1. 개요

사용자의 입력에 따라 회원 및 비회원으로 영화를 예매하는 프로그램인 Ticketer를 검사한 결과임.

Ticketer.py는 외부 라이브러리인 pandas 가 설치되어 있어야 정상적으로 구동됨.

설치 방법(파이참)

좌측 상단 File -> Settings -> Project : Tickecter -> Project Interpreter -> 우측에 + 버튼 클릭 ->

상단 검색창에 pandas 입력 -> pandas 클릭 후 좌측 하단 Install Package 클릭.

1. 단위 검사

# Ticketer 클래스

* + 1. main 함수( 수정 전 )

MenuManager 클래스를 MM으로 호출한 상태, MM.getMI().getwhere() 의 값을 0,1,2만 가능하게 assert 문을 걸어준 상태.

1. 부분 설계

MM.getMI().getwhere()의 값이 0,1,2이 아니라면 AssertionError를 출력함.

사용자의 입력을 무한 루프를 탈출할 때까지 받아가며 다음과 같은 진행을 한다.

1. “EXIT”가 입력된 경우 : 루프를 탈출하고 “프로그램이 종료됩니다.”를 출력하고 프로그램을 종료한다.
2. “RESTART”가 입력된 경우 :

2-1. MM.getMI().getwhere == 0 or MM.getMI().getmenuNum == 4100 일 때 – 아무 것도 진행하지 않고 입력을 대기한다.

2-2. MM.getMI().getwhere != 0 and MM.getMI().getmenuNum != 4100 일 때

MM.getMI().setMI()함수를 호출해 self.\_\_menuNum = 4200, self.\_\_isMember = False, self.\_\_where = 1로 변경한다.

MM의 변수 username과 password를 “”으로 변경한다.

출력 화면을 clear한다.

MM.print\_login\_menu() 함수를 호출해 로그인 메뉴를 출력한다.

입력을 대기한다.

1. “BACK”이 입력된 경우 :

3-1. MM.getMI().getwhere != 2 or MM.getMI().getmenuNum < 4300 일 때 – 아무 것도 진행하지 않고 입력을 대기한다.

3-2. MM.getMI().getwhere == 2 and MM.getMI().getmenuNum >= 4300 일 때

MM.getMI().setMI()함수를 호출해 self.\_\_menuNum = 4300 으로 변경한다.(나머지는 그대로)

출력 화면을 clear한다.

MM.print\_main\_menu() 함수를 호출해 로그인 메뉴를 출력한다.

입력을 대기한다.

1. 그 외 :

MM.manageMenu() 함수를 input을 인자로 호출한다.

입력을 대기한다.

1. 검사 방법

전체 프로그램을 실행시킨다. 입력값 “EXIT”,”RESTART”,”BACK” 과 MM.getMI().getwhere()의 값에 의해서 결과가 달라지므로 Menu 클래스에서 menuInfo.menuInfo() 함수 초기화를 통해 MM.getMI().getwhere() 값을 바꿔가며 검사한다. 그리고 각 입력마다 print()문으로 변한 변수값들을 확인한다.

1. Test Cases 및 결과 정리

목적 : 입력값과 MM.getMI().getwhere() 의 값에 따른 출력 확인 및 assert 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(4100,False,0) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 맨 첫줄엔 “현재 시간을 입력해 주세요.” 출력 |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(4100,False,1) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | ‘’ |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(4100,False,2) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4100,False,0) | 부분 설계 2-1번 결과 | 부분 설계 2-1번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4100,False,1) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4100,False,2) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4100,True,2) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4100,False,0) | 부분 설계 3-1번 결과 | 부분 설계 3-1번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4100,False,1) | 부분 설계 3-1번 결과 | 부분 설계 3-1번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4100,False,2) | 부분 설계 3-2번 결과 | 부분 설계 3-2번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4100,True,2) | 부분 설계 3-2번 결과 | 부분 설계 3-2번 결과 | ‘’ |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(4100,True,-1) | AssertionError | AssertionError | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4100,True,-1) | AssertionError | AssertionError | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4100,True,-1) | AssertionError | AssertionError | ‘’ |
| 입력 : “hello”  self.MI = menuInfo.menuInfo(4100,True,0) | 입력 대기 | 입력 대기 | ‘’, 입력형식이 맞지 않으므로 “입력 형식이 맞지 않습니다.” 출력 |
| 입력 : “hello”  self.MI = menuInfo.menuInfo(4100,True,0) | 입력 대기 | 입력 대기 | ‘’, ‘’ |
| 입력 : “-1231234”  self.MI = menuInfo.menuInfo(4100,True,1) | 입력 대기 | 입력 대기 | ‘’, ‘’ |
| 입력 : “안녕”  self.MI = menuInfo.menuInfo(4100,True,2) | 입력 대기 | 입력 대기 | ‘’, ‘’ |
| 입력 : “ asd asd ”  self.MI = menuInfo.menuInfo(4100,True,2) | 입력 대기 | 입력 대기 | ‘’, ‘’ |
| 입력 : “안녕”  self.MI = menuInfo.menuInfo(4100,True,15000) | AssertionError | AssertionError | ‘’ |

목적 : MM.getMI().getwhere() 과 MM.getMI().getmenuNum() 의 값에 따른 “RESTART”, “BACK” 진행 확인.

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4100,False,0) | 부분 설계 2-1번 결과 | 부분 설계 2-1번 결과 | 맨 첫줄엔 “현재 시간을 입력해 주세요.” 출력 |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4100,False,1) | 부분 설계 2-1번 결과 | 부분 설계 2-1번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4100,False,2) | 부분 설계 2-1번 결과 | 부분 설계 2-1번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4200,False,0) | 부분 설계 2-1번 결과 | 부분 설계 2-2번 결과 | menuInfo의 첫번째 인자와 세 번째 인자의 충돌 |

* + 1. main 함수( 수정 후 )

MenuManager 클래스를 MM으로 호출한 상태

1. 부분 설계

사용자의 입력을 무한 루프를 탈출할 때까지 받아가며 다음과 같은 진행을 한다.

1. “EXIT”가 입력된 경우 : 루프를 탈출하고 “프로그램이 종료됩니다.”를 출력하고 프로그램을 종료한다.
2. “RESTART”가 입력된 경우 :

2-1. MM.getMI().getmenuNum == 4100 일 때 – 아무 것도 진행하지 않고 입력을 대기한다.

2-2. MM.getMI().getmenuNum != 4100 일 때

MM.getMI().setMI()함수를 호출해 self.\_\_menuNum = 4200, self.\_\_isMember = False로 변경한다.

MM의 변수 username과 password를 “”으로 변경한다.

출력 화면을 clear한다.

MM.print\_login\_menu() 함수를 호출해 로그인 메뉴를 출력한다.

입력을 대기한다.

1. “BACK”이 입력된 경우 :

3-1. MM.getMI().getmenuNum < 4300 일 때 – 아무 것도 진행하지 않고 입력을 대기한다.

3-2. MM.getMI().getmenuNum >= 4300 일 때

MM.getMI().setMI()함수를 호출해 self.\_\_menuNum = 4300 으로 변경한다.(나머지는 그대로)

출력 화면을 clear한다.

MM.print\_main\_menu() 함수를 호출해 로그인 메뉴를 출력한다.

입력을 대기한다.

1. 그 외 :

MM.manageMenu() 함수를 input을 인자로 호출한다.

입력을 대기한다.

1. 검사 방법

전체 프로그램을 실행시킨다. 입력값 “EXIT”,”RESTART”,”BACK” 과 MM.getMI().getmenuNum()의 값에 의해서 결과가 달라지므로 Menu 클래스에서 menuInfo.menuInfo() 함수 초기화를 통해 MM.getMI().getmenuNum() 값을 바꿔가며 검사한다. 그리고 각 입력마다 print()문으로 변한 변수값들을 확인한다.

1. Test Cases 및 결과 정리

목적 : 입력값과 MM.getMI().getmenuNum() 의 값에 따른 출력 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(4100,False) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 맨 첫줄엔 “현재 시간을 입력해 주세요.” 출력 |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(4200,False) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | ‘’ |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(4312,False) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | ‘’ |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(43212,False) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | ‘’ |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(1234,False) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | ‘’ |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo  (-123,False) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | ‘’ |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(99999,True) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | ‘’ |
| 입력 : “EXIT”  self.MI = menuInfo.menuInfo(0,True) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4100,False) | 부분 설계 2-1번 결과 | 부분 설계 2-1번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4200,False) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4300,False) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(43142,False) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4212,False) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(1234,False) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo  (-1234,False) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(99999,False) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(4101,False) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 | ‘’ |
| 입력 : “RESTART”  self.MI = menuInfo.menuInfo(0,False) | 부분 설계 2-2번 결과 | 부분 설계 2-2번 결과 |  |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4100, False) | 부분 설계 3-1번 결과 | 부분 설계 3-1번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4200, False) | 부분 설계 3-1번 결과 | 부분 설계 3-1번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4300, False) | 부분 설계 3-2번 결과 | 부분 설계 3-2번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(43142, False) | 부분 설계 3-2번 결과 | 부분 설계 3-2번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4212, False) | 부분 설계 3-1번 결과 | 부분 설계 3-1번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4222, False) | 부분 설계 3-1번 결과 | 부분 설계 3-1번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo(4101, False) | 부분 설계 3-1번 결과 | 부분 설계 3-1번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo  (43333, False) | 부분 설계 3-2번 결과 | 부분 설계 3-2번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo  (9999999, False) | 부분 설계 3-2번 결과 | 부분 설계 3-2번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo  (-1234, False) | 부분 설계 3-1번 결과 | 부분 설계 3-1번 결과 | ‘’ |
| 입력 : “BACK”  self.MI = menuInfo.menuInfo  (0, False) | 부분 설계 3-1번 결과 | 부분 설계 3-1번 결과 | ‘’ |
| 입력 : “hello!!”  self.MI = menuInfo.menuInfo(4100,False) | 부분 설계 4번 결과 | 부분 설계 4번 결과 | ‘’, 입력형식이 맞지 않으므로 “입력 형식이 맞지 않습니다.” 출력 |
| 입력 : “hello”  self.MI = menuInfo.menuInfo(4100, False) | 부분 설계 4번 결과 | 부분 설계 4번 결과 | ‘’, ‘’ |
| 입력 : “-1231234”  self.MI = menuInfo.menuInfo(4100, False) | 부분 설계 4번 결과 | 부분 설계 4번 결과 | ‘’, ‘’ |
| 입력 : “안녕”  self.MI = menuInfo.menuInfo(4100, False) | 부분 설계 4번 결과 | 부분 설계 4번 결과 | ‘’, ‘’ |
| 입력 : “ asd asd ”  self.MI = menuInfo.menuInfo(4100, False) | 부분 설계 4번 결과 | 부분 설계 4번 결과 | ‘’, ‘’ |

# menuInfo 클래스

* + 1. \_\_init\_\_ 함수( 수정 전 )

1. 부분 설계

인자로 전달받은 mN, iM, w 을 setMI 함수에게 넘겨주어 초기화 해줌.

1. 검사 방법

단순히 인자 전달만을 수행하기 때문에 검사하지 않음.

1. Test Cases 및 결과 정리

위와 같음.

* + 1. \_\_init\_\_ 함수( 수정 후 )

1. 부분 설계

인자로 전달받은 mN, iM을 setMI 함수에게 넘겨주어 초기화 해줌.

1. 검사 방법

단순히 인자 전달만을 수행하기 때문에 검사하지 않음.

1. Test Cases 및 결과 정리

위와 같음.

* + 1. getmenuNum, getisMember, getwhere 함수 ( 수정 전 )

단순히 return만을 수행하는 getter 함수이기 때문에 검사하지 않음.

* + 1. getmenuNum, getisMember함수 ( 수정 전 )

단순히 return만을 수행하는 getter 함수이기 때문에 검사하지 않음.

* + 1. setMI 함수( 수정 전 )

변수 mN은 int형, 변수 iM은 bool형, 변수 w는 int형으로 assert 조건을 걸어준 상태.

1. 부분 설계

인자로 받은 mN, iM, w 를 클래스 변수인 self.\_\_menuNum, self.\_\_isMember, self.\_\_where에 각각 넣어줘야함.

mN, iM, w 의 자료형이 assert 조건에 위배된다면, AssertionError 메시지를 출력해야함.

정확히 세 개의 인자를 받아야함. 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야함.

1. 검사 방법

Python console에서 해당 클래스 전부를 입력하고 menuInfo 객체를 생성해서 함수를 호출하여 변경되는 클래스 변수 mN, iM, w 의 값을 확인 및 에러 메시지를 체크함.

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 3개) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| setMI() | TypeError | TypeError |  |
| setMI(1) | TypeError | TypeError |  |
| setMI(1,0) | TypeError | TypeError |  |
| setMI(1,1,1) | TypeError없음 | TypeError없음 | 하지만 AssertionError |
| setMI(1,True,1) | TypeError없음 | TypeError없음 |  |
| setMI(1,2,15124,55) | TypeError | TypeError |  |
| setMI(1,2,3,4,5,1,1,3434) | TypeError | TypeError |  |

목적 : 첫 번째 인자 조건(int형) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| setMI(0,True,1) | AssertionError없음 | AssertionError없음 |  |
| setMI(“a”,True,1) | AssertionError | AssertionError |  |
| setMI(“ㅁㅁㅁ”,True,1) | AssertionError | AssertionError |  |
| setMI(-13322,True,1) | AssertionError없음 | AssertionError없음 |  |
| setMI(0.5,True,1) | AssertionError | AssertionError |  |
| setMI(13123123,True,1) | AssertionError없음 | AssertionError없음 |  |

목적 : 두 번째 인자 조건(bool형) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| setMI(1,True,1) | AssertionError없음 | AssertionError없음 |  |
| setMI(1,1,1) | AssertionError | AssertionError |  |
| setMI(1,”abc”,1) | AssertionError | AssertionError |  |
| setMI(1,False,1) | AssertionError없음 | AssertionError없음 |  |
| setMI(1,0.5,1) | AssertionError | AssertionError |  |
| setMI(1,0,1) | AssertionError | AssertionError |  |

목적 : 세 번째 인자 조건(int형) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| setMI(1,True,0) | AssertionError없음 | AssertionError없음 |  |
| setMI(1,True,”a”) | AssertionError | AssertionError |  |
| setMI(1,True, “ㅁㅁㅁ”) | AssertionError | AssertionError |  |
| setMI(1,True, -13322) | AssertionError없음 | AssertionError없음 |  |
| setMI(1,True, 0.5) | AssertionError | AssertionError |  |
| setMI(1,True, 13123123) | AssertionError없음 | AssertionError없음 |  |

목적 : 클래스 변수 self.\_\_menuNum, self.\_\_isMember, self.\_\_where 이 변경되는지 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| setMI(1,True,0) | self.\_\_menuNum: 1  self.\_\_isMember: True  self.\_\_where: 0 | self.\_\_menuNum: 1  self.\_\_isMember: True  self.\_\_where: 0 |  |
| setMI(1,True,-1) | self.\_\_menuNum: 1  self.\_\_isMember: True  self.\_\_where: -1 | self.\_\_menuNum: 1  self.\_\_isMember: True  self.\_\_where: -1 |  |
| setMI(-123,True, 1000) | self.\_\_menuNum: -123  self.\_\_isMember: True  self.\_\_where: 1000 | self.\_\_menuNum: -123  self.\_\_isMember: True  self.\_\_where: 1000 |  |
| setMI(1,False, -13322) | self.\_\_menuNum: 1  self.\_\_isMember: False  self.\_\_where: -13322 | self.\_\_menuNum: 1  self.\_\_isMember: False  self.\_\_where: -13322 |  |
| setMI(0,False, 55) | self.\_\_menuNum: 0  self.\_\_isMember: False  self.\_\_where: 55 | self.\_\_menuNum: 0  self.\_\_isMember: False  self.\_\_where: 55 |  |

* + 1. setMI 함수( 수정 후 )

변수 mN은 int형, 변수 iM은 bool형으로 assert 조건을 걸어준 상태.

1. 부분 설계

인자로 받은 mN, iM를 클래스 변수인 self.\_\_menuNum, self.\_\_isMember에 각각 넣어줘야함.

mN, iM의 자료형이 assert 조건에 위배된다면, AssertionError 메시지를 출력해야함.

정확히 두 개의 인자를 받아야함. 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야함.

1. 검사 방법

Python console에서 해당 클래스 전부를 입력하고 menuInfo 객체를 생성해서 함수를 호출하여 변경되는 클래스 변수 mN, iM의 값을 확인 및 에러 메시지를 체크함.

