예약 메뉴로 돌아갑니다.

- "Y" 입력 시 예약한 건물명(builds 배열 사용), 호실, 예약 인원, 교시를 출력하고 tables 의 해당 인덱스의 원래 있던 문자열을 지우고 다시 해당 인덱스에 사용자 아이디를 추가합니다.

  filename 에 해당 건물의 파일명을 저장합니다. ex) "src/StudyRoom/Room" + buildnum + "_Information.txt"

  file_change 함수에 매개변수로 filename 과 talbes 의 현재 건물 인덱스의 요소를 넣고 함수를 실행해서 파일 정보를 수정합니다.(file_change 함수의 매개 변수 종류가 변화한 부분은 **5-5. 파일 수정**에서 설명합니다).

- "N" 입력 시 다시 예약 메뉴로 돌아갑니다.

[예외 처리]

- **건물 입력**

  건물 입력 시 번호가 1 이상 3 이하가 아니라면 다시 입력 받습니다.

  숫자가 입력이 안됐다면 입력 문자열이 "q"인지 확인하고 "q"라면 예약 메뉴로 돌아가고 아니라면 다시 입력 받습니다.

- **호실 입력**

  기존 호실은 8 개로 고정이었으나 현재 각 건물마다 호실이 0 개 이상 9 개 이하로 가변적으로 변했습니다.

  호실 입력값이 1 미만 rooms(현재 건물 호실 개수) 초과라면 오류 메시지 출력 후 다시 입력 받습니다.

- **해당 호실 예약 가능 유무**

새로운 예외처리로 tables 의 해당 인덱스의 문자열이 "X"라면 예약이 불가능한 것이므로 "해당 호실은 현재 예약이 불가능합니다."라는 오류 메세지 출력 후 예약 메뉴로 돌아갑니다.

- **현재 사용자의 예약 유무(1 일 1 예약 초과)**

  이중 반복문이 아닌 삼중 반복문을 이용합니다.

- 첫 번째 반복문은 건물 인덱스 접근용으로 카운터 변수는 0 부터 2 까지 1 씩 증가합니다.

- 두 번째 반복문은 각 호실 접근용으로 카운터 변수는 기존 0 부터 7 까지 1 씩 증가하던 것에서 0 부터 해당 건물의 호실 개수(tables 의 해당 건물의 size)에 -1 한 숫자까지 1 씩 증가합니다.

- 세 번째 반복문은 각 교시 접근용으로 교시 인덱스인 2 부터 7 까지 1 씩 증가합니다.

- tables 의 모든 요소를 탐색해서 현재 사용자 아이디와 같은 아이디가 발견될 시 예약된 건물, 호실, 교시를 출력하고 예약 메뉴로 돌아갑니다.

5-4. 예약 취소 및 조회

[추가 변수]

filename: 파일명 저장용, 문자열

- 예약 취소 및 조회는 반복문 실행 과정이 **5-3. 예약 신청**의 1 일 1 예약 초과 확인에서 설명한 반복문 실행 과정과 동일합니다.

- 예약 취소, 조회 두 기능 모두 사용자 아이디와 같은 요소가 발견되면 해당 건물명, 호실, 교시를 출력해줍니다.

-

[동작]

- **예약 취소**

  1. 1 일 1 예약 초과 확인용 반복문과 동일하게 실행합니다.
  2. 예약 내역이 확인되면 현재 예약된 건물명, 호실, 교시를 출력 후 예약을 취소할 것인지 묻습니다. 예약 내역이 없으면 예약 메뉴로 돌아갑니다.

- "Y" 입력 시 tables 의 해당 인덱스의 문자열을 지우고 다시 동일 인덱스에 "0"을 add 합니다.

  filename 에 해당 건물 파일명을 저장하고 file_change 함수에 매개변수로 filename 과 tables 의 현재 건물의 요소를 넣고 실행합니다. 그 후 예약 메뉴로 돌아갑니다.

- "N" 입력 시 예약 메뉴로 돌아갑니다.

- **예약 조회**

  1. 1 일 1 예약 초과 확인용 반복문과 동일하게 실행합니다.
  2. 예약 내역이 확인되면 현재 예약된 건물명, 호실, 교시를 출력 후 예약 메뉴로 돌아갑니다. 예약 내역이 없으면 예약 메뉴로 돌아갑니다.