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 2개) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| setMI() | TypeError | TypeError |  |
| setMI(1) | TypeError | TypeError |  |
| setMI(1,0) | TypeError없음 | TypeError없음 | 하지만 AssertionError |
| setMI(1,True) | TypeError없음 | TypeError없음 |  |
| setMI(1,1,1) | TypeError | TypeError |  |
| setMI(1,True,1) | TypeError | TypeError |  |
| setMI(1,2,15124,55) | TypeError | TypeError |  |
| setMI(1,2,3,4,5,1,1,3434) | TypeError | TypeError |  |

목적 : 첫 번째 인자 조건(int형) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| setMI(0,True) | AssertionError없음 | AssertionError없음 |  |
| setMI(“a”,True) | AssertionError | AssertionError |  |
| setMI(“ㅁㅁㅁ”,True) | AssertionError | AssertionError |  |
| setMI(-13322,True) | AssertionError없음 | AssertionError없음 |  |
| setMI(0.5,True) | AssertionError | AssertionError |  |
| setMI(13123123,True) | AssertionError없음 | AssertionError없음 |  |

목적 : 두 번째 인자 조건(bool형) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| setMI(1,True) | AssertionError없음 | AssertionError없음 |  |
| setMI(1,1) | AssertionError | AssertionError |  |
| setMI(1,”abc”) | AssertionError | AssertionError |  |
| setMI(1,False) | AssertionError없음 | AssertionError없음 |  |
| setMI(1,0.5) | AssertionError | AssertionError |  |
| setMI(1,0) | AssertionError | AssertionError |  |

목적 : 클래스 변수 self.\_\_menuNum, self.\_\_isMember 이 변경되는지 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| setMI(1,True) | self.\_\_menuNum: 1  self.\_\_isMember: True | self.\_\_menuNum: 1  self.\_\_isMember: True |  |
| setMI(9999,True) | self.\_\_menuNum: 9999  self.\_\_isMember: True | self.\_\_menuNum: 9999  self.\_\_isMember: True |  |
| setMI(-123,True) | self.\_\_menuNum: -123  self.\_\_isMember: True | self.\_\_menuNum: -123  self.\_\_isMember: True |  |
| setMI(1,False) | self.\_\_menuNum: 1  self.\_\_isMember: False | self.\_\_menuNum: 1  self.\_\_isMember: False |  |
| setMI(0,False) | self.\_\_menuNum: 0  self.\_\_isMember: False | self.\_\_menuNum: 0  self.\_\_isMember: False |  |

# MenuManager 클래스

* + 1. getMI 함수

MI 객체를 단순히 return해주는 getter 함수이기 때문에 따로 테스트하지 않음.

* + 1. manageMenu 함수

menuInfo 객체인 MI에서 get해 온 menu 는 int로 이미 보장되어 있는 상태.

1. 부분 설계

하나의 인자만을 받아야 함. 아니라면 TypeError를 출력함

menu 값에 따라 부모 클래스의 함수를 input을 인자로 넘겨주어 호출하고, menu가 조건문에 해당하지 않는 값이라면 아무것도 하지 않고 -1을 리턴함.

1. 검사 방법

1 – main 함수는 manageMenu 함수 호출 부분을 제외하고 주석 처리함, 전체 프로그램을 실행함.

2 – 전체 프로그램을 실행(파일 리스트값들을 일일히 수정하기에 한계가 있음.)할 때, Menu 클래스에서 MI 를 초기화하는데, 초기화할 때의 첫 번째 인자만을 바꿔가며 실행하고, main 함수에서 manageMenu()의 리턴값을 print문을 이용해 출력함, main 함수는 manageMenu 함수 호출 부분을 제외하고 주석 처리함

1. Test Cases 및 결과 정리

1 - 목적 : manageMenu 함수의 전달 인자 개수 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| manageMenu(“123”) | TypeError없음 | TypeError없음 | 맨 첫줄엔 “현재 시간을 입력해 주세요.” 출력 |
| manageMenu(“123”,1) | TypeError | TypeError | “” |
| manageMenu(“123”,”123”) | TypeError | TypeError | “” |
| manageMenu() | TypeError | TypeError | “” |
| manageMenu(“123”,True,’123123’) | TypeError | TypeError | “” |
| manageMenu(“123”,1,1,1,1) | TypeError | TypeError | “” |

2 - 목적 : 변수 menu값에 따른 조건문 분기 확인( menu는 int로 보장됨)

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4100, True, 0) | None | None | Input이 “123”이므로 메뉴가 불렸을 때 항상 입력 형식이 맞지않아 “입력형식이 맞지 않습니다.” 출력, 맨 첫줄엔 “현재 시간을 입력해 주세요.” 출력 |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4200, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4211, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4212, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4222, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4223, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4300, True, 0) | None | None | “” |
| “”manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4311, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4312, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4313, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(43141, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4322, True, 0) | None | None | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(43142, True, 0) | KeyError | KeyError | Self.username과 self.password가 정해지지 않은 상태로 바로 menu43142()를 호출했기 때문에 KeyError 발생, 첫줄엔 “현재 시간을 입력해 주세요” 출력 |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(-123, True, 0) | -1 | -1 | 첫줄엔 “현재 시간을 입력해 주세요” 출력 |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(0, True, 0) | -1 | -1 | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4219, True, 0) | -1 | -1 | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4331, True, 0) | -1 | -1 |  |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4210, True, 0) | -1 | -1 | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(3, True, 0) | -1 | -1 | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(55759, True, 0) | -1 | -1 | “” |
| manageMenu(“123”)  Self.MI = menuInfo.menuInfo(4001, True, 0) | -1 | -1 | “” |

# Menu 클래스

* + 1. menu4100 함수

1. 부분 설계

사용자로부터 정확히 하나의 인자(input)를 입력받아야 함. 인자의 문자길이는 숫자 12개로 한다. (ex. 201910301100) (타입은 string.)

만일 인자의 개수가 다르거나 허용되는 날짜범위를 벗어나면 오류메시지를 뿜어야 한다.

형식에 위배되는 경우:

인자의 개수가 12가 아닐 경우:＂입력형식이 맞지 않습니다.“ 출력.

input에 알파벳이나 특수문자(\*,^,$,%,? 등등)가 포함되어 있는 경우: ”입력형식이 맞지 않습니다.“ 출력.

인자가 허용되는 날짜의 범위를 벗어난 경우: ”현재 날짜보다 이전 날짜입니다. 다시 입력해주세요“ 또는

”최대 날짜보다 이후 날짜입니다. 다시 입력해주세요.“ 출력.

형식에 맞게 성공적으로 입력받으면,

초기값인 self.MI = menuInfo.menuInfo(4100,False)를

self.MI.setMI(4200, self.MI.getisMember()) 으로 바꿈.

함수를 끝마치고 4200번 메뉴로 이동해야 함.

\*부가설명:

setMI 함수의 각각의 인자들은

메뉴번호(\_\_menuNum), 멤버여부(\_\_isMember)를 의미한다.

1. 검사 방법

파이참에서 main을 실행시킨 후 검사하려는 함수의 부분에서 함수의 인자 값을 다르게 입력한다.

print(MM.getMI().getmenuNum()), print(MM.getMI().getisMember())을 main함수에 넣어 각 실행마다 \_\_menuNum값과 \_\_isMember 값을 출력하여 확인할 수 있게 한다.

인자(input) 형식체크:

1 - 인자의 글자수가 12개보다 적거나 12개보다 많은 경우.

2 - 인자에 숫자 이외의 것(ex. 알파벳, 한글, 특수문자)들이 들어간 경우

3 - 인자에 지정해놓은 날짜의 범위를 벗어난 수를 입력한 경우

4 - 인자의 개수가 1개가 아닌 경우（0개, 2개이상)

\_\_menuNum(메뉴번호) 체크:

1 - self.MI.setMI(4200,self.MI.getisMember())에서 첫 번째 인자값을 다른 수로 바꾼다. 바뀐 번호의 메뉴로 이동하면 성공, 메뉴번호에 없는 값을 입력하면 오류메시지를 뿜어야 한다.

1. Test Cases 및 결과 정리

목적: 인자의 개수에 따른 결과확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상 결과 | 실제 결과 | 비고 |
| menu4100(201910301100) | 로그인 메뉴로 이동 | 로그인 메뉴로 이동 |  |
| menu4100() | “입력형식이 맞지 않습니다.” | “입력형식이 맞지 않습니다.” |  |
| menu4100(201910301100,201911102200) | “입력형식이 맞지 않습니다.” | “입력형식이 맞지 않습니다.” |  |

목적: 인자 입력형식에 따른 결과확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| menu4100(201910301100) | TypeError없음 | TypeError없음 |  |
| menu4100(201908092200) | 입력형식오류 | 입력형식오류 | “입력형식이 맞지 않습니다.” |
| menu4100(202001302200) | “최대 날짜보다 이후 날짜입니다.” | “최대 날짜보다 이후 날짜입니다.” |  |
| menu4100(20191030220) | 입력형식오류 | 입력형식오류 | 숫자적음 |
| menu4100(20191030220000) | 입력형식오류 | 입력형식오류 | 숫자많음 |
| menu4100(2019abcc2200) | 입력형식오류 | 입력형식오류 |  |
| menu4100(!&\*+=%%@?!?!) | 입력형식오류 | 입력형식오류 |  |
| menu4100(안녕하세요) | 입력형식오류 | 입력형식오류 |  |

목적: menu4100 함수안에서 setMI 함수의 인자값 바꾸기 (사용자가 입력하는 값은 아님.) 인자의 입력값에 따른 결과 확인.

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| setMI(4800,self.getisMember()) | 동작안함. | 동작안함 | 4800은 애초에 없는 메뉴번호(함수)이므로 동작하지 않는다. |
| setMI(4200,self.getisMember()) | menu4200함수로 이동 | menu4200함수로 이동 | 정상작동 |
| setMI(self.getisMember()) | AssertionError | AssertionError | 인자의 개수가 부족함 |
| setMI(4200) | AssertionErrpr | AssertionError | 인자의 개수가 부족함 |

* + 1. menu4200 함수

1. 부분 설계

숫자로 정확히 하나의 인자(input)를 입력받음.

반드시 숫자 1,2,3 중 정확히 하나를 입력해야 하며 위배 시 “입력 형식이 맞지 않습니다.” 출력.

1 - 입력받은 값이 1일 때

"ID를 입력해 주세요.(되돌아 가려면 "RESTART입력“ )" 출력

아이디는 string 타입. 알파벳 소문자, 숫자를 조합하여 4~10글자 이내여야 함.

위배시 “입력 형식에 맞지 않습니다.” 출력

입력형식이 맞으면,

MI.setMI(4211, self.MI.getisMember()) 수행.

4211번 메뉴로 이동.

2 - 입력받은 값이 2일 때

"ID를 입력해 주세요.(되돌아 가려면 "RESTART"입력)" 출력.

아이디는 string 타입. 알파벳 소문자, 숫자를 조합하여 4~10글자 이내여야 함.

위배 시 “입력 형식에 맞지 않습니다.” 출력.

입력형식이 맞으면,

self.MI.setMI(4221, self.MI.getisMember()) 수행. 4221번 메뉴로 이동해야 함.

3 - 입력받은 값이 3일때

화면을 지운 후 메인메뉴를 출력.

self.MI.setMI(4300, self.MI.getisMember()) 수행. 4300 메뉴로 이동.

RESTART를 눌렀을 때 메뉴번호에 해당하는 함수로 돌아와야 함.

4 - RESTART를 입력 했을 때

로그인 메뉴를 띄우고 메뉴 4200으로 돌아와야 함.

1. 검사 방법

파이참에서 main을 실행시킨 후 검사하려는 함수의 부분에서 인자 값을 다르게 입력한다.

print(MM.getMI().getmenuNum()), print(MM.getMI().getisMember())을 main함수에 넣어 각 실행마다 \_\_menuNum값과 \_\_isMember 값을 출력하여 확인할 수 있게 한다.

menu4200(input)

1 - 인자값이 숫자 1, 2, 3 중 하나인 경우를 확인

2 - 인자값이 1, 2, 3 중 하나가 아닌 다른 값인 경우를 확인

3 - 인자값이 숫자가 아닌 값인 경우

4 - 인의 개수가 1개가 아닌 경우

self.MI.setMI(4221, self.MI.getisMember())에서

5 - 첫 번째 인자값을 바꾼 경우 확인

6 - 인자의 개수가 맞지 않는 경우 확인

1. Test Cases 및 결과 정리

목적: 인자개수 조건(1개)확인, 인자값 입력에 대한 결과 확인.

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상 결과 | 실제 결과 | 비고 |
| menu4200(1) | 4211 메뉴로 이동 | 4211 메뉴로 이동 |  |
| menu4200(2) | 4221 메뉴로 이동 | 4221 메뉴로 이동 |  |
| menu4200(3) | 4300 메뉴로 이동 | 4300 메뉴로 이동 |  |
| menu4200(14) | 입력형식오류 | “입력형식이 맞지 않습니다.” |  |
| menu4200(abc) | 입력형식오류 | “입력형식이 맞지 않습니다.” |  |
| menu4200(!@$^#) | 입력형식오류 | “입력형식이 맞지 않습니다.” |  |
| menu4200() | 입력형식오류 | “입력형식이 맞지 않습니다.” |  |
| menu4200(1,2) | 입력형식오류 | “입력형식이 맞지 않습니다.” |  |

목적: RESTART를 눌렀을 때 그때의 메뉴번호로 돌아가는지 확인.

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| “RESTART” | 로그인 화면으로 되돌아감. | 로그인 화면으로 되돌아감. | input==’1’: |
| “RESTART” | 로그인 화면으로 되돌아감. | 로그인 화면으로 되돌아감. | intput==’2’: |

목적: MI.SetMI() 에서 인자값을 다르게 했을 때의 결과확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| self.MI.setMI(4211, self.MI.getisMember()) | 4211 메뉴로 이동 | 4211 메뉴로 이동 |  |
| self.MI.setMI(4200, self.MI.getisMember()) | 4200 메뉴로 이동 | 4200 메뉴로 이동 |  |
| self.MI.setMI(4800,self.MI.getisMember()) | 동작안함 | 동작안함 | 4800메뉴 없음 |
| self.MI.setMI(4211) | AssertionError | AssertionError | 인자개수 부족 |

* + 1. menu4211 함수

1. 부분 설계

사용자로부터 정확히 하나의 인자(input)를 입력받는다. (string 타입)

알파벳 소문자. 숫자를 조합하여 4~10글자 이내여야 함.

형식 위배 시 "입력형식에 맞지 않습니다." 출력 후 다시 입력받음.

형식이 맞으면, UserLIst에 아이디가 존재하는지 확인한다.

확인결과 아이디가 존재하지 않으면 "존재하지 않는 ID입니다. 다시 입력해주세요." 출력 후, 다시 입력받음.

아이디가 존재하면 “password를 입력해주세요. (되돌아가려면 ”RESTART“입력)” 출력 후 self.MI.setMI(4212, self.MI.getisMember()) 수행.

4212 메뉴로 이동.

1. 검사 방법

파이참에서 main을 실행시킨 후 검사하려는 함수의 부분에서 함수의 인자 값을 다르게 입력한다.

print(MM.getMI().getmenuNum()), print(MM.getMI().getisMember())을 main함수에 넣어 각 실행마다 \_\_menuNum값과 \_\_isMember 값을 출력하여 확인할 수 있게 한다.

1 - 존재하는 ID를 입력한 경우

2 - 존재하지 않는 ID를 입력한 경우

3 - 입력형식을 지키지 않은 경우

4 - setMI()의 인자값을 다르게 설정한 경우

1. Test Cases 및 결과 정리

목적: 2.3.2 검사방법의 3가지 경우에 대한 결과확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| menu4211(“us5e1r”) | “Password를 입력해 주세요.(되돌아가려면 "RESTART"입력)” | “Password를 입력해 주세요.(되돌아가려면 "RESTART"입력)” |  |
| menu4211(“하정수”) | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |
| menu4211(“uqw123”) | “존재하지 않는 ID입니다. 다시 입력해주세요” | “존재하지 않는 ID입니다. 다시 입력해주세요” |  |
| menu4211(“a”) | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |

목적: MI.setMI() 에서 인자값을 다르게 했을 때의 결과확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| MI.setMI(4212,self.MI.getisMEmber()) | 메뉴 4212로 이동 | 메뉴 4212로 이동 |  |
| MI.setMI(4200,self.MI.getisMEmber()) | 메뉴 4200으로 이동 | 메뉴 4200으로 이동 |  |
| MI.setMI(4800,self.MI.getisMember()) | 작동안함 | 작동안함 |  |
| MI.setMI(1212) | AssertionError | AssertionError |  |

* + 1. menu4212 함수

1. 부분 설계

사용자로부터 하나의 인자(input)를 입력받음.

string 타입으로 알파벳 소문자, 숫자의 조합으로 4~10 글자여야 한다.