5-5. 파일 수정(함수 file_change)

[입력 매개변수]

- 입력 매개변수가 추가됩니다.

  build: 수정할 건물의 요소, ArrayList<ArrayList<String>>

[동작]

1. 현재 파일에 들어있는 모든 내용을 삭제합니다.
2. 기존 방식과 동일하게 파일 한 줄에 요소들을 작성합니다.

- 기존에는 호실이 8 개로 고정되었기에 외부 반복문은 카운터 변수가 0 부터 7 까지 1 씩 증가로 고정이었습니다. 이번 프로그램에서는 호실이 0 개부터 9 개까지 가변적이므로 0 부터 build 의 size 에 -1 한 값까지 1 씩 증가하며 실행됩니다.(build 의 size = 호실의 개수)
- 배열을 사용하던 부분은 build 를 이용하여 해당 인덱스의 요소를 파일에 작성합니다.
- 줄바꿈 조건은 외부 반복문 카운터 변수가 고정된 호실 개수인 8 에서 1 을 뺀 7 이 아닐 때였으나 이번 프로그램에서는 build 의 호실 개수인 build 의 size 에 1 을 뺀 값이 아닐 때로 수정합니다.

6. 관리자 메뉴 - ManagerInfo.java

6-1. ManagerInfo 클래스 멤버 변수

String 배열 타입의 builds 는 각 관의 이름(공학관, 생명공학관, 상허연구관)을 저장합니다. **4. 현황판 출력**과 동일하게 build1, 2, 3, tables 변수를 생성합니다 members 변수는 arraylist 타입의 학생 정보로, MemberInfo.java 에서 members 변수와 동일하게 작동합니다.

deleteStudent 는 int 타입으로 학생 정보 삭제할 학생의 index 값을 의미합니다.

String 배열 타입의 builds:

각 관의 이름(공학관, 생명공학관, 상허연구관)을 저장합니다

String 형태의 build1, build2, build3, tables:

<4. 현황판 출력>과 동일하게 각 관의 현황판을 저장하는 변수입니다.

build1 은 "공학관"의 현황판 정보를, build2 는 "생명공학관"의 현황판 정보를, build3 은 "상허연구관"의 현황판 정보를 저장합니다.

ArrayList<Student> 타입의 member:

학생 정보를 담는 변수입니다.

ArrayList 에 추가되는 값들은 제네릭 Student 타입으로 명시했습니다.

int 형태의 deleteStudent:

학생 정보 삭제 시 삭제할 학생의 member 내 인덱스 값을 저장하는 변수입니다.

6-2. 생성자

ManagerInfo 생성자는 다음과 같이 동작합니다:

1. StudentInfo 파일을 받아서 검사합니다.

   이 과정은 MemberInfo 객체 생성 시 진행되는 과정과 동일합니다.

2. 파일 검사에 문제가 없다면, Student 타입의 student1 을 생성하고 student1 을 member 에 추가합니다.

3. build1, build2, build3 에 file_check 함수를 해당 파일명에 맞게 실행한 후 결과를 대입합니다.

4. 각각의 build 정보를 tables 에 추가합니다.

5. NotificationFileCheck 변수에 NotificationFile 의 경로를 저장하고 해당 파일의 존재 여부를 검사합니다.

6. ManagerInfo_menu 메소드를 실행합니다.

6-3. 메소드

6-3-1. 관리자 메뉴 함수

-ManagerInfo_menu()

ManagerInfo_menu() 메소드는 다음과 같이 동작합니다:

[변수]

sc : Scanner 객체로 사용자의 입력을 받습니다.

[동작]

1. while(true) 문을 사용하여 메뉴를 계속 실행합니다.

2. 사용자로부터 입력을 받습니다.

3. 입력값(sc)을 trim() 함수를 사용하여 공백을 제거하고 parseInt() 함수를 사용하여 정수형으로 변환합니다.
4. try-catch 문을 사용하여 입력값(sc)를 검사합니다.
5. switch case 문을 사용하여 입력값에 따라 해당하는 메뉴를 선택하고 실행합니다.

[예외처리]

try-catch 문을 사용하여 입력값(sc)의 예외를 처리합니다.

입력값이 정수형으로 변환되지 않거나 예외가 발생한 경우, 오류 메시지를 출력하고 다시 입력을 받습니다.

6-3-2. 스터디룸 관리 함수

- Choose_building()

Choose_building() 메소드는 다음과 같이 동작합니다:

[변수]

sc: Scanner 객체로 사용자의 입력을 받습니다.

building_choice: 선택한 건물 번호를 저장하는 변수입니다.

[동작]

1. while(true) 문을 사용하여 건물 선택을 계속 실행합니다.
2. 사용자로부터 건물 번호를 입력받습니다.
3. 입력값(sc)을 trim() 함수를 사용하여 공백을 제거하고 parseInt() 함수를 사용하여 정수형으로 변환합니다.
4. try-catch 문을 사용하여 입력값(building_choice)를 검사합니다.

5. 입력값(building_choice)이 1 부터 3 사이의 값인 경우, tables.get(building_choice-1)을 사용하여 선택한 건물의 ArrayList 를 가져옵니다.
6. 가져온 ArrayList 를 인자로 전달하여 Studyroomediting(ArrayList<ArrayList<String>> build) 메소드로 이동합니다.
7. 입력값(building_choice)이 4 인 경우, return 문을 사용하여 메소드를 빠져나와 ManagerInfo_menu() 메소드로 이동합니다.

[예외처리]

try-catch 문을 사용하여 입력값(building_choice)의 예외를 처리합니다.

입력값이 정수형으로 변환되지 않거나 예외가 발생한 경우, 오류 메시지를 출력하고 다시 입력을 받습니다.

-Studyroomediting(ArrayList<ArrayList<String>> build))

Studyroomediting(ArrayList<ArrayList<String>> build) 메소드는 다음과 같이 동작합니다:

[변수]

sc: Scanner 객체로 사용자의 입력을 받습니다.

[동작]

1. while(true) 문을 사용하여 메뉴를 계속 실행합니다.
2. 사용자로부터 메뉴 번호를 입력받습니다.
3. 입력값(sc)을 trim() 함수를 사용하여 공백을 제거하고 parseInt() 함수를 사용하여 정수형으로 변환합니다.
4. try-catch 문을 사용하여 입력값(sc)를 검사합니다.

5. 입력값(sc)에 해당하는 메뉴에 대해 switch case 문을 사용하여 해당 메소드를 실행합니다.
6. 선택한 메소드에는 관리자가 선택한 건물의 ArrayList(build)을 인자로 전달합니다.

[예외처리]

try-catch 문을 사용하여 입력값(sc)의 예외를 처리합니다.

입력값이 정수형으로 변환되지 않거나 예외가 발생한 경우, 오류 메시지를 출력하고 다시 입력을 받습니다.

-Studyroom_add(ArrayList<ArrayList<String>> build))

Studyroom_add(ArrayList<ArrayList<String>> build)메소드는 다음과 같이 동작합니다:

[변수]

sc: Scanner 객체로 사용자의 입력을 받습니다.

Member: 추가 할 스터디룸의 한도인원을 저장하는 변수입니다.

[동작].

1. 사용자로부터 한도 인원을 입력받습니다.
2. 입력값(sc)을 trim() 함수를 사용하여 공백을 제거하고 parseInt() 함수를 사용하여 정수형으로 변환합니다.
3. try-catch 문을 사용하여 입력값(sc)를 검사합니다.
4. 입력값(member)이 2 부터 9 사이인 값인 경우 해당 건물에 한도인원이 member 인 build.size()+1 호실이 생성됩니다.
5. 선택한 메소드에는 관리자가 선택한 건물의 ArrayList(build)을 인자로 전달합니다.

[예외처리]

try-catch 문을 사용하여 입력값(sc)의 예외를 처리합니다.

입력값이 정수형으로 변환되지 않거나 예외가 발생한 경우, 오류 메시지를 출력하고 다시 입력을 받습니다.