입력형식 위배 시, “입력 형식에 맞지 않습니다.” 출력.

입력형식이 맞으면, UserList에 입력했던 ID와 password 일치 여부를 확인.

맞지 않으면, “일치하지 않는 password입니다.” 출력.

맞으면, 화면을 지우고 메인 메뉴를 출력, self.MI.setMI(4300, True)를 수행함.

메뉴 4300으로 이동.

1. 검사 방법

print(MM.getMI().getmenuNum()), print(MM.getMI().getisMember())을 main함수에 넣어 각 실행마다 \_\_menuNum값과 \_\_isMember 값을 출력하여 확인할 수 있게 한다.

1 - 패스워드가 아이디와 성립하는 경우

2 - 패스워드가 아이디와 성립하지 않는 경우

3 - 패스워드 입력형식을 어겼을 경우

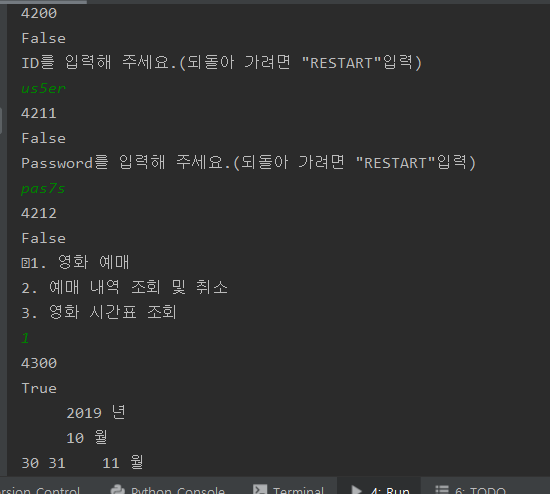
4 - setMI()함수의 인자값 바꾸는 경우 확인.

1. Test Cases 및 결과 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| 423p0as66s | 4300 메뉴로 이동 | 4300 메뉴로 이동 |  |
| 123456 | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |
| pas4s | “일치하지않는 passwo rd입니다. 다시 입력해주세요.” | “일치하지않는 passwo rd입니다. 다시 입력해주세요.” |  |
| !2안녕하세요 | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |

목적: SetMI()함수의 인자값을 바꿈에 따른 결과확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| MI.setMI(4300,True) | 메뉴4300으로 이동 | 메뉴 4300으로 이동 |  |
| MI.setMI(4800.True) | 동작안함 | 동작안함 |  |
| MI.setMI(4300, True) | 4300 False 가 출력 | 4300 False 가 출력 |  |
| MI.setMI() | AssertionError | AssertionError |  |
| MI.setMI(4200,True) | 메뉴 4200으로 이동 | 메뉴 4200으로 이동 |  |



3번째 칸과 같이 실행하면, 아래 사진과 같이

-> 4300

True 가 출력됨을 확인할 수 있었다.

* + 1. menu4221 함수

1. 부분 설계

사용자로부터 하나의 인자(input)을 입력받음.

input은 string 타입으로 알파벳 소문자. 숫자의 조합으로 4~10 글자여야 한다.

입력형식 위배 시, “입력형식에 맞지 않습니다.” 출력.

입력형식이 맞으면 input을 UserList에 이미 같은 ID가 존재하는지 확인한다.

중복되는 아이디가 있으면, “이미 존재하는 ID입니다, 다시 입력해주세요.”를 출력, 다시 입력받는다.

중복되는 아이디가 없으면, self.userName 에 input을 저장한 후

“Password를 입력해주세요. (되돌아가려면 ”RESTART“ 입력)” 을 출력한다.

그 후 self.MI.setMI(4222, self.MI.getisMember())를 수행.

“RESTART”를 눌렀을 때 로그인 메뉴로 돌아와야 함.

1. 검사 방법

1 - 사용자가 인자에 입력형식과 맞지 않는 값을 입력한 경우

2 - 인자의 개수가 1개가 아닐 경우

2 - 사용자가 list에 없는 ID를 입력한 경우

3 - 사용자가 list에 있는 ID를 입력한 경우

4 - “RESTART”를 입력했을 때의 결과 확인

1. Test Cases 및 결과 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| menu4221(“jkjk2013”) | “Password 를 입력해 주세요.(되돌아가려면 ”RESTART“입력)” | “Password 를 입력해 주세요.(되돌아가려면 ”RESTART“입력)” |  |
| menu4221(“u18se33r”) | "이미 존재하는 ID입니다. 다시 입력해 주세요." | "이미 존재하는 ID입니다. 다시 입력해 주세요." |  |
| menu4221(“1234”) | “입력 형식에 맞지 않습니다.” | “입력 형식에 맞지 않습니다.” |  |
| menu4221(“sqwqa54564qwf”) | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다. |  |
| menu4221(“asd123”,“123asd”) | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |
| “RESTART” | 로그인 메뉴 보여줌 | 로그인 메뉴 보여줌 |  |

* + 1. menu4222 함수

1. 부분 설계

사용자로부터 하나의 인자(input)을 입력받음.

input은 string 타입으로 알파벳 소문자. 숫자의 조합으로 4~10 글자여야 한다.

입력형식 위배 시, “입력 형식에 맞지 않습니다.”를 출력.

입력형식이 맞으면, self.password 에 input을 저장하고

“등록할 카드 번호를 입력해주세요. (되돌아가려면 ”RESTART“ 입력)” 출력.

self.MI.setMI(4223,self.MI.getisMember()) 수행. 메뉴 4223으로 이동.

1. 검사 방법

1 - 사용자가 인자로 입력형식에 맞지 않는 값을 입력했을 경우

2 - 인자의 개수가 다른 경우

3 - MI.setMI()의 인자값이 다를 경우

1. Test Cases 및 결과 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| menu4222() | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |
| menu4222(“matt1235”) | menu4223으로 이동 | menu4223으로 이동 |  |
| menu4222(“asd”) | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |
| menu4422(“asd123?”) | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |
| MI.setMI(4223,self.MI.getisMember()) | munu4223으로 이동 | menu4223으로 이동 |  |
| MI.setMI(4223) | AssertionError | AssertionError |  |
| MI.setMI(4800,self.MI.getisMember()) | 작동안함 | 잗동안함 |  |

* + 1. menu4223 함수

1. 부분 설계

사용자로부터 하나의 인자값을 입력받는다. 카드번호의 숫자는 12자리 숫자이며 공백이 있어서는 안된다.

입력형식에 위배되는 경우, “입력 형식에 맞지 않습니다.”를 출력.

입력형식에 맞는 경우,

cardlist에 있는 카드번호와 비교하여, 이미 등록된 카드 번호이면

“이미 등록된 카드 번호입니다. 다시 입력해주세요.” 출력. 다시 입력받음.

존재하지 않는 카드 번호이면

“존재하지 않는 카드번호입니다. 다시 입력해주세요” 출력. 다시 입력받음.

유효한 카드 번호이면, userlist를 업데이트함. 그 후 로그인 창을 띄우고 다시 menu4200으로 돌아감.

1. 검사 방법

1 - 사용자가 입력한 인자가 입력형식에 위배되는 경우

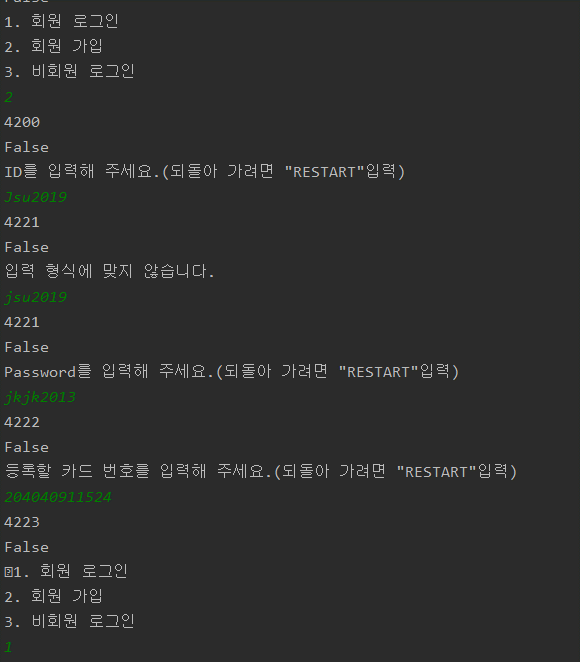
2 - 사용자가 이미 등록된 카드번호를 입력한 경우

3 - 사용자가 없는 카드번호를 입력한 경우

4 - menu4221, menu4222, menu4223을 통해 회원가입을 한 사용자의 회원로그인 가능 여부 확인 (업데이트 확인)

1. Test Cases 및 결과 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | 예상결과 | 실제결과 | 비고 |
| menu4223(“204040911524”) | 성공후 menu4200 실행 | 성공후 menu4200 실행 |  |
| menu4223(“204026171508”) | “이미 등록된 카드번호입니다.” | “이미 등록된 카드번호입니다.” |  |
| menu4223(“asdqwe”) | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |
| menu4223(“2040 456”) | “입력형식에 맞지 않습니다.” | “입력형식에 맞지 않습니다.” |  |
| menu4223(“209144557788”) | “존재하지 않는 카드번호입니다.” | “존재하지 않는 카드번호입니다.” |  |



-회원가입을 통해 아이디 jsu2019, 비밀번호 jkjk2013을 만들었음.

카드번호는 204040911524

그 후 로그인 화면으로 돌아아서 1번(회원로그인)을 선택함.

* + 1. \_\_init\_\_ 함수

1. 부분 설계

이 클래스에서 사용될 FileManager, TypeChecker() 객체 및 여러 변수들을 선언 및 초기화함.

1. 검사 방법

단순히 선언 및 초기화만을 수행하기 때문에 검사하지 않음.

1. Test Cases 및 결과 정리

위와 같음.

* + 1. menu4300 함수

인자로 전달받은 input의 자료형을 str로 assert문을 걸어준 상태.

1. 부분 설계

하나의 인자만을 받아야 함. 아니라면 TypeError를 출력함

Input은 str형 이어야 하고 아니라면 AssertionError를 출력함.

Input과 self.MI.getMember()의 값에 따라 다음 중 하나의 기능을 수행함.

1. “입력형식에 맞지 않습니다.” 을 출력함.
2. 현재 날짜로부터 10일 후 까지의 날짜를 달력 형식으로 출력하고, “상영을 원하는 날짜를 8자리로 입력해 주세요.” 을 출력함.
   1. self.MI.getMember() == True (회원이면)

회원의 현재 예매 내역을 출력하고, “취소하시려는 영화의 예매 코드를 입력해 주세요.(취소하지 않고 메인 메뉴로 돌아가시려면”BACK”을 입력해 주세요.)” 을 출력함.

self.MI.setMI 함수를 통해 MI.menuNum 값만 4322로 바꿔줌.

3.2 self.MI.getMember() == False (비회원이면)

“예매 코드를 입력해 주세요.” 을 출력함.

self.MI.setMI 함수를 통해 MI.menuNum 값만 43212로 바꿔줌.

1. 검사 방법

전체 프로그램을 실행시킨 뒤, menu4300에서의 입력을 input과 self.MI.getMember()을 바꿔가며 조건 분기를 검사함.

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 1개) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu4300() | TypeError | TypeError |  |
| Menu4300(“123”) | TypeError없음 | TypeError없음 |  |
| Menu4300(“123”,”123”) | TypeError | TypeError |  |
| Menu4300(1,1,1,1) | TypeError | TypeError |  |
| Menu4300(1,2,3,4,5,6,7) | TypeError | TypeError |  |

목적 : 인자 input 조건(str형) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu4300(“1”) | AssertionError없음 | AssertionError없음 |  |
| Menu4300(0.5) | AssertionError | AssertionError |  |
| Menu4300(-10023) | AssertionError | AssertionError |  |
| Menu4300(“1.asd”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음. |
| Menu4300(“하이”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음. |
| Menu4300(123) | AssertionError | AssertionError |  |
| Menu4300(0) | AssertionError | AssertionError |  |
| Menu4312(20191111) | AssertionError | AssertionError |  |

목적 : input과 self.MI.getMember() 에 따른 조건 분기 확인.

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input=”0”  Self.MI.getMember()  =True | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”-123”  Self.MI.getMember()  =True | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”3”  Self.MI.getMember()  =True | 영화 시간표 출력 | 부분 설계 1번 결과 | 원래 기능이 있지만 아직 미구현 |
| Input=”abc”  Self.MI.getMember()  =True | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”0.1”  Self.MI.getMember()  =True | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”ㅎㅇ”  Self.MI.getMember()  =True | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”0”  Self.MI.getMember()  =False | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”-123”  Self.MI.getMember()  =False | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”3”  Self.MI.getMember()  =False | 영화 시간표 출력 | 부분 설계 1번 결과 | 원래 기능이 있지만 아직 미구현 |
| Input=”abc”  Self.MI.getMember()  =False | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”0.1”  Self.MI.getMember()  =False | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”ㅎㅇ”  Self.MI.getMember()  =False | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input=”1  Self.MI.getMember()  =True | 부분 설계 2 결과 | 부분 설계 2번 결과 | 회원일 때 |
| Input=”2”  Self.MI.getMember()  =True | 부분 설계 3.1 결과 | 부분 설계 3.1번 결과 | 회원일 때 |
| Input=”1  Self.MI.getMember()  =False | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 비회원일 때 |
| Input=”2”  Self.MI.getMember()  =False | 부분 설계 3.2번 결과 | 부분 설계 3.2번 결과 | 비회원일 때 |

* + 1. menu4311 함수

인자로 전달받은 input의 자료형을 str로 assert문을 걸어준 상태.

함수에서 사용 될 self.\_\_now\_time 변수를 TypeChecker 클래스의 time\_check 함수와 자료형 str로 assert문을 걸어준 상태.

self.time\_check 함수와 self.date\_check 함수는 인자값이 8자리 or 12자리의 날짜 시간 정보인지 True, False값으로 return 해주는 함수임.

self.\_\_FM.day\_movielist 함수는 8자리의 날짜 및 4자리의 시간 정보를 인자로 받아 그 시간 이후의 영화 정보를 리스트로 return 하는 함수임.

self.printday\_movie 함수는 전달 받은 영화 정보 리스트의 값을 특정 형태로 모두 출력해주고 True를 return, 만약 빈 리스트라면 False를 return 해주는 함수임.

1. 부분 설계

하나의 인자만을 받아야 함. 아니라면 TypeError를 출력함

Input은 str형 이어야 하고, self.\_\_now\_time은 str형이며 time\_check 함수가 True값이 나오게 해야함(8자리의 날짜정보 여야함). 아니라면 AssertionError를 출력함.

인자값 input과 변수 self.\_\_now\_time에 따라 다음 중 하나의 기능을 수행함.

1. “입력 형식에 맞지 않습니다.” 을 출력함.
2. “현재 날짜보다 이전 날짜입니다. 다시 입력해 주세요.” 을 출력함.
3. “상영중인 영화가 없습니다. 다시 입력해 주세요.” 을 출력함.
4. 해당 날짜(input), self.\_\_now\_time 이후의 시간대의 영화 목록을 출력하고, “예매할 영화를 선택하세요.(숫자.영화명 입력)” 을 출력하고, self.MI.setMI 함수를 통해 MI.menuNum 값만 4312로 바꿔줌.
5. “최대 날짜보다 이후 날짜입니다. 다시 입력해 주세요.” 을 출력함.
6. 검사 방법

FileManager, TypeChecker, menuInfo 클래스를 모두 콘솔에 불러오고, Menu 객체를 새로 생성하여 menu4311 함수를 직접 호출해 검사함. self.\_\_now\_time 값을 직접 바꿔줌.(Menu의 생성자에서 패키지 참조하는 부분을 수정함)

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 1개) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu4311() | TypeError | TypeError |  |
| Menu4311(“123”) | TypeError없음 | TypeError없음 | 하지만 AssertionError |
| Menu4311(“123”,”123”) | TypeError | TypeError |  |
| Menu4311(1,1,1,1) | TypeError | TypeError |  |
| Menu4311(1,2,3,4,5,6,7) | TypeError | TypeError |  |

목적 : 인자 input ,변수 self.\_\_now\_time 조건 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| self.\_\_now\_time = “201911111200”  Menu4311(“20191230”) | AssertionError없음 | AssertionError없음 |  |
| self.\_\_now\_time = “201--913111200”  Menu4311(“123”) | AssertionError | AssertionError | 특수문자x |
| self.\_\_now\_time = “201913111200”  Menu4311(“20191230”) | AssertionError | AssertionError | 13월x |
| self.\_\_now\_time = “000012111200”  Menu4311(“20191230”) | AssertionError없음 | AssertionError없음 |  |
| self.\_\_now\_time = 123  Menu4311(“20191230”) | AssertionError | AssertionError | Now\_time은 Int형x |
| self.\_\_now\_time = “000012111200”  Menu4311(123) | AssertionError | AssertionError | Input은 Int형x |
| self.\_\_now\_time = “2019121112000”  Menu4311(“20191230”) | AssertionError | AssertionError | Now\_time의길이는 무조건12자리 |
| self.\_\_now\_time = “201912111200”  Menu4311(“201912300”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 |

목적 : self.\_\_now\_time, input, 해당 날짜의 영화 리스트에 따른 조건 분기 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재하지 않음.  즉, self.day\_movielist==[]  Menu4311(“20191110”) | 부분설계의 2번 결과 | 부분설계의 2번 결과 | 이전 날짜임 |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재함.  즉, self.day\_movielist!=[]  Menu4311(“20191110”) | 부분설계의 2번 결과 | 부분설계의 2번 결과 | 이전 날짜임 |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재함.  즉, self.day\_movielist!=[]  Menu4311(“20191230”) | 부분 설계의 5번 결과 | 부분 설계의 5번 결과 | 최대 날짜보다 이후임 |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재하지 않음.  즉, self.day\_movielist==[]  Menu4311(“20191230”) | 부분 설계의 3번 결과 | 부분 설계의 3번 결과 | 영화가 없음 |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재하지 않음.  즉, self.day\_movielist==[]  Menu4311(“20191111”) | 부분 설계의 3번 결과 | 부분 설계의 3번 결과 | 영화가 없음 |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재함.  즉, self.day\_movielist!=[]  Menu4311(“20191111”) | 부분 설계의 4번 결과 | 부분 설계의 4번 결과 | 이전 날짜임 |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재하지 않음.  즉, self.day\_movielist==[]  Menu4311(“20181229”) | 부분 설계의 2번 결과 | 부분 설계의 2번 결과 | 이전 날짜임 |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  즉, self.day\_movielist!=[]  Menu4311(“20181229”) | 부분 설계의 2번 결과 | 부분 설계의 2번 결과 | 이전 날짜임 |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재하지 않음.  즉, self.day\_movielist==[]  Menu4311(“20190101”) | 부분 설계의 3번 결과 | 부분 설계의 3번 결과 | 영화가 없음 |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  즉, self.day\_movielist!=[]  Menu4311(“20190101”) | 부분 설계의 4번 결과 | 부분 설계의 4번 결과 |  |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  즉, self.day\_movielist!=[]  Menu4311(“201901010”) | 부분 설계의 1번 결과 | 부분 설계의 1번 결과 | Input은 무조건8자리 |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  즉, self.day\_movielist!=[]  Menu4311(“2019-01-01”) | 부분 설계의 1번 결과 | 부분 설계의 1번 결과 | Input은 특수문자x |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  즉, self.day\_movielist!=[]  Menu4311(“abcdefg”) | 부분 설계의 1번 결과 | 부분 설계의 1번 결과 | Input은 숫자만 가능 |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  즉, self.day\_movielist!=[]  Menu4311(“0.123456”) | 부분 설계의 1번 결과 | 부분 설계의 1번 결과 | Input은 특수문자x |

* + 1. menu4312 함수

인자로 전달받은 input의 자료형을 str로 assert문을 걸어준 상태.

self.print.seat 함수는 입력 받은 영화 제목이 존재하는 영화 제목인지 True, False로 return 해주는 함수임. - True하면 좌석표를 출력함.

1. 부분 설계

하나의 인자만을 받아야 함. 아니라면 TypeError를 출력함

Input은 str형 이어야 함. 아니라면 AssertionError를 출력함.

인자값 input에 따라 다음 중 하나의 기능을 수행함.

1. “입력 형식에 맞지 않습니다.” 을 출력.
2. “존재하지 않는 영화입니다. 다시 입력해 주세요.” 을 출력
3. 좌석표와 “예약할 좌석을 고르십시오.” 을 출력하고, self.MI.setMI 함수를 통해 MI.menuNum 값만 4313로 바꿔줌.
4. 검사 방법

FileManager, TypeChecker, menuInfo 클래스를 모두 콘솔에 불러오고, Menu 객체를 새로 생성하여 menu4312 함수를 직접 호출하고, self.print\_seat 함수의 return 값을 강제로 True,False로 수정해가며 검사함.(Menu의 생성자에서 패키지 참조하는 부분을 수정함)

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 1개) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu4312() | TypeError | TypeError |  |
| Menu4312(“123”) | TypeError없음 | TypeError없음 | 하지만 AssertionError |
| Menu4312(“123”,”123”) | TypeError | TypeError |  |
| Menu4312(1,1,1,1) | TypeError | TypeError |  |
| Menu4312(1,2,3,4,5,6,7) | TypeError | TypeError |  |

목적 : 인자 input 조건 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu4312(“abc”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 |
| Menu4312(0.5) | AssertionError | AssertionError |  |
| Menu4312(-10023) | AssertionError | AssertionError |  |
| Menu4312(“1.asd”) | AssertionError없음 | AssertionError없음 | 하지만 존재하지 않는 영화 제목임. |
| Menu4312(“하이”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 |
| Menu4312(123) | AssertionError | AssertionError |  |
| Menu4312(0) | AssertionError | AssertionError |  |
| Menu4312(20191111) | AssertionError | AssertionError |  |

목적 : 인자 input 과 self.print\_seat 함수의 return 값에 따른 조건 분기 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Self.print\_seat() =True  Menu4312(“abc”) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Self.print\_seat() =True  Menu4312(“1..movie”) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Self.print\_seat() =True  Menu4312(“ 2.movie5”) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Self.print\_seat() =True  Menu4312(“1.1234”) | 부분 설계 3번 결과 | 부분 설계 3번 결과 | 영화 제목은 숫자도 가능 |
| Self.print\_seat() =True  Menu4312(“1-2.movie”) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Self.print\_seat() =True  Menu4312(“1234.movie”) | 부분 설계 3번 결과 | 부분 설계 3번 결과 |  |
| Self.print\_seat() =True  Menu4312(“1234.하이”) | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 영화 제목은 한글x |
| Self.print\_seat() =True  Menu4312(“1. “) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Self.print\_seat() =True  Menu4312(“1.a a“) | 부분 설계 3번 결과 | 부분 설계 3번 결과 |  |
| Self.print\_seat() =True  Menu4312(“1. a“) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Self.print\_seat() =True  Menu4312(“1.a a“) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Self.print\_seat() =True  Menu4312(“1movie“) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Self.print\_seat() =True  Menu4312(“1.movieaaaaaaaaaaaaaaaaaaa“) | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Self.print\_seat() =False  Menu4312(“1.1234”) | 부분 설계 3번 결과 | 부분 설계 3번 결과 |  |
| Self.print\_seat() =False  Menu4312(“1234.movie”) | 부분 설계 3번 결과 | 부분 설계 3번 결과 |  |
| Self.print\_seat() =False  Menu4312(“1.a a“) | 부분 설계 3번 결과 | 부분 설계 3번 결과 |  |

* + 1. menu4313 함수

인자로 전달받은 input의 자료형을 str로 assert문을 걸어준 상태.

self.\_\_TC.checkSeatsList 함수는 input을 인자로 받아 좌석의 입력 형식인지 True,False 로 return 해주는 함수임.

self.count\_seat 함수는 좌석 리스트 인자로 받아 몇 개의 좌석이 있는지 return 해주는 함수임.

1. 부분 설계

하나의 인자만을 받아야 함. 아니라면 TypeError를 출력함

Input은 str형 이어야 한다. 아니라면 AssertionError를 출력함.

Input에 따라 다음과 같은 기능 중 하나를 수행한다.

1. “입력 형식에 맞지 않습니다.”을 출력함.
2. 예약할 수 없는 좌석입니다. 다시 입력해 주세요.”을 출력함.

3-1. “결제하실 금액은 총 (좌석 수x7000)원 입니다.” 을 출력함.

3-2. “결제하실 금액은 총 (좌석 수x10000)원 입니다.” 을 출력함.

4-1. “회원님의 마일리지 잔액은 (해당 회원의 마일리지)원 입니다. 얼마를 사용하시겠습니까?” 을 출력함.

self.MI.setMI 함수를 통해 MI.menuNum 값만 43141로 바꿔줌.

4-2. “결제하실 카드번호를 입력해 주세요.” 을 출력함.

self.MI.setMI 함수를 통해 MI.menuNum 값만 43142로 바꿔줌.

1. 검사 방법
2. FileManager, TypeChecker, menuInfo 클래스를 모두 콘솔에 불러오고, Menu 객체를 새로 생성하여 menu4312 함수를 직접 호출함.(Menu의 생성자에서 패키지 참조하는 부분을 수정함)
3. 전체 프로그램을 실행시키고, 좌석 선택 메뉴에서 입력->”BACK”입력을 반복함.(파일 리스트값들을 일일히 수정하기에 한계가 있음.)

self.username, self.password, 현재 선택된 영화, 회원 비회원 여부(self.getMI().getisMember()), UserList.csv, MovieList.csv의 데이터를 바꿔가며 검사함.

함수의 return 값을 강제로 바꿔가며 검사함.

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 1개) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu4313() | TypeError | TypeError |  |
| Menu4313(“123”) | TypeError없음 | TypeError없음 | 하지만 AssertionError |
| Menu4313(“123”,”123”) | TypeError | TypeError |  |
| Menu4313(1,1,1,1) | TypeError | TypeError |  |
| Menu4313(1,2,3,4,5,6,7) | TypeError | TypeError |  |

목적 : 인자 input 조건 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu4313(“abc”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 |
| Menu4313(0.5) | AssertionError | AssertionError |  |
| Menu4313(-10023) | AssertionError | AssertionError |  |
| Menu4313(“1.asd”) | AssertionError없음 | AssertionError없음 | 하지만 존재하지 않는 영화 제목임. |
| Menu4313(“하이”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 |
| Menu4313(123) | AssertionError | AssertionError |  |
| Menu4313(0) | AssertionError | AssertionError |  |
| Menu4313(20191111) | AssertionError | AssertionError |  |

목적 : self.username, self.password, 현재 선택된 영화의 좌석표, UserList.csv,MovieList.csv의 데이터와 input값에 따른 조건 분기 확인.

UserList.csv



MovieList.csv ( 시작시간 12시 이후 )



좌석표 ( ■ = 예약된 좌석, □ = 예약 가능 좌석 )

---------Screen---------

1 2 3 4 5 6 7 8 9

A ■ □ □ □ □ □ □ □ □

B □ □ □ □ □ □ □ □ □

C □ □ □ □ □ □ □ □ □

D □ □ □ ■ □ □ □ □ □

E □ □ □ □ □ □ □ □ □

F □ □ □ □ □ □ □ □ □

G □ □ □ □ □ □ □ □ □

H ■ □ □ □ □ □ □ □ □

I □ □ □ □ □ □ □ □ □

J □ □ □ □ □ □ □ □ □

self.getMI().getisMember() = True -> 회원 로그인 상태

* 입력 형식에 중점

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input = “A1” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 이미 예약된 좌석 |
| Input = “A5~A1” | 부분 설계 1번 결과 | IndexError | self.\_\_FM.seats\_to\_list 함수 수정 필요 ->수정 후 확인o |
| Input = “A1~A5” | 부분 설계 2번 결과 | 부분 설계 1번 결과 | self.\_\_TC.checkyourseat 함수 수정 필요 ->수정 후 확인o |
| Input = “A1,A5” | 부분 설계 2번 결과 | 부분 설계 1번 결과 | self.\_\_TC.checkyourseat 함수 수정 필요 ->수정 후 확인o |
| Input = “123” | 부분 설계 1번 결과 | 부분 설계 1번 결과 |  |
| Input = “A2” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 좌석 당 10000원으로 계산 |
| Input = “J11” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 좌석표를 넘어감 |
| Input = “A-1” | 부분 설계 1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | self.\_\_TC.checkyourseat 함수 수정 필요 ->수정 후 확인o |
| Input = “A0” | 부분 설계 1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | self.\_\_TC.checkyourseat 함수 수정 필요 ->수정 후 확인o |
| Input = “Z1” | 부분 설계 2번 결과 | IndexError | 좌석표를 넘어감 - self.count\_seat 함수 수정 필요 -> 수정 후 확인o |
| Input = “A2~A5” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 1번 결과 | self.\_\_TC.checkyourseat 함수 수정 필요 -> 수정 후 확인o |
| Input = “A2,A5,H5” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 1번 결과 | self.\_\_TC.checkyourseat 함수 수정 필요 -> 수정 후 확인o |
| Input = “A2,A5,H5” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 1번 결과 | self.\_\_TC.checkyourseat 함수 수정 필요 -> 수정 후 확인o |

* 예약 가능 좌석 확인에 중점( 입력 형식이 맞게 들어온 경우 )

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input = “A1” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 이미 예약된 좌석임 |
| Input = “A1~A5” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 이미 예약된 좌석으로 시작하면 안됨. |
| Input = “D1~D4” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 이미 예약된 좌석으로 끝나면 안됨. |
| Input = “D3~D5” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 이미 예약된 좌석 중간에 있으면 안됨. |
| Input = “A2,A4,H3,A1” | 부분 설계 2번 결과 | 부분 설계 2번 결과 |  |
| Input = “D1~D3” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 예약 가능 |
| Input = “A2,A3,A4” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 예약 가능 |
| Input = “J1” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 예약 가능 |

UserList.csv



MovieList.csv ( 시작시간 12시 이후 )



좌석표 ( ■ = 예약된 좌석, □ = 예약 가능 좌석 )

---------Screen---------

1 2 3 4 5 6 7 8 9

A ■ □ □ □ □ □ □ □ □

B □ □ □ □ □ □ □ □ □

C □ □ □ □ □ □ □ □ □

D □ □ □ ■ □ □ □ □ □

E □ □ □ □ □ □ □ □ □

F □ □ □ □ □ □ □ □ □

G □ □ □ □ □ □ □ □ □

H ■ □ □ □ □ □ □ □ □

I □ □ □ □ □ □ □ □ □

J □ □ □ □ □ □ □ □ □

self.getMI().getisMember() = True -> 회원 로그인 상태

* 마일리지 값의 범위에 중점(예약 가능한 좌석을 고른 경우)

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input = “A2” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 |  |
| Input = “A2~A5” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 |  |
| Input = “A2,A5,H5” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 |  |

UserList.csv



MovieList.csv ( 시작시간 12시 이후 )



좌석표 ( ■ = 예약된 좌석, □ = 예약 가능 좌석 )

---------Screen---------

1 2 3 4 5 6 7 8 9

A ■ □ □ □ □ □ □ □ □

B □ □ □ □ □ □ □ □ □

C □ □ □ □ □ □ □ □ □

D □ □ □ ■ □ □ □ □ □

E □ □ □ □ □ □ □ □ □

F □ □ □ □ □ □ □ □ □

G □ □ □ □ □ □ □ □ □

H ■ □ □ □ □ □ □ □ □

I □ □ □ □ □ □ □ □ □

J □ □ □ □ □ □ □ □ □

self.getMI().getisMember() = True -> 회원 로그인 상태

* 마일리지 값의 범위에 중점

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input = “A2” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 마일리지가 음수인 경우 따로 예외처리x |

UserList.csv



MovieList.csv ( 시작시간 12시 이후 )



좌석표 ( ■ = 예약된 좌석, □ = 예약 가능 좌석 )

---------Screen---------

1 2 3 4 5 6 7 8 9

A ■ □ □ □ □ □ □ □ □

B □ □ □ □ □ □ □ □ □

C □ □ □ □ □ □ □ □ □

D □ □ □ ■ □ □ □ □ □

E □ □ □ □ □ □ □ □ □

F □ □ □ □ □ □ □ □ □

G □ □ □ □ □ □ □ □ □

H ■ □ □ □ □ □ □ □ □

I □ □ □ □ □ □ □ □ □

J □ □ □ □ □ □ □ □ □

self.getMI().getisMember() = True -> 회원 로그인 상태

* 마일리지 값의 범위에 중점

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input = “A2” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 |  |
| Input = “A2~A5” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 |  |
| Input = “A2,A5,H5” | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-2번 결과,  부분 설계 4-1번 결과 |  |

UserList.csv



MovieList.csv ( 시작시간 12시 이전 )



좌석표 ( ■ = 예약된 좌석, □ = 예약 가능 좌석 )

---------Screen---------

1 2 3 4 5 6 7 8 9

A □ □ □ □ □ □ □ □ □

B □ □ □ □ □ □ □ □ □

C □ □ □ □ □ □ □ □ □

D □ □ □ □ □ □ □ □ □

E □ □ □ □ □ □ □ □ □

F □ □ □ □ □ □ □ □ □

G □ □ □ □ □ □ □ □ □

H □ □ □ □ □ □ □ □ □

I □ □ □ □ □ □ □ □ □

J □ □ □ □ □ □ □ □ □

self.getMI().getisMember() = True -> 회원 로그인 상태

* 영화 시작 시간에 중점

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input = “A2” | 부분 설계 3-1번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-1번 결과,  부분 설계 4-1번 결과 | 좌석 당 7000원으로 계산 |
| Input = “H3” | 부분 설계 3-1번 결과,  부분 설계 4-1번 결과 | 부분 설계 3-1번 결과,  부분 설계 4-1번 결과 | 좌석 당 7000원으로 계산 |

UserList.csv



MovieList.csv ( 시작시간 12시 이전 )



좌석표 ( ■ = 예약된 좌석, □ = 예약 가능 좌석 )

---------Screen---------

1 2 3 4 5 6 7 8 9

A □ □ □ □ □ □ □ □ □

B □ □ □ □ □ □ □ □ □

C □ □ □ □ □ □ □ □ □

D □ □ □ □ □ □ □ □ □

E □ □ □ □ □ □ □ □ □

F □ □ □ □ □ □ □ □ □

G □ □ □ □ □ □ □ □ □

H □ □ □ □ □ □ □ □ □

I □ □ □ □ □ □ □ □ □

J □ □ □ □ □ □ □ □ □

self.getMI().getisMember() = False-> 비회원 로그인 상태

* 비회원 상태 시 수행에 중점

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input = “A2” | 부분 설계 3-1번 결과,  부분 설계 4-2번 결과 | 부분 설계 3-1번 결과,  부분 설계 4-2번 결과 |  |
| Input = “A2~A5” | 부분 설계 3-1번 결과,  부분 설계 4-2번 결과 | 부분 설계 3-1번 결과,  부분 설계 4-2번 결과 |  |
| Input = “A2,A5,H5” | 부분 설계 3-1번 결과,  부분 설계 4-2번 결과 | 부분 설계 3-1번 결과,  부분 설계 4-2번 결과 |  |

목적 : UserList.csv 파일의 데이터 형식 확인.