-Studyroom_edit(ArrayList<ArrayList<String>> build)

Studyroom_edit(ArrayList<ArrayList<String>> build) 메소드는 다음과 같이 동작합니다:

이 메소드에서 입력받는 모든 문자열에는 기본적으로 trim()함수를 통해 앞뒤의 공백을 제거한 상태로 변수에 대입됩니다

[변수]

roomNum: 수정할 스터디룸의 호실 번호를 저장하는 변수입니다.

member: 수정할 스터디룸의 한도 인원을 저장하는 변수입니다.

sc: Scanner 객체로 사용자의 입력을 받습니다.

[동작]

1. while(true) 문을 사용하여 수정할 호실을 입력받습니다.
2. roomNum 변수에 사용자로부터 입력받은 호실 번호를 저장합니다.
3. 만약 roomNum 이 build.size()보다 크다면, 해당 호실이 존재하지 않는다는 메시지를 출력하고 다시 수정할 호실을 입력받습니다.

   만약 roomNum 이 build.size()보다 작거나 같다면, while(true) 문을 사용하여 수정할 호실의 한도 인원(member)을 입력받습니다.
4. 입력받은 한도 인원(member)이 2 에서 9 사이의 정수인지 확인하기 위해 if 문과 try-catch 문을 사용합니다.

   만약 한도 인원이 2 에서 9 사이의 정수가 아니라면 다시 입력 받습니다.

   만약 한도 인원이 2 에서 9 사이의 정수라면 while(true) 문을 사용하여 호실의 한도 인원을 수정할 것인지 확인합니다.
5. 사용자에게 수정할 호실의 한도 인원을 수정하겠냐고 물어봅니다.

6. 사용자의 입력값을 받고, 입력값이 'Y' 또는 'N'인지 확인합니다.

-입력값이 'Y'인 경우:

ArrayList 형 변수 room 을 선언합니다.

room 에 set() 함수를 사용하여 한도 인원의 값을 수정합니다.

기존 한도 인원과 수정한 한도 인원을 비교하여 수정한 한도 인원이 기존 한도 인원보다 작다면, 해당 호실에 대한 예약된 학번 값을 모두 '0'으로 변경하고 예약 취소 안내 문자열을 출력합니다.

build.set(roomNum-1, room)을 사용하여 build 에서 수정된 호실에 해당하는 ArrayList 를 변경합니다.

-입력값이 'N'인 경우:

메소드를 종료하고 스터디룸 관리 메뉴로 이동합니다.

-입력값이 'Y' 또는 'N'이 아닌 경우:

에러 메시지를 출력하고 다시 입력을 받습니다.

[예외처리]

try-catch 문을 사용하여 입력값의 예외를 처리합니다.

입력값이 정수형으로 변환되지 않거나 예외가 발생한 경우, 'q'인 경우 스터디룸 수정을 종료하고 그 외의 경우에는 에러 메시지를 출력하고 다시 입력을 받습니다.

-Studyroom_delete(ArrayList<ArrayList<String>> build)

Studyroom_delete(ArrayList<ArrayList<String>> build) 메소드는 다음과 같이 동작합니다:

이 메소드에서 입력받는 모든 문자열에는 기본적으로 trim()함수를 통해 앞뒤의 공백을 제거한 상태로 변수에 대입됩니다

[변수]

roomNum: 삭제할 스터디룸의 호실 번호를 저장하는 변수입니다.

sc: Scanner 객체로 사용자의 입력을 받습니다.

[동작]

1. while(true) 문을 사용하여 삭제할 호실을 입력받습니다.
2. roomNum 변수에 사용자로부터 입력받은 호실 번호를 저장합니다.
3. 만약 roomNum 이 build.size()보다 크다면, 해당 호실이 존재하지 않는다는 메시지를 출력하고 다시 삭제할 호실을 입력받습니다.

   만약 roomNum 이 build.size()보다 작거나 같다면, while(true) 문을 사용하여 호실을 삭제할 것인지 확인합니다.
4. 사용자에게 호실을 삭제하겠냐고 묻고,사용자의 입력값을 받습니다.
5. 입력값이 'Y' 또는 'N'인지 확인합니다.

   -입력값이 'Y'인 경우:

   for 문과 if 문을 사용하여 삭제할 호실에 예약되어 있는 학번 값을 찾고, 해당 호실과 교시에 예약이 취소되었다는 문자열을 변수에 저장합니다.

   build.remove(roomNum-1)을 사용하여 build 에서 삭제할 호실에 해당하는 ArrayList 를 삭제합니다.