예상 결과 : 부분 설계 3-2번 까지 진행 후, ValueError

실제 결과 : 부분 설계 3-2번 까지 진행 후, ValueError

비고 : 마일리지 값이 문자열이면x

목적 : MovieList.csv 파일의 좌석 정보 데이터 형식 확인.



self.getMI().getisMember() = True-> 회원 로그인 상태

Input = “I2”

예상 결과 : 부분 설계 3-2번 결과, 부분 설계 4-1번 결과

실제 결과 : 부분 설계 3-2번 결과, 부분 설계 4-1번 결과

비고 : 좌석이 ‘0’ 이면 빈 좌석, 아니라면 예약된 좌석으로 인식함.

* + 1. menu43141 함수

인자로 전달받은 input의 자료형을 str로 assert문을 걸어준 상태.

1. 부분 설계

하나의 인자만을 받아야 함. 아니라면 TypeError를 출력함.

Input은 str형 이어야 한다. 아니라면 AssertionError를 출력함.

인자값 input에 따라 다음 중 하나의 기능을 수행함.

1. “입력 형식에 맞지 않습니다.” 을 출력.
2. “초과된 마일리지 금액입니다. 다시 입력해 주세요.” 을 출력
3. “나머지 금액은 (남은 마일리지 – 입력값)원 입니다. 등록된 카드로 결제하겠습니다.” 을 1초간 출력함.

“결제가 완료되었습니다. 예매 코드 : (예매 코드)” 을 출력함.

Self.\_\_FM.getuser 함수와 self.\_\_FM.bookmovie 함수, self.\_\_FM.savefile 함수를 이용해

UserList, ReservationList, MovieList 파일을 update 함.

출력들을 모두 지우고(clear), 메인 메뉴를 출력함.

self.MI.setMI 함수를 통해 MI.menuNum 값만 4300로 바꿔줌.

1. 검사 방법
2. FileManager, TypeChecker, menuInfo 클래스를 모두 콘솔에 불러오고, Menu 객체를 새로 생성하여 menu4312 함수를 직접 호출함.(Menu의 생성자에서 패키지 참조하는 부분을 수정함)
3. 전체 프로그램을 실행시키고, 마일리지 입력 메뉴에서 입력->”BACK”입력을 반복함.

Mileage, 현재 선택된 영화의 데이터를 바꿔가며 검사함.

코드를 추가해 내부 변수들의 값을 바꿈

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 1개) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu43141() | TypeError | TypeError |  |
| Menu43141(“123”) | TypeError없음 | TypeError없음 | 하지만 AssertionError |
| Menu43141(“123”,”123”) | TypeError | TypeError |  |
| Menu43141(1,1,1,1) | TypeError | TypeError |  |
| Menu43141(1,2,3,4,5,6,7) | TypeError | TypeError |  |

목적 : 인자 input 조건 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu43141(“abc-1-”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 |
| Menu43141(0.5) | AssertionError | AssertionError |  |
| Menu43141(-10023) | AssertionError | AssertionError |  |
| Menu43141(“1234”) | AssertionError없음 | AssertionError없음 |  |
| Menu43141(“하이”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 |
| Menu43141(123) | AssertionError | AssertionError |  |
| Menu43141(0) | AssertionError | AssertionError |  |
| Menu43141(20191111) | AssertionError | AssertionError |  |

목적 : input과 mileage값에 따른 조건 분기 확인

Mileage = 5000

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input : “abcde” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “하이” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “0.5” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x – 정수만 |
| Input : “-123” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “0123” | 부분 설계 3번 결과 | 부분 설계 3번 결과 | 123원 |
| Input : “1 1” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “0.0” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “9-5” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “5001” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 금액 초과 |
| Input : “1000000000” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 금액 초과 |
| Input : “5000” | 부분 설계 3번 결과 | 부분 설계 3번 결과 |  |
| Input : “00000000” | 부분 설계 3번 결과 | 부분 설계 3번 결과 | 마일리지 사용x |
| Input : “000005000” | 부분 설계 3번 결과 | 부분 설계 3번 결과 | 5000원 |

Mileage = -1

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input : “0” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 금액 초과 |
| Input : “-1” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “5000” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 금액 초과 |

이 경우 전체 프로그램이 애초에 동작하지 않기 때문에(Mockup) 영향x

목적 : 알맞은 예매코드 출력 확인.



|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input : “0”  self.seat\_First=”A1” | 부분 설계 3번 결과  예매 코드  20191111AE2340A1 | 부분 설계 3번 결과  예매 코드  20191111AE2340A1 |  |
| Input : “0”  self.seat\_First=”I1” | 부분 설계 3번 결과  예매 코드  20191111AE2340I1 | 부분 설계 3번 결과  예매 코드  20191111AE2340I1 |  |
| Input : “0”  self.seat\_First=”Z1” | 부분 설계 3번 결과  예매 코드  20191111AE2340Z1 | 부분 설계 3번 결과  예매 코드  20191111AE2340Z1 | 좌석 범위를 벗어남. – 하지만 예약 안됨, 애초에 걸러짐. |

예약 결과





|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input : “0”  self.seat\_First=”D8” | 부분 설계 3번 결과  예매 코드  20191020AA0620D8 | 부분 설계 3번 결과  예매 코드  20191020AA0620D8 |  |
| Input : “0”  self.seat\_First=”H5” | 부분 설계 3번 결과  예매 코드  20191020AA0620H5 | 부분 설계 3번 결과  예매 코드  20191020AA0620H5 |  |
| Input : “0”  self.seat\_First=”A11” | 부분 설계 3번 결과  예매 코드  20191020AA0620A11 | 부분 설계 3번 결과  예매 코드  20191020AA0620A11 | 좌석 범위를 벗어남. – 하지만 예약 안됨, 애초에 걸러짐. |

예약 결과



목적 : self.selected\_movie 의 값 확인

self.selected\_movie = “”

예상 결과 : IndexError

실제 결과 : IndexError

비고 :

* + 1. menu43142 함수

인자로 전달받은 input의 자료형을 str로 assert문을 걸어준 상태.

1. 부분 설계

하나의 인자만을 받아야 함. 아니라면 TypeError를 출력함.

Input은 str형 이어야 한다. 아니라면 AssertionError를 출력함.

인자값 input에 따라 다음 중 하나의 기능을 수행함.

1. “입력 형식에 맞지 않습니다.” 을 출력.
2. “존재하지 않는 카드 번호입니다. 다시 입력해 주세요.” 을 출력
3. “이미 등록된 카드 번호입니다. 다시 입력해 주세요.” 을 출력
4. “결제가 완료되었습니다. 예매 코드 : (예매 코드)” 을 출력함.

Self.\_\_FM.getuser 함수와 self.\_\_FM.bookmovie 함수, self.\_\_FM.savefile 함수를 이용해

UserList, ReservationList, MovieList 파일을 update 함.

출력들을 모두 지우고(clear), 메인 메뉴를 출력함.

self.MI.setMI 함수를 통해 MI.menuNum 값만 4300로 바꿔줌.

1. 검사 방법
2. FileManager, TypeChecker, menuInfo 클래스를 모두 콘솔에 불러오고, Menu 객체를 새로 생성하여 menu4312 함수를 직접 호출함.(Menu의 생성자에서 패키지 참조하는 부분을 수정함)
3. 전체 프로그램을 실행(파일 리스트값들을 일일히 수정하기에 한계가 있음.)시키고, 마일리지 입력 메뉴에서 입력->”BACK”입력을 반복함.

Mileage, 현재 선택된 영화의 데이터를 바꿔가며 검사함.

코드를 추가해 내부 변수들의 값을 바꿈

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 1개) 확인

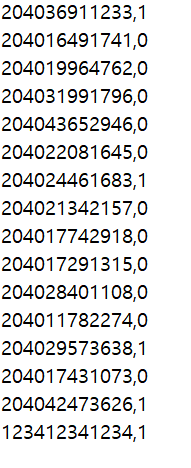
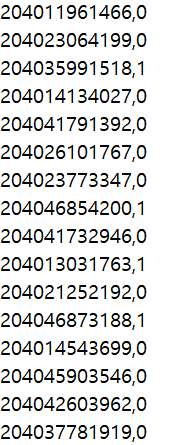
|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu43142() | TypeError | TypeError |  |
| Menu43142(“123”) | TypeError없음 | TypeError없음 | 하지만 AssertionError |
| Menu43142(“123”,”123”) | TypeError | TypeError |  |
| Menu43142(1,1,1,1) | TypeError | TypeError |  |
| Menu43142(1,2,3,4,5,6,7) | TypeError | TypeError |  |

목적 : 인자 input 조건 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Menu43142(“abc-1-”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 |
| Menu43142(0.5) | AssertionError | AssertionError |  |
| Menu43142(-10023) | AssertionError | AssertionError |  |
| Menu43142(“123412341234”) | AssertionError없음 | AssertionError없음 |  |
| Menu43142(“하이”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 |
| Menu43142(123) | AssertionError | AssertionError |  |
| Menu43142(0) | AssertionError | AssertionError |  |
| Menu43142(20191111) | AssertionError | AssertionError |  |

목적 : input 값, CardList 에 따른 조건 분기 확인

CardList



|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input : “abcde” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “하이” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “0.5” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x – 정수만 |
| Input : “-123” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “01230123012” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x - 12자리여야함 |
| Input : “1 1” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “0.0” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “9-5” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “5001” | 부분 설계 1번 결과 | 부분 설계 1번 결과 | 입력 형식x |
| Input : “100000000000” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 카드 존재x |
| Input : “124352239949” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 카드 존재x |
| Input : “999999999999” | 부분 설계 2번 결과 | 부분 설계 2번 결과 | 카드 존재x |
| Input : “123412341234” | 부분 설계 3번 결과 | 부분 설계 3번 결과 | 등록 여부가 1임 |
| Input : “204042473626” | 부분 설계 3번 결과 | 부분 설계 3번 결과 | 등록 여부가 1임 |
| Input : “204013031763” | 부분 설계 3번 결과 | 부분 설계 3번 결과 | 등록 여부가 1임 |
| Input : “204045903546” | 부분 설계 4번 결과 | 부분 설계 4번 결과 |  |
| Input : “204042603962” | 부분 설계 4번 결과 | 부분 설계 4번 결과 |  |
| Input : “204037781919” | 부분 설계 4번 결과 | 부분 설계 4번 결과 |  |

목적 : 알맞은 예매코드 출력 확인.



|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input : “0”  self.seat\_First=”H1” | 부분 설계 4번 결과  예매 코드  20191111AE2340H1 | 부분 설계 4번 결과  예매 코드  20191111AE2340H1 |  |
| Input : “0”  self.seat\_First=”F1” | 부분 설계 4번 결과  예매 코드  20191111AE2340F1 | 부분 설계 4번 결과  예매 코드  20191111AE2340F1 |  |
| Input : “0”  self.seat\_First=”Z1” | 부분 설계 4번 결과  예매 코드  20191111AE2340Z1 | 부분 설계 4번 결과  예매 코드  20191111AE2340Z1 | 좌석 범위를 벗어남. – 하지만 예약 안됨, 애초에 걸러짐. |

예약 결과





|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Input : “0”  self.seat\_First=”D1” | 부분 설계 4번 결과  예매 코드  20191020AA0620D1 | 부분 설계 4번 결과  예매 코드  20191020AA0620D1 |  |
| Input : “0”  self.seat\_First=”H8” | 부분 설계 4번 결과  예매 코드  20191020AA0620H8 | 부분 설계 4번 결과  예매 코드  20191020AA0620H8 |  |
| Input : “0”  self.seat\_First=”A11” | 부분 설계 4번 결과  예매 코드  20191020AA0620A11 | 부분 설계 4번 결과  예매 코드  20191020AA0620A11 | 좌석 범위를 벗어남. – 하지만 예약 안됨, 애초에 걸러짐. |

예약 결과



목적 : self.selected\_movie 의 값 확인

self.selected\_movie = “”

예상 결과 : IndexError

실제 결과 : IndexError

비고 :

* + 1. print\_10days 함수

1. 부분 설계

인자를 받지 말아야 함. 아니라면 TypeError를 출력함.

self.\_\_now\_time 의 값에 따라 해당 날짜로부터 10일 후 까지의 날짜를 일반적인 달력 형식으로 출력함.

숫자 8자리를 인식하는데, 앞에부터 4자리는 년도, 2자리는 월, 2자리는 일로 계산.

4,6,9,11월은 31일 이상이면 다음 월로 바뀜.

2월은 29일 이상이면 다음 월로 바뀜,

그 외 월은 32일 이상이면 다음 월로 바뀜.

12월은 32일 이상이면 다음 년도로 바뀜(9999년이면 0년이됨.)

1. 검사 방법

클래스 메소드 형식을 지우고 파이썬 콘솔에 함수를 입력하고 \_\_now\_time의 값을 바꿔가며 검사함.

1. Test Cases 및 결과 정리

목적 : 인자 개수 확인(0개)

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Print\_10days() | TypeError없음 | TypeError없음 |  |
| Print\_10days (1) | TypeError | TypeError |  |
| Print\_10days (1,0) | TypeError | TypeError |  |
| Print\_10days (1,1,1) | TypeError | TypeError |  |
| Print\_10days (1,True,1) | TypeError | TypeError |  |
| Print\_10days (1,2,15124,55) | TypeError | TypeError |  |
| Print\_10days (1,2,3,4,5,1,1,3434) | TypeError | TypeError |  |

목적 : self.\_\_now\_time의 자료형 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| self.\_\_now\_time=123  Print\_10days() | TypeError | TypeError | Int형x |
| self.\_\_now\_time=0.5  Print\_10days() | TypeError | TypeError | Float형x |
| self.\_\_now\_time=”abc”  Print\_10days() | ValueError | ValueError | 숫자여야함. |
| self.\_\_now\_time=  ”1-333”  Print\_10days() | ValueError | ValueError | 특수문자x |
| self.\_\_now\_time=”01234”  Print\_10days() | ValueError | ValueError | 8자리 이상이어햠. |
| self.\_\_now\_time=”00000000”  Print\_10days() | 0 년  0 월  0 1 2 3 4 5 6  7 8 9 10 | 0 년  0 월  0 1 2 3 4 5 6  7 8 9 10 |  |
| self.\_\_now\_time=  ”0000000012341234”  Print\_10days() | 0 년  0 월  0 1 2 3 4 5 6  7 8 9 10 | 0 년  0 월  0 1 2 3 4 5 6  7 8 9 10 | 8자리까지만 인식 |
| self.\_\_now\_time=”20151230”  Print\_10days() | 2015 년  12 월  30 31  2016 년  1 월  1 2 3 4 5 6 7  8 9 | 2015 년  12 월  30 31  2016 년  1 월  1 2 3 4 5 6 7  8 9 | 년,월 넘어감 |
| self.\_\_now\_time=”20150229”  Print\_10days() | 2015 년  2 월  3 월  1 2 3 4 5 6 7  8 9 10 11 | 2015 년  2 월  3 월  1 2 3 4 5 6 7  8 9 10 11 | 2월은 29일이 없음 |
| self.\_\_now\_time=”20159999”  Print\_10days() | 2015 년  99 월  100 월  1 2 3 4 5 6 7  8 9 10 11 | 2015 년  99 월  100 월  1 2 3 4 5 6 7  8 9 10 11 | 존재하지 않는 월,일 이지만 오류x |
| self.\_\_now\_time=”99991230”  Print\_10days() | 9999 년  12 월  30 31  0 년  1 월  1 2 3 4 5 6 7  8 9 | 9999 년  12 월  30 31  0 년  1 월  1 2 3 4 5 6 7  8 9 |  |
| self.\_\_now\_time=”99991222”  Print\_10days() | 9999 년  12 월  22 23 24 25 26 27 28  29 30 31  0 년  1 월  1 | 9999 년  12 월  22 23 24 25 26 27 28  29 30 31  0 년  1 월  1 |  |

* + 1. printday\_movie 함수

1. 부분 설계

인자를 받지 말아야 함. 아니라면 TypeError를 출력함.

self.day\_movielist 리스트의 값들을 특정 형식에 맞춰 모두 출력하고 True를 return함.

빈 리스트라면 아무것도 하지 않고 False를 return함.

1. 검사 방법

클래스 메소드 형식을 지우고 파이썬 콘솔에 함수를 입력하고 self.day\_movielist 의 공백 여부만 검사함.

self.day\_movielist의 값에 따른 출력은 무엇이 되도 상관없음.

1. Test Cases 및 결과 정리

목적 : 인자 개수 확인(0개)

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| printday\_movie() | TypeError없음 | TypeError없음 |  |
| printday\_movie(1) | TypeError | TypeError |  |
| printday\_movie(1,0) | TypeError | TypeError |  |
| printday\_movie(1,1,1) | TypeError | TypeError |  |
| printday\_movie(1,True,1) | TypeError | TypeError |  |
| printday\_movie(1,2,15124,55) | TypeError | TypeError |  |
| printday\_movie(1,2,3,4,5,1,1,3434) | TypeError | TypeError |  |

목적 : self.day\_movielist 에 따른 return값 확인.

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| day\_movielist=[]  printday\_movie() | False | False |  |
| day\_movielist=[1,2,3,4]  printday\_movie() | TypeError | TypeError |  |
| day\_movielist=[“abcabc”]  printday\_movie() | IndexError | IndexError |  |
| day\_movielist=-1  printday\_movie() | TypeError | TypeError |  |
| day\_movielist=0  printday\_movie() | False | False |  |
| day\_movielist=”abc”  printday\_movie() | IndexError | IndexError |  |
| day\_movielist=False  printday\_movie() | False | False |  |
| day\_movielist=True  printday\_movie() | TypeError | TypeError |  |

* + 1. print\_seat 함수

인자값 movie\_name을 str형으로, 인자값 index를 int형(0이상)으로 assert문을 걸어준 상태

1. 부분 설계

두개의 인자만을 받아야 함. 아니라면 TypeError를 출력함

movie\_name은 str형 이어야 함. 아니라면 AssertionError를 출력함.

Index는 int형(0이상) 이어야 함. 아니라면 AssertionError를 출력함.

self.day\_movielist의 영화 인덱스와 인자값 index, movie\_name을 비교해 다르면 False를 return 하고, 같으면 해당 영화의 좌석 표를 출력하고 True를 return 한다.

Index가 day\_movielist의 길이보다 크거나 같으면 False를 return한다.

1. 검사 방법

Python Console에서 해당 함수를 클래스 형식을 제외하고 입력함.(self를 빼고)

day\_movielist를 임의로 생성하여 검사함.

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 3개) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Print\_seat() | TypeError | TypeError |  |
| Print\_seat(1) | TypeError | TypeError |  |
| Print\_seat(1,0) | TypeError없음 | TypeError없음 | 하지만 AssertionError |
| Print\_seat(1,”abc”) | TypeError없음 | TypeError없음 |  |
| Print\_seat(1,1,1) | TypeError | TypeError |  |
| Print\_seat(1,True,1) | TypeError | TypeError |  |
| Print\_seat(1,2,15124,55) | TypeError | TypeError |  |
| Print\_seat(1,2,3,4,5,1,1,3434) | TypeError | TypeError |  |

목적 : 첫 번째 인자 조건(int형, 0이상) 확인

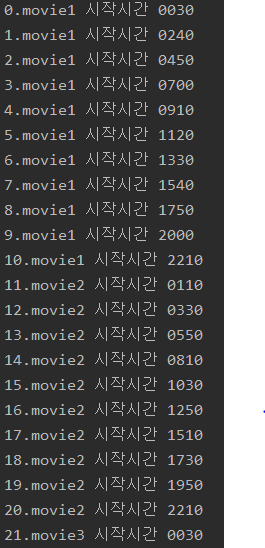
|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Print\_seat(0,”abc”) | AssertionError없음 | AssertionError없음 |  |
| Print\_seat(“a”, ”abc”) | AssertionError | AssertionError |  |
| Print\_seat  (“ㅁㅁㅁ”,”abc”) | AssertionError | AssertionError |  |
| Print\_seat  (-13322,”abc”) | AssertionError없음 | AssertionError없음 |  |
| Print\_seat(0.5,”abc”) | AssertionError | AssertionError |  |
| Print\_seat(123,”abc”) | AssertionError없음 | AssertionError없음 |  |
| Print\_seat(-1,”abc”) | AssertionError | AssertionError |  |
| Print\_seat(-1123,”abc”) | AssertionError | AssertionError |  |

목적 : 두 번째 인자 조건(str형) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Print\_seat(1,”abc”) | AssertionError없음 | AssertionError없음 |  |
| Print\_seat(1,123) | AssertionError | AssertionError |  |
| Print\_seat(1,-123) | AssertionError | AssertionError |  |
| Print\_seat(1,”movie-2”) | AssertionError없음 | AssertionError없음 |  |
| Print\_seat(1,True) | AssertionError | AssertionError |  |
| Print\_seat(1,0.5) | AssertionError | AssertionError |  |

목적 : 인자값과 self.day\_movielist에 따른 결과 확인

영화 정보





영화 movie1 은 좌석이 8x8



영화 movie3 은 좌석이 9x9

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Print\_seat(0,”movie1”) |  |  |  |
| Print\_seat(0,”movie1”) |  |  | A1~A8 예약 시 |
| Print\_seat(21,”movie3”) |  |  |  |
| Print\_seat(21,”movie3”) |  |  | 1열 모두 예약 시 |
| Print\_seat(0,”movie1”) |  |  | 8x1로 바꿈(A1 예약) |
| Print\_seat(0,”movie1”) |  |  | 1x1로 바꿈 |
| Print\_seat(-1,”movie5”) | AssertionError | AssertionError |  |
| Print\_seat  (-123,”movie3”) | AssertionError | AssertionError |  |
| Print\_seat(5,”movie0”) | False | False | 존재하지 않는 영화명 |
| Print\_seat  (21,”movie 3”) | False | False | 존재하지 않는 영화명 |
| Print\_seat(20,”movie3”) | False | False | 영화명과 index 매치x |
| Print\_seat(0,”1.movie1”) | False | False | 존재하지 않는 영화명 |
| Print\_seat(123,”movie1”) | False | False | 영화 목록 초과 |

* + 1. count\_seat 함수 ( 수정 전 )

인자값 input을 str형으로 assert문을 걸어준 상태

Input은 좌석 입력 형식에 맞게 들어와야함. 아니라면 결과를 보장하지 못함.

1. 부분 설계

하나의 인자만을 받아야 함. 아니라면 TypeError를 출력함

Input은 str형 이어야 함. 아니라면 AssertionError를 출력함.

self.selected\_movie의 좌석 정보 중 행의 최대값과 입력한 좌석들의 숫자부분을 비교해 좌석 범위를 초과하면 -1을 return 함.

self.selected\_movie의 좌석 정보 중 해당 좌석이 1(예약된 좌석)이면 -1을 return함.

모든 좌석이 예약할 수 있는 좌석이라면 좌석의 개수를 return함.

1. 검사 방법

Python Console에서 해당 함수를 클래스 형식을 제외하고 입력함.(self를 빼고)

Selected\_movie 를 임의로 생성하여 검사함.

1. Test Cases 및 결과 정리

목적 : 좌석 개수 계산 확인.

selected\_movie = ['20191112AE2210', ['20191112', 'AE', 'movie5', '2210', '0010', 'screen5', '10x9', '000000000', '000000000', '000000000', '000100000', '000000000', '000000000', '000000000', '000000000', '000000000', '000000000']]

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Count\_seat(“Z2”) | Return -1 | IndexError | 좌석 열에 대한 예외x |

* + 1. count\_seat 함수 ( 수정 후 )

인자값 input을 str형으로 assert문을 걸어준 상태

Input은 좌석 입력 형식에 맞게 들어와야함. 아니라면 결과를 보장하지 못함.

1. 부분 설계

하나의 인자만을 받아야 함. 아니라면 TypeError를 출력함

Input은 str형 이어야 함. 아니라면 AssertionError를 출력함.

self.selected\_movie의 좌석 정보 중 행의 최대값과 입력한 좌석들의 숫자부분을 비교해 좌석 범위를 초과하면 -1을 return 함.

self.selected\_movie의 좌석 정보 중 열의 최대값과 입력한 좌석들의 알파벳부분을 비교해 좌석 범위를 초과하면 -1을 return 함.

self.selected\_movie의 좌석 정보 중 해당 좌석이 1(예약된 좌석)이면 -1을 return함.

모든 좌석이 예약할 수 있는 좌석이라면 좌석의 개수를 return함.

1. 검사 방법

Python Console에서 해당 함수를 클래스 형식을 제외하고 입력함.(self를 빼고)

Selected\_movie 를 임의로 생성하여 검사함.

1. Test Cases 및 결과 정리

목적 : 인자 개수 조건(정확히 1개) 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Count\_seat() | TypeError | TypeError |  |
| Count\_seat(“123”) | Return -1 | Return -1 |  |
| Count\_seat(“A1”) | Return 1 | Return 1 |  |
| Count\_seat(“123”,”123”) | TypeError | TypeError |  |
| Count\_seat(1,1,1,1) | TypeError | TypeError |  |
| Count\_seat(1,2,3,4,5,6,7) | TypeError | TypeError |  |

목적 : 인자 input 조건 확인

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Count\_seat(“abc”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 -> ValueError |
| Count\_seat(0.5) | AssertionError | AssertionError |  |
| Count\_seat(-10023) | AssertionError | AssertionError |  |
| Count\_seat(“A1”) | AssertionError없음  Return 1 | AssertionError없음  Return 1 |  |
| Count\_seat(“하이”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 -> ValueError |
| Count\_seat(123) | AssertionError | AssertionError |  |
| Count\_seat(0) | AssertionError | AssertionError |  |
| Count\_seat(20191111) | AssertionError | AssertionError |  |
| Count\_seat(“A1-A2”) | AssertionError없음 | AssertionError없음 | 하지만 입력형식에 맞지 않음 -> ValueError |

목적 : 좌석 개수 계산 확인.

selected\_movie = ['20191112AE2210', ['20191112', 'AE', 'movie5', '2210', '0010', 'screen5', '10x9', '000000000', '000000000', '000000000', '000100000', '000000000', '000000000', '000000000', '000000000', '000000000', '000000000']]

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Count\_seat(“Z2”) | Return -1 | Return -1 | 행 개수 확인 |
| Count\_seat(“A1”) | Return 1 | Return 1 |  |
| Count\_seat(“A1~A5”) | Return 5 | Return 5 |  |
| Count\_seat(“A6~B2”) | IndexError | IndexError | 입력 형식x |
| Count\_seat(“A2~B3”) | Return 2 | Return 2 | 입력 형식x |
| Count\_seat(“A2~A10”) | Return -1 | Return -1 |  |
| Count\_seat(“A123”) | Return -1 | Return -1 | 열 개수 확인 |
| Count\_seat(“1A”) | ValueError | ValueError | 입력 형식x |
| Count\_seat(“A1-A2”) | ValueError | ValueError | 입력 형식x |
| Count\_seat(“A1,A2,A3,A4”) | Return 4 | Return 4 |  |
| Count\_seat(“I1,B2,A3,H4”) | Return 4 | Return 4 |  |
| Count\_seat(“A5~A1”) | IndexError | IndexError | 입력 형식x |
| Count\_seat(“D1~D9”) | Return -1 | Return -1 | 중간에 예약된 좌석이 있음 |
| Count\_seat(“D4”) | Return -1 | Return -1 | 이미 예약된 좌석 |
| Count\_seat(“D1~D4”) | Return -1 | Return -1 | 마지막에 예약된 좌석이 있음 |
| Count\_seat(“D4~D5”) | Return -1 | Return -1 | 처음에 예약된 좌석이 있음 |
| Count\_seat(“D4,D5”) | Return -1 | Return -1 | 예약된 좌석이 있음 |
| Count\_seat(“A1~A4,A6”) | ValueError | ValueError | 입력 형식x |
| Count\_seat(“a1”) | Return -1 | Return -1 | 입력 형식x |
| Count\_seat(“A0”) | Return -1 | Return -1 | 좌석 범위x |
| Count\_seat(“A-1”) | Return -1 | Return -1 | 입력 형식x |
| Count\_seat(“A--1”) | ValueError | ValueError | 입력 형식x |
| Count\_seat(“A1~,4”) | Return 4 | Return 4 | 입력 형식x |
| Count\_seat(“A1~~A4”) | ValueError | ValueError | 입력 형식x |
| Count\_seat(“A1,,A4”) | ValueError | ValueError | 입력 형식x |
| Count\_seat(“A1 A4”) | ValueError | ValueError | 입력 형식x |
| Count\_seat(“ A1”) | ValueError | ValueError | 입력 형식x |

목적 : selected\_movie 의 값에 따른 출력 확인.

selected\_movie = ['20191112AE2210', ['20191112', 'AE', 1, '2210', '0010', 'screen5', '10x9', '123412344', '000000000', '000000000', '000100000', '000000000', '000000000', '000000000', '000000000', '000000000', '000000000']]

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| Count\_seat(“A1”) | 정상 동작 | 정상 동작 | Selected\_movie에 이상한 값이 들어가도 영향x |
| Count\_seat(“A2”) | 정상 동작 | 정상 동작 | ‘’ |
| Count\_seat(“A1~A5”) | 정상 동작 | 정상 동작 | ‘’ |
| Count\_seat(“Z1”) | 정상 동작 | 정상 동작 | ‘’ |
| Count\_seat(“D4~D5”) | 정상 동작 | 정상 동작 | ‘’ |

# TypeChecker 클래스

* + 1. ID\_check 함수

1. 부분 설계

* ID를 입력받아 검증하여 형식에 맞지 않으면 False, 형식에 맞고 존재하지 않는(사용 가능한) 아이디이면 True를 반환하는 boolean 타입의 함수이다.   
  특정 문자열을 인자로 이 함수를 호출하면 먼저 입력받은 문자열의 길이가 4자 이상 10자 이하인지 확인한다. 맞으면 계속 진행하고 틀리면 “입력 형식에 맞지 않습니다.”라는 메시지를 출력하고 False를 반환한다.   
  다음으로 문자열에 포함된 숫자와 문자의 개수를 센다. 문자는 소문자여야 한다. 문자나 숫자가 한 개도 없거나, 공백이 포함된 경우 “입력 형식에 맞지 않습니다.”라는 메시지를 출력 후 False를 반환한다.  
  숫자나 영문 소문자로 이루어진 4자 이상 10자 이하의 공백이 없는 문자열이라면, 회원 정보가 저장된 “UserList.csv”파일을 불러와서 해당 파일에 입력된 문자열과 같은 아이디가 존재하는지 확인한다. 존재하면 “존재하는 아이디입니다.” 출력 후 False 반환, 존재하지 않으면 “사용 가능한 아이디입니다.” 출력 후 True를 반환한다.