   삭제된 호실 뒤의 호실들의 번호를 1 씩 감소시키기 위해 for 반복문과 ArrayList 의 set() 함수를 사용하여 호실 번호를 변경합니다.

   -입력값이 'N'인 경우:

   메소드를 종료하고 스터디룸 관리 메뉴로 이동합니다.

   -입력값이 'Y' 또는 'N'이 아닌 경우:

에러 메시지를 출력하고 다시 입력을 받습니다.

[예외처리]

try-catch 문을 사용하여 입력값의 예외를 처리합니다.

입력값이 정수형으로 변환되지 않거나 예외가 발생한 경우, 'q'인 경우 스터디룸 삭제를 종료하고 그 외의 경우에는 에러 메시지를 출력하고 다시 입력을 받습니다.

-Reservation_banmenu(ArrayList<ArrayList<String>> build)

이 메소드는 관리자가 예약 불가 설정 메뉴에 들어왔을 때 사용 불가 설정 또는 사용 가능 설정을 선택할 수 있도록 하며, 선택에 따라 해당 메뉴로 이동합니다.

Reservation_banmenu 메소드는 다음과 같이 동작합니다:

[변수]

build: 스터디룸 정보를 담고 있는 2 차원 ArrayList 입니다.

[동작]

1. while(true)문을 사용하여 메뉴 선택을 반복합니다.
2. 정수형 변수 choice 를 입력 받습니다.
3. choice 값이 1, 2, 3 중 하나인지 확인합니다. choice 값에 따라 switch 문을 사용하여 해당하는 메소드를 호출하여 해당 메뉴로 이동합니다.

   - 입력된 값이 1, 2, 3 중 하나가 아니라면 에러 메시지를 출력하고 continue 를 사용하여 다시 정수를 입력 받습니다.

   - 입력된 값이 정수형이 아니라면 try-catch 문을 사용하여 에러 메시지를 출력하고 정수를 다시 입력 받습니다.

4. 각 메소드 호출 후에 return 문을 사용하여 메소드가 종료되면 바로 스터디룸 관리 메뉴로 돌아가도록 합니다.

[예외처리]

정수형이 아닌 값이 입력될 경우 try-catch 문을 사용하여 에러 메시지를 출력하고 정수를 다시 입력 받도록 합니다.

-Reservation_ban(ArrayList<ArrayList<String>> build)

이 메소드는 Reservation_banmenu 메소드에서 관리자가 '예약불가 설정'하는 메뉴를 선택했을 때 실행되는 메소드입니다.

Reservation_ban(ArrayList<ArrayList<String>> build) 메소드는 다음과 같이 동작합니다:

[변수]

build: 스터디룸 정보를 담고 있는 2 차원 ArrayList 입니다.

roomNum: 예약 불가 설정할 호실을 저장하는 정수형 변수입니다.

room: 선택된 호실에 대한 정보를 담은 ArrayList 입니다.

roomTim: 예약 불가 설정할 교시를 저장하는 정수형 변수입니다.

notice: 예약 정보 취소 메시지를 저장하는 문자열 변수입니다.

[동작]

1. while(true)문을 사용하여 호실 선택을 반복합니다.
2. 정수형 변수 roomNum 을 입력받습니다.
3. roomNum 값이 1 보다 작거나 현재 건물에 있는 호실 수를 넘는지 확인합니다.

   -조건을 만족하지 않으면 에러 메시지를 출력하고 continue 를 사용하여 다시 호실을 입력받습니다.

   - roomNum 값이 정수형이 아니라면 try-catch 문을 사용하여 예외 처리를 합니다.

   - catch 문 안에서 입력된 값이 'q'인지 확인하고 'q'일 경우 return 을 사용하여 함수를 종료합니다.

-'q'가 아닐 경우 에러 메시지를 출력한 후 다시 한 번 roomNum 를 입력받습니다.

4. room 에 build.get(roomNum-1)을 사용하여 선택된 호실에 대한 정보를 저장합니다.
5. while(true)문을 사용하여 교시 선택을 반복합니다.
6. 정수형 변수 roomTim 을 입력받습니다.
7. roomTim 값이 1 부터 6 사이의 정수인지 확인합니다.