1. 검사 방법

* TypeChecker 클래스 상에서 이 함수를 호출하고, 출력하는 메시지를 확인한다. 반환값에 따라 출력하는 메시지가 다르도록 설정되어 있다.

1. Test Cases 및 결과 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| ‘effwfoj’ | 사용 가능한 아이디입니다. | 사용 가능한 아이디입니다. |  |
| ‘efefe’ | 사용 가능한 아이디입니다. | 사용 가능한 아이디입니다. |  |
| ‘13u8ser’ | 존재하는 아이디입니다. | 존재하는 아이디입니다. |  |
| ‘ ‘(공백 한칸) | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| ‘ ‘(공백 다섯칸) | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| u ser | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| us | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| &&$%$ | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| ‘사용자’ | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| 1 | 입력 형식에 맞지 않습니다. | TypeError |  |
| 22323 | 존재하지 않는 아이디입니다. | TypeError |  |

int형으로 적으면 코드가 계속 진행되다가 len()함수를 사용하는 부분부터 오류가 나는데, int형으로 입력된 경우 코드 진행을 제한하기 위한 장치가 없다. 올바른 동작을 위해서는 입력 형식을 String으로 제한해야 한다.

* + 1. date\_check 함수

1. 부분 설계
2. 검사 방법

* 스트링 또는 다른 타입을 인자로 date\_check 함수를 호출한다. 길이를 정상 입력과 다르게도 해 보고, 스트링이 아닌 다른 타입도 넣어 본다. 그리고 올바른 타입의 올바른 길이일 때는 입력 형식을 얼마나 잘 구분하는지 테스트한다.

1. Test Cases 및 결과 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 20191004 | (출력 없음) | AttributeError | isdigit를 사용하지 못함 |
| 2019 | 날짜가 입력 형식에 맞지 않습니다. | AttributeError |  |
| 1.11 | 날짜가 입력 형식에 맞지 않습니다. | AttributeError |  |
| 20191020 | (출력 없음) | (출력 없음) |  |
| ‘20191020’ | (출력 없음) | (출력 없음) |  |
| ‘99991020’ | (출력 없음) | (출력 없음) | 연도는 9999까지 입력 받는 것으로 기획서에 명시되어 있음 |
| ‘20191120’ | (출력 없음) | (출력 없음) |  |
| ‘20191130’ | (출력 없음) | (출력 없음) |  |
| ‘20191131’ | 날짜가 입력 형식에 맞지 않습니다. | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘20191229’ | (출력 없음) | (출력 없음) |  |
| ‘20191230’ | (출력 없음) | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘20191231’ | (출력 없음) | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘20190531’ | (출력 없음) | (출력 없음) |  |
| ‘20190530’ | (출력 없음) | (출력 없음) |  |
| ‘20190431’ | 날짜가 입력 형식에 맞지 않습니다. | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘20190430’ | (출력 없음) | (출력 없음) |  |
| ‘20190831’ | (출력 없음) | (출력 없음) |  |
| ‘20190731’ | (출력 없음) | (출력 없음) |  |
| ‘20190228’ | (출력 없음) | (출력 없음) |  |
| ‘20190229’ | 날짜가 입력 형식에 맞지 않습니다. | (출력 없음) | 2월은 28일까지만 입력받는 것으로 기획서에 명시되어 있으므로 이 경우는 오류 |
| ‘20190230 | 날짜가 입력 형식에 맞지 않습니다. | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘2019022’ | 8자리 숫자만 입력 가능합니다. | 8자리 숫자만 입력 가능합니다. |  |
| ‘201902200’ | 8자리 숫자만 입력 가능합니다. | 8자리 숫자만 입력 가능합니다. |  |
| ‘201912033’ | 8자리 숫자만 입력 가능합니다. | 8자리 숫자만 입력 가능합니다. |  |

* + 1. time\_check 함수

1. 부분 설계

* date\_check 함수에 시간 검증하는 코드가 추가된 함수. date\_check와의 차이점은 인자를 16자리 스트링으로 받고, date\_check함수가 True를 반환할 조건에서 시간 부분 입력 형식을 확인하는 부분이 더 있다는 점이다. 시간이 23시를 넘거나 분이 59분을 넘으면(23시 59분까지 정상입력) False를 반환한다.

1. 검사 방법

* date\_check 함수와 동일

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| ‘201912291230’ | (출력 없음) | (출력 없음) |  |
| ‘201912301230’ | (출력 없음) | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘201912311230’ | (출력 없음) | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘201913300000’ | 날짜가 입력 형식에 맞지 않습니다. | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘201900121330’ | 날짜가 입력 형식에 맞지 않습니다. | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘201912001330’ | 날짜가 입력 형식에 맞지 않습니다. | 날짜가 입력 형식에 맞지 않습니다. |  |
| ‘201912012540’ | 시간이 입력 형식에 맞지 않습니다. | 시간이 입력 형식에 맞지 않습니다. |  |
| ‘201912013030’ | 시간이 입력 형식에 맞지 않습니다. | 시간이 입력 형식에 맞지 않습니다. |  |
| ‘201912019999’ | 시간이 입력 형식에 맞지 않습니다. | 시간이 입력 형식에 맞지 않습니다. |  |
| ‘201912011990’ | 시간이 입력 형식에 맞지 않습니다. | 시간이 입력 형식에 맞지 않습니다. |  |
| ‘201912012700’ | 시간이 입력 형식에 맞지 않습니다. | 시간이 입력 형식에 맞지 않습니다. |  |
| ‘201912012370’ | 시간이 입력 형식에 맞지 않습니다. | 시간이 입력 형식에 맞지 않습니다. |  |
| ‘201912012369’ | 시간이 입력 형식에 맞지 않습니다. | 시간이 입력 형식에 맞지 않습니다. |  |
| ‘201912012359’ | (출력 없음) | (출력 없음) |  |
| ‘2019 2012359’ | 8자리 숫자만 입력 가능합니다 | 8자리 숫자만 입력 가능합니다 | 8자리🡪12자리로 수정 필요 |
| ‘201912ff2359’ | 8자리 숫자만 입력 가능합니다 | 8자리 숫자만 입력 가능합니다 |  |
| ‘2019120123ff’ | 8자리 숫자만 입력 가능합니다 | 8자리 숫자만 입력 가능합니다. |  |
| ‘201912022359 ’ | 8자리 숫자만 입력 가능합니다 | 8자리 숫자만 입력 가능합니다 |  |
| null(문자열 없음) | 8자리 숫자만 입력 가능합니다 | 8자리 숫자만 입력 가능합니다 |  |
| (공백 1칸) | 8자리 숫자만 입력 가능합니다. | 8자리 숫자만 입력 가능합니다. |  |

* + 1. pw\_check 함수

1. 부분 설계

* 비밀번호를 검증하는 함수로, 길이가 4자 이상 10자 이하인지 우선 확인하고, 길이가 다를 경우에는 “입력 형식에 맞지 않습니다.”라는 메시지를 출력하고 False를 반환한다. 길이가 맞으면 문자나 숫자로 이루어진 문자열인지 확인하고 문자의 개수, 숫자의 개수를 센다. 숫자와 문자의 개수의 합이 0이거나 공백이 포함된 경우 “입력 형식에 맞지 않습니다.”라는 메시지를 출력하고 False를 반환하고, 그 조건에 해당되지 않으면 “사용 가능한 비밀번호입니다.”라는 메시지를 출력하고 True를 반환한다.

1. 검사 방법

* 정수, 소수를 인자로 pw\_check함수를 호출해보고, 입력을 스트링으로 자유롭게 받아서도 호출해본다.

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 2345234 | 사용 가능한 비밀번호입니다. | TypeError | int사용, int에 대해len()함수를 사용 못함 |
| 234.234 | 입력 형식에 맞지 않습니다. | TypeError | float사용, float에 대해 len()함수를 사용 못함 |
| null(문자열 없음) | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| (공백 1) | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| (공백 2) | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| (공백 4) | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| (공백 5) | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| d | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| ddd | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| dddd | 사용 가능한 비밀번호입니다. | 사용 가능한 비밀번호입니다. |  |
| dddddddddd | 사용 가능한 비밀번호입니다. | 사용 가능한 비밀번호입니다. |  |
| ddddddddddd | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| 1111111111 | 사용 가능한 비밀번호입니다. | 사용 가능한 비밀번호입니다. |  |
| 11111111111 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| 1 1111111111 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| $123123 | 입력 형식에 맞지 않습니다. | 사용 가능한 비밀번호입니다. |  |
| $abcabc | 입력 형식에 맞지 않습니다. | 사용 가능한 비밀번호입니다. |  |
| abcabc$ | 입력 형식에 맞지 않습니다. | 사용 가능한 비밀번호입니다. |  |
| abc$abc | 입력 형식에 맞지 않습니다. | 사용 가능한 비밀번호입니다. |  |
| 123$123 | 입력 형식에 맞지 않습니다. | 사용 가능한 비밀번호입니다. |  |
| 123123$ | 입력 형식에 맞지 않습니다. | 사용 가능한 비밀번호입니다. |  |
| 123 123 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |

특수문자가 들어있는데도 True를 반환하는 것은 입력된 문자열을 검증하는 조건문에서 발생한 오류이다. 숫자 또는 문자인 조건에만 카운트하고, 숫자나 문자가 아닌 조건에는 False를 반환하며 함수를 종료해야 했는데, 아닌 조건에 바로 끝내지 않고 다음 글자를 검증하기 때문에 발생한 오류였다.

* + 1. movieTitle 함수

1. 부분 설계
2. 검사 방법
3. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출 | 예상 결과 | 실제 결과 | 비고 |
| 232 | 상영중인 영화가 아닙니다. | TypeError | int입력 |
| 2.32 | 입력 형식에 맞지 않습니다. | TypeError | float입력 |
| 23223 | 상영중인 영화가 아닙니다. | 상영중인 영화가 아닙니다. |  |
| ‘movie1’ | (출력 없음) | (출력 없음) |  |
| ‘movi1’ | 상영중인 영화가 아닙니다. | 상영중인 영화가 아닙니다. |  |
| ‘moviemoviemoviemovie’ | 상영중인 영화가 아닙니다. | 상영중인 영화가 아닙니다. |  |
| ‘moviemoviemoviemovie2’ | 입력 형식에 맞지 않습니다. | 상영중인 영화가 아닙니다 | 21자 입력 |
| (아무것도 입력하지 않음) | 입력 형식에 맞지 않습니다. | 상영중인 영화가 아닙니다. |  |
| (공백 1칸) | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| (공백 4칸) | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| movie | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| m ovie | 상영중인 영화가 아닙니다. | 입력 형식에 맞지 않습니다. |  |
| m ovie | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| $#$%#$% | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| movie$ | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| $movie | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |

* + 1. cardNum 함수

1. 부분 설계

* 카드번호를 검증하는 함수로, 입력이 12자리의 숫자인지 우선 확인한 후 맞으면 CasrList.csv 파일 안의 내용과 대조해서 해당 파일 안에 입력된 문자열과 같은 카드번호가 존재하는지 확인하고 존재하면 True, 존재하지 않으면 False를 반환한다.

1. 검사 방법

* int, float, str을 인자로 cardNum함수를 호출한다. 데이터 타입은 맞지만 숫자 12자리만으로 이루어지지 않은 입력도 넣는다. 그리고 이 함수가 출력하는 메시지를 확인한다.

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 20190929 | 입력 형식에 맞지 않습니다. | AttributeError | int입력 |
| 123412341234 | 해당하는 카드번호가 없습니다 | AttributeError | int입력 |
| 1234.1234 | 입력 형식에 맞지 않습니다. | AttributeError | float입력 |
| 123 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| 12341234123 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| 1234123412345 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| 123412341234123412341234 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| 123412341234 | 해당하는 카드번호가 없습니다 | 입력 형식에 맞지 않습니다. |  |
| 204023801186 | (출력 없음) | 입력 형식에 맞지 않습니다. |  |
| 204023092007 | (출력 없음) | 입력 형식에 맞지 않습니다. |  |
| ff123412341234 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |  |
| 12ff12341234 | 입력 형식에 맞지 않습니다. | 해당하는 카드번호가 없습니다. |  |
| 1234ff341234 | 입력 형식에 맞지 않습니다. | 해당하는 카드번호가 없습니다 |  |
| 123412ff1234 | 입력 형식에 맞지 않습니다. | 해당하는 카드번호가 없습니다 |  |
| 12341234ff34 | 입력 형식에 맞지 않습니다. | 해당하는 카드번호가 없습니다 |  |
| 1234123412ff | 입력 형식에 맞지 않습니다. | 해당하는 카드번호가 없습니다 |  |

입력은 스트링 타입으로 받기 때문에 입력된 글자가 모두 숫자인지 뿐만 아니라, 입력이 숫자로 이루어진 스트링인지를 확인할 필요가 있다. 그리고 올바른 형식일 때 False를 반환하고, 틀린 형식일 때 True를 반환하므로 조건문을 수정할 필요가 있다.

* + 1. checkyourseat 함수

1. 부분 설계
2. 검사 방법
3. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| ‘aaaaa’ | AssertionError | AssertionError |  |
| 123123 | AssertionError | AssertionError |  |
| 3.14 | AssertionError | AssertionError |  |
| [‘A1’, ‘A2’] | True | True |  |
| [‘A1’] | True | True |  |
| [‘A1’, ‘A2’, ‘A3’] | True | True |  |

* + 1. checkReservationCode 함수

1. 부분 설계
2. 검사 방법
3. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 2019201920192019 | AssertionError | AssertionError |  |
| 2019.2019 | AssertionError | AssertionError |  |
| 20190929AA0930F | False | False |  |
| 20190929AA0930F1 | True | True |  |
| 20190929A00930F1 | False | True |  |
| 20190929AA093FF1 | False | True |  |
| 2019092AAA0930F1 | False | True |  |
| 20190929AA0930FF | False | False |  |
| 20190929AAA930F1 | False | False |  |

# TypeChecker 클래스 2차 설계

* + 1. ID\_check 함수

1. 부분 설계

* FileManager 클래스가 추가되어 csv 파일은 FileManager클래스 내의 함수로 처리하므로 입력 받은 아이디를 UserList.csv 파일 내의 내용과 대조하는 구문 삭제. assert문 추가

1. 검사 방법

* 임의의 변수에 ID\_check 함수의 반환값을 저장해서 그 값을 출력한다.

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 1 | AssertionError | AssertionError |  |
| 1.5 | AssertionError | AssertionError |  |
| 11111 | AssertionError | AssertionError |  |
| 1231234 | AssertionError | AssertionError |  |
| ‘12341234’ | True | True |  |
| ‘e12e’ | True | True |  |
| ‘e’ | False | False |  |
| ‘ ‘ | False | False |  |
| ‘ ‘ | False | False |  |
| ‘ e’ | False | False |  |
| ‘e12f31f13fe1212fe’ | False | False |  |
| ‘#$#%#’ | False | False |  |
| ‘#$e#%%’ | False | False |  |
| ‘1111111111’ | True | True |  |
| ‘11111111111’ | False | False |  |
| ‘eeeeeeeeee’ | True | True |  |
| ‘eeeeeeeeeee’ | False | False |  |
| ‘1e1e1e1e1e1’ | False | False |  |
| 111 | False | False |  |
| 1111 | True | True |  |
| eee | False | False |  |
| eeee | True | True |  |
| 1e1 | False | False |  |
| 1e1e | True | True |  |
| 가 | False | False |  |
| 가나다 | False | False |  |
| 가나다라 | False | False |  |
| 가1나1가 | False | False |  |
| 1가1나 | False | False |  |

* + 1. date\_check 함수

1. 부분 설계

* 12월 30일, 31일에 대해 False를 반환하는 문제 수정, Assert문 추가하여 입력은 str로만 받는다.