   - 조건을 만족하지 않으면 에러 메시지를 출력하고 continue 를 사용하여 roomTim 을 다시 입력받습니다.

   - roomTim 값이 위의 모든 조건을 만족하고 입력이 완료되었다면 while(true)문을 빠져나옵니다.

8. 세 번째 while(true)문에서 'Y' 또는 'N'을 입력받습니다.

-'N'을 입력받을 경우:

return 을 사용하여 함수를 종료합니다.

-'Y'나 'N'이 아닌 문자를 입력받을 경우:

에러 메시지를 출력한 후 다시 입력받습니다.

-'Y'를 입력받았을 경우:

if 문으로 현재 예약 불가 설정하려는 호실의 교시를 예약한 학생이 있는지 확인합니다.

학생이 있다면 notice 문자열에 해당 예약 정보가 취소되었다는 메시지를 추가합니다.

room.set() 메소드를 사용하여 해당하는 교시를 "X"로 변경합니다.

build.set() 메소드를 사용하여 build 에 대한 정보도 수정합니다.

build 에 대한 정보가 변경되었다면 file_change() 메소드를 사용하여 해당하는 건물의 txt 파일을 수정합니다.

[예외처리]

roomNum 값의 유효성 확인:

만약 roomNum 이 1 보다 작거나 현재 건물에 있는 호실 수를 넘는 경우, 에러 메시지를 출력하고 continue 를 사용하여 다시 호실을 입력받도록 합니다.

roomNum 값의 데이터 타입 예외처리:

try-catch 문을 사용하여 roomNum 이 정수형이 아닐 경우 예외를 처리합니다.

catch 문 안에서 입력된 값이 'q'인지 확인하고 'q'일 경우 return 을 사용하여 함수를 종료합니다.

'q'가 아닌 다른 값일 경우, 에러 메시지를 출력한 후 다시 한 번 roomNum 를 입력받도록 합니다.

roomTim 값의 유효성 확인:

roomTim 값이 1 부터 6 사이의 정수가 아니라면, 에러 메시지를 출력하고 continue 를 사용하여 다시 교시를 입력받도록 합니다.

roomTim 값의 데이터 타입 예외처리:

try-catch 문을 사용하여 roomTim 이 정수형이 아닐 경우 예외를 처리합니다.

catch 문 안에서 입력된 값이 'q'인지 확인하고 'q'일 경우 return 을 사용하여 함수를 종료합니다.

'q'가 아닌 다른 값일 경우, 에러 메시지를 출력한 후 다시 한 번 roomTi 을 입력받도록 합니다.

-Reservation_allow(ArrayList<ArrayList<String>> build)

이 메소드는 예약 불가 설정된 호실의 교시를 예약 가능으로 변경하여 스터디룸 정보를 업데이트하고 파일을 수정합니다.

Reservation_allow(ArrayList<ArrayList<String>> build) 메소드는 다음과 같이 동작합니다:

[변수]

build: 스터디룸 정보를 담고 있는 2 차원 ArrayList 입니다.

roomNum: 예약 가능으로 설정할 호실을 저장하는 정수형 변수입니다.

room: 선택된 호실에 대한 정보를 담은 ArrayList 입니다.

roomTim: 예약 가능으로 설정할 교시를 저장하는 정수형 변수입니다.

[동작]

Reservation_ban(ArrayList<ArrayList<String>> build) 메소드와 유사하게 동작합니다

- Reservation_ban(ArrayList<ArrayList<String>> build) [동작] 7 번 과정까진 동일하게 동작합니다.

8. 세 번째 while(true)문에서 'Y' 또는 'N'을 입력받습니다.

-'N'을 입력받을 경우:

return 을 사용하여 함수를 종료합니다.

-'Y'나 'N'이 아닌 문자를 입력받을 경우:

에러 메시지를 출력한 후 다시 입력받습니다.

-'Y'를 입력받았을 경우:

room.set() 메소드를 사용하여 해당하는 교시를 "0"으로 변경합니다.

build.set() 메소드를 사용하여 build 에 대한 정보도 수정합니다.

build 에 대한 정보가 변경되었다면 file_change() 메소드를 사용하여 해당하는 건물의 txt 파일을 수정합니다.

[예외처리]

- Reservation_ban(ArrayList<ArrayList<String>> build) 메소드와 동일하게 동작합니다

6-3-3. 사용자 관리 함수

-StudentEditing()

StudentEditing() 메소드는 학생을 관리하는 기능을 수행하는 메소드입니다. 다음과 같이 동작합니다:

[변수]

check: while(true)문을 탈출하기 위한 변수로, check 가 1 일 때 while(true)문을 종료합니다.

inputid: 사용자로부터 입력받은 학번을 저장하는 변수입니다.

sc: Scanner 객체로 사용자의 입력을 받습니다.

[동작]

1. while(true) 문을 사용하여 해당 기능을 반복합니다.
2. check 변수를 사용하여 while(true)문을 탈출합니다.
3. 사용자로부터 학번을 입력받습니다.
4. inputid 변수에 사용자로부터 입력받은 학번을 저장합니다.
5. 만약 inputid 값이 'q'라면, break 문을 사용하여 관리자 메뉴로 돌아갑니다.

   만약 inputid 값이 'q'가 아니라면, try-catch 문을 사용하여 숫자가 아닌 문자열인지 검사합니다.
6. 입력받은 학번이 형식에 맞는지 if 문을 사용하여 검사합니다.

   입력받은 학번이 형식에 맞는 경우, **finduser() 메소드**를 사용하여 해당 학번이 member 에 있는지 확인합니다.
7. finduser() 메소드를 통해 inputid 가 member 에 저장되어 있다면, 사용자에게 학번을 삭제할 것인지 물어봅니다.
8. 사용자가 'Y' 또는 'N'을 입력하여 삭제 여부를 결정할 수 있습니다.

사용자가 'Y'를 입력한 경우:

- member 에서 해당 학번을 삭제합니다.

- 삭제된 학번의 행을 출력합니다.

- BufferWriter 함수를 사용하여 StudentInfoFile 을 받아와 전체 내용을 member 에 저장된 학생들의 정보로 교체합니다.

- for 문을 사용하여 member 에서 한 명씩 학생 정보를 받아와 write 함수로 한 줄씩 입력합니다.

- 이후 check 변수에 1 을 저장합니다.

사용자가 'N'을 입력한 경우:

check 변수에 1 을 저장하고 관리자 메뉴로 돌아갑니다.

[예외처리]

try-catch 문을 사용하여 입력값의 예외를 처리합니다.

입력값이 숫자로 변환되지 않거나 예외가 발생한 경우, 'q'인 경우 학생 삭제를 종료하고 그 외의 경우에는 에러 메시지를 출력하고 다시 입력을 받습니다.

-finduser(string uid)

MemberInfo.java 에서 사용한 isUniqueID 메서드와 동일한 구성입니다.

단 인자로 넣은 string 자료형 uid 변수와 동일한 문자열이 발견되면 해당 인덱스를 deleteSutdent 에 대입하고 종료합니다.

6-3-4. 공지 설정 함수

-WriteNotification()

WriteNotification() 메소드는 다음과 같이 동작합니다:

[변수]

inputText: 공지 내용을 저장하는 string 타입 변수입니다. 사용자로부터 Scanner 를 사용하여 입력받습니다.

[동작]

inputText 를 Scanner 를 통해 사용자로부터 입력받습니다.

공지 수정 확인 문구를 출력합니다.

- "Y" 입력 시

BufferWriter 를 사용하여 공지 파일(Notification.txt)을 받아옵니다.

BufferWriter 의 write 메소드를 사용하여 inputText 를 파일에 기록합니다.

파일에 기록된 후 수정되었다는 메시지를 출력합니다.

관리자 메뉴로 돌아갑니다.

- "N" 입력 시

관리자 메뉴로 돌아갑니다.

[예외처리]

이 메소드에서는 주로 파일에 데이터를 기록하는 역할을 수행하므로 예외처리는 별도로 필요하지 않습니다.

6-3-5. 기타

file_check(String filename)

– 이 메소드는 <**4. 현황판 출력**>의 file_check(string filename) 메소드에서 언급된 내용과 동일합니다.