1. 검사 방법

* 임의의 변수를 만들어 date\_check함수의 반환값을 저장하고 해당 변수값을 출력시켜 출력값을 확인, 이때 date\_check 함수는 정수, 소수 데이터 타입의 값을 인자로 호출. 데이터 타입이 맞지 않을 때 assert 오류가 제대로 출력되는지 확인

1. Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 20191021 | AssertError | AssertError |  |
| 1.2 | AssertError | AssertError |  |
| 3.14 | AssertError | AssertError |  |
| ‘20191021’ | True | True |  |
| ‘201910ff’ | False | False |  |
| ‘20191031’ | True | True |  |
| ‘20191131’ | False | False |  |
| ‘20190229’ | False | True |  |
| ‘20190230’ | False | False |  |
| ‘20191229’ | True | True |  |
| ‘20191230’ | True | True |  |
| ‘20191331’ | False | False |  |
| ‘20192020’ | False | False |  |
| ‘f0191021’ | False | False |  |
| ‘2f191021’ | False | False |  |
| ‘20f91021’ | False | False |  |
| ‘201f1021’ | False | False |  |
| ‘2019f021’ | False | False |  |
| ‘20191f21’ | False | False |  |
| ‘201910f1’ | False | False |  |
| ‘2019102f’ | False | False |  |

* + 1. time\_check 함수

1. 부분 설계

* date\_check 함수의 변경사항과 동일

1. 검사 방법

* date\_check 함수의 검사 방법과 동일

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 3.14 | AssertError | AssertError |  |
| 1 | AssertError | AssertError |  |
| ‘201910201230’ | True | True |  |
| '201910202440' | False | True | 시간 입력 24시까지 가능하게 잘못 설계됨 |
| ‘201910202540’ | False | False |

* + 1. pw\_check 함수

1. 부분 설계

* if문을 수정해서 한 글자씩 검증하다 숫자나 문자가 아닌 글자를 발견하면 바로 False를 반환하도록 하였다. Assert문 추가해서 함수 코드가 계속 진행되는 입력은 str뿐이다.

1. 검사 방법

* 임의의 변수에 pw\_check 함수의 반환값을 저장해서 그 값을 출력한다.

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 1 | AssertionError | AssertionError |  |
| 2.2 | AssertionError | AssertionError |  |
| 123234 | AssertionError | AssertionError |  |
| ‘111’ | False | False |  |
| ‘11111’ | True | True |  |
| ‘e123efe’ | True | True |  |
| ‘e123efe455’ | True | True |  |
| ‘e123efe455e’ | False | False |  |
| ‘eee’ | False | False |  |
| ‘eeee’ | True | True |  |
| ‘vvv’ | False | False |  |
| ‘vvvv’ | True | True |  |
| ‘vcvcvcvcvc’ | True | True |  |
| ‘cvcvcvcvcvc’ | False | False |  |
| ‘12vv34eee’ | True | True |  |
| ‘12vv34eeee’ | True | True |  |
| ‘12vv34eee345’ | False | False |  |
| ‘가나다’ | False | False |  |
| ‘가나다라’ | False | False |  |
| ‘@#$!’ | False | False |  |
| ‘1@#$’ | False | False |  |
| ‘@#1#’ | False | False |  |
| ‘@1@#’ | False | False |  |
| ‘@##$1’ | False | False |  |
| ‘가1나1’ | False | False |  |
| ‘가11나’ | False | False |  |
| ‘$eee$’ | False | False |  |
| ‘a$%^a’ | False | False |  |
| ‘a가나b’ | False | False |  |
| ‘a1825b’ | True | True |  |

* + 1. checkMovieTitle 함수

1. 부분 설계

* FileManager 클래스가 추가되어 csv 파일은 FileManager클래스 내의 함수로 처리하므로 입력 받은 영화 제목을 MovieList.csv 파일 내의 내용과 대조하는 구문 삭제   
  출력 뿐만 아니라 입력도 숫자+영화제목의 5.movie와 같은 형식으로 이루어져야 하므로 영화 제목 앞에 숫자,온점(‘.’)이 포함되었는지도 검증하도록 부분 설계 변경.   
  공백으로 시작하는 경우를 제외하고 한 칸씩 들어간 공백은 허용하도록 기획서에 명시되어 있는데, 1차 설계 때 제대로 반영되어 있지 않았음. 해당 내용 반영하여 수정.  
  오류 메시지는 Menu함수에서 처리하므로 함수 내부의 오류 메시지 삭제. assert문 추가

1. 검사 방법

* Test1 : 값을 지정해서 해당 값을 인자로 받는 checkMovieTitle 함수를 호출하고 반환값을 출력시킨다.
* Test2 : String으로 자유롭게 입력받아 어떤 경우에 True를, 어떤 경우에 False를 반환하는지 확인한다.

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 1 | AssertionError | AssertionError | Test1 |
| 2.2 | AssertionError | AssertionError |  |
| ‘121’(이하 따옴표 생략) | False | False | Test2 |
| 3 | False | False |
| 3.3 | False | False |
| 3.3.3 | False | False |
| (공백 1칸) | False | False |
| (공백 3칸) | False | False |
| 3.3.3.lionking | False | False |
| 3.3.lionking | False | False |
| lionking | False | False |
| 4.joker | True | False |
| 4. joker | False | False |
| 4.j oker | True | True |
| 4.joker (공백 1) | True | True |
| 4.joker (공백 4) | False | False |
| 4.%$%$ | False | False |
| 4.Joker | False | False |
| .joker | False | False |
| 4.jokerjokerjokerjoke | True | False | Test2, 숫자 포함 20자 넘는 영화제목에 오류 출력되게 잘못 설계됨 |

기획서에 따르면 번호를 제외한 영화 제목만 공백 포함 20자 이하여야 하는데, 번호를 포함해서 20자를 넘으면 오류를 출력하도록 잘못 설계되어 있음을 확인.

* + 1. cardNum 함수

1. 부분 설계

* 형식에 맞는 경우에도 False를 반환하던 것을 수정했다. 문자열이 12자, 숫자, 공백 없음을 모두 만족해야 True를 반환한다. Assert문도 추가했다.

1. 검사 방법

* int, float를 인자로 함수를 호출해보고, str을 자유롭게 입력받아 그 입력을 인자로도 함수를 호출한다. 함수의 반환값을 임의의 변수에 저장해 그 변수를 출력한다.

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 2 | AssertError | AssertError |  |
| 2.2 | AssertError | AssertError |  |
| 123412341234 | AssertError | AssertError |  |
| ‘123412341234’ | True | True |  |
| ‘1 23412312’ | False | False |  |
| ‘1234123412 4’ | False | False |  |
| ‘ 23412341234’ | False | False |  |
| ‘12ff12341234’ | False | False |  |
| ‘1234ff341234’ | False | False |  |
| ‘1234 ff1234’ | False | False |  |
| ‘fffff’ | False | False |  |
| ‘12341234123’ | False | False |  |
| ‘1234 2341234’ | False | False |  |
| ‘12341234123f’ | False | False |  |
| ‘1234123412ff’ | False | False |  |
| ‘ewfefefefefe’ | False | False |  |
| ‘!@#$!@#$!@#$’ | False | False |  |
| 123412341234e | False | False |  |
| ‘ 23412341234’ | False | False |  |
| ‘1234 341234’ | False | False |  |
| ‘123 12341234’ | False | False |  |
| ‘123412341234 ‘ | False | False |  |

* + 1. checkReservationCode 함수

1. 부분 설계

* 파이썬에서는 배열의 명시된 범위 바로 앞의 요소까지만 인자로 받는다는 것을 확인.  
  code[0:7] 🡪 code[0:8]  
  code[10:13] 🡪 code[10:15]  
  code[8:9] 🡪 code[8:10]으로 수정

1. 검사 방법

* 1차 설계와 동일

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 20190929 | AssertionError | AssertionError |  |
| 3.3 | AssertionError | AssertionError |  |
| 21123 | AssertionError | AssertionError |  |
| 20190929AA0930F1 (이하 스트링) | True | True |  |
| 20190929AA0930FF | False | False |  |
| 20190929AA0930f1 | False | False |  |
| 20190929aA0930F1 | False | False |  |
| 20190929AAA930F1 | False | False |  |
| 20190929AA093FF1 | False | False |  |
| 20190929AA0A30F1 | False | False |  |
| 2019092AAA0930F1 | False | False |  |
| 2019AA29AA0930F1 | False | False |  |
| 20AA0929AA0930F1 | False | False |  |
| 20190929AA0930F | False | False |  |
| 20190929AA0930 | False | False |  |
| 2222222222222222 | False | False |  |
| 20190929AA0930F1 | False | False |  |
| 20190929AA0930F1 | False | False |  |
| 20190929AA0930F1 | False | False |  |
| 20190929AA0930F1 | False | False |  |
| 20190929AA0930F1 | False | False |  |

# TypeChecker 클래스 3차 설계

checkyourseat함수만 단독으로 테스트할 땐 좌석이 반환된 배열을 인자로 받는 상황만 가정하여서, 올바르게 동작하는 줄 알았으나, 프로그램의 일부로 동작할 땐 제대로 동작하지 않음. 원인은 입력 문자열 스트링을 FileManager 클래스의 seats\_to\_list 함수로 한 좌석씩 나눠서 좌석들이 담긴 리스트를 반환한 후 checkyourseat함수로 검증하였는데, 형식 검증보다 문자열을 나누는 과정이 먼저 이루어져 발생한 오류였다. 좌석 문자열의 형식이 먼저 검증된 후에 배열로 반환되어야 한다고 판단, 이 클래스에서 좌석 문자열 형식을 검증하고 배열로 반환하는 기능까지 수행하도록 설계 변경하였다. 이 과정에서 FileManager의 seats\_to\_list는 사용할 필요가 없어졌고, 코드의 효율성을 위해 한 좌석씩 검증하는 checkOneSeat 함수를 추가했다. time\_check 23시까지로 수정, cardNum의 조건문을 수정했다. checkMovieTitle 함수도 번호 제외 20자까지 받도록 수정했다.

* + 1. date\_check 함수

1. 부분 설계

* 12월을 구분하게 수정하니 2월 29일을 제대로 구분하지 못해, 같은 자리에 2가 오는 12월과 02월을 명확하게 구분하여 처리하도록 수정

1. 검사 방법

* 2차 설계와 동일

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 20190530 | AssertionError | AssertionError |  |
| 2019.0530 | AssertionError | AssertionError |  |
| ‘201900530’ | False | False |  |
| ‘2019530’ | False | False |  |
| ‘20190229’ | False | False |  |
| ‘20191230’ | True | True |  |
| ‘20191130’ | True | True |  |
| ‘20191230’ | True | True |  |
| ‘20190228’ | True | True |  |
| ‘20192030’ | False | False |  |
| ‘20190230’ | False | False |  |
| ‘20191131’ | False | False |  |
| ‘20191130’ | True | True |  |
| ‘20191030’ | True | True |  |
| ‘20191031’ | True | True |  |
| ‘20191231’ | True | True |  |
| ‘20191331’ | False | False |  |
| ‘20190731’ | True | True |  |
| ‘20190831’ | True | True |  |
| ‘20190531’ | True | True |  |
| ‘20190431’ | True | True |  |
| ‘gg191230’ | False | False |  |
| ‘20gg1230’ | False | False |  |
| ‘2019gg30’ | False | False |  |
| ‘201912gg’ | False | False |  |
| ‘#0191230’ | False | False |  |
| ‘2##91230’ | False | False |  |
| ‘201##230’ | False | False |  |
| ‘20191##0’ | False | False |  |
| ‘2019123#’ | False | False |  |

* + 1. time\_check 함수

1. 부분 설계

* 시간 검증에서 False를 출력하는 조건을 25시부터에서 24시부터로 수정. 다른 부분은 2차 설계와 상동

1. 검사 방법

* 2차 설계와 동일

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 2 | AssertError | AssertError |  |
| 4.5 | AssertError | AssertError |  |
| '2019092' | False | False |  |
| '201910201230' | True | True |  |
| '201910202440' | False | False |  |
| '201910202540' | False | False |  |
| '201910201960' | False | False |  |
| '201910401230' | False | False |  |
| 'ff1910301230' | False | False |  |
| '20ff10301230' | False | False |  |
| '2019ff301230' | False | False |  |
| '201910ff1230' | False | False |  |
| '20191030ff30' | False | False |  |
| '2019103012ff' | False | False |  |
| '20191012213031' | False | False |  |
| '201910 1230' | False | False |  |
| '2019102 1230' | False | False |  |
| '201910203040' | False | False |  |

* + 1. checkMovieTitle 함수

1. 부분 설계

* 번호 제외한 영화 제목만 공백 포함 20자 이하만 받도록 설계 수정.

1. 검사 방법

* 3차 설계와 동일

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 3 | AssertionError | AssertionError | Test1 |
| 5.12 | AssertionError | AssertionError |
| ‘4.joker’(이하 따옴표 생략) | True | True | Test2 |
| 4.jokerjokerjokerjoker | True | True |
| 4.jokerjokerjokerjokerr | False | False |
| 4. | False | False |
| 4. (공백 1) | False | False |
| (공백1) | False | IndexError |
| 4.4.4 | False | False |
| 4.4 | False | False |
| 4.%$% | False | False |
| #$$3.#$# | False | False |
| 12.ee | True | True |
| R.RR | False | False |
| r.rr | False | False |
| e.joker | False | False |
| 6.육 | False | False |
| 4.4444444444444444 | False | False |
| 121vvv | False | IndexError |  |

‘.’이 포함되지 않은 문자열에서 인덱스 오류가 발생함

* + 1. checkOneSeat 함수

1. 입력된 문자열의 길이가 2이고, 첫번째 글자는 대문자이고 두번째 글자는 숫자인지 검증.  
   세 조건이 모두 맞다면 True 반환, 하나라도 틀리면 False 반환.
2. 검사 방법  
   TypeChecker 클래스 상에 테스트를 위한 코드를 작성 : 반환값에 따라 출력 메시지가 달라지는 조건문을 추가. 올바른 형식이면 “올바른 형식” 출력, 틀린 형식이면 “틀린 형식 출력. 반복문에 조건문을 넣고 checkOneSeat 함수를 반복해서 호출하고 반복해서 입력.
3. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 11 | 틀린 형식 | 틀린 형식 |  |
| AA | 틀린 형식 | 틀린 형식 |  |
| (공백 하나) | 틀린 형식 | 틀린 형식 |  |
| (공백 둘) | 틀린 형식 | 틀린 형식 |  |
| (공백 셋) | 틀린 형식 | 틀린 형식 |  |
| A1 | 올바른 형식 | 올바른 형식 |  |
| A2 | 올바른 형식 | 올바른 형식 |  |
| a1 | 틀린 형식 | 틀린 형식 |  |
| a2 | 틀린 형식 | 틀린 형식 |  |
| E# | 틀린 형식 | 틀린 형식 |  |
| V% | 틀린 형식 | 틀린 형식 |  |
| V5 | 올바른 형식 | 올바른 형식 |  |
| VV3 | 틀린 형식 | 틀린 형식 |  |
| V33 | 틀린 형식 | 틀린 형식 |  |
| $% | 틀린 형식 | 틀린 형식 |  |
| M9\ | 틀린 형식 | 틀린 형식 |  |
| B7 | 올바른 형식 | 올바른 형식 |  |
| g8 | 틀린 형식 | 틀린 형식 |  |
| ㅎㅎㅎ | 틀린 형식 | 틀린 형식 |  |
| ㅎ2 | 틀린 형식 | 틀린 형식 |  |

* + 1. checkSeatsList 함수

1. 부분 설계

* 기존의 checkyourseat함수 대신 형식 검증, 배열 반환까지 수행하기 위해 만든 함수로 문자열을 입력받아 길이가 2 미만인지 우선 확인. 좌석 형식 상 문자열의 길이 2미만일 수는 없으므로 길이부터 틀리면 빈 배열을 반환.  
  다음으로 길이가 3 미만인지 확인. 맞으면 checkOneSeat함수를 호출해 형식 검증하고 True를 반환하면 좌석 한 개가 올바르게 입력된 문자열이므로 해당 문자열을 list라는 배열에 저장해서 반환, checkOneSeat함수가 False를 반환하면 빈 배열을 반환  
  문자열의 길이가 3 이상이면 ‘~’가 1개 포함되어 있는지 확인. 있으면 ~를 기준으로 문자열을 자르고, 잘린 문자열에 대해 checkOneSeat 함수로 형식 검증. 맞으면 firstseat에 저장된 앞 문자열의 영문+숫자, … , lastseat에 저장된 뒤 문자열 영문+숫자인 배열을 반환. checkOneSeat함수가 False를 반환하면 빈 배열을 반환.  
  문자열의 길이가 3 이사이고 ‘~’가 1개 포함하지 않고 ‘,’는 포함한다면(~~나 ~, 반드시 검증) ‘,’를 기준으로 문자열을 잘라서 list라는 배열에 담는다. 배열의 모든 인자를 checkOneSeat함수로 검증하고 false를 반환하는 인자가 있으면 list 배열을 비워서 반환하고 함수 종료. 한 개도 없다면 ‘,’를 기준으로 잘린 문자열 저장된 list배열 반환

1. 검사 방법

* 입력된 값을 인자로 checkSeatsList함수를 호출하고 반환된 배열의 내용을 출력하게 한다. 올바른 형식이면 좌석번호를 반환하고, 형식에 오류가 있으면 빈 배열을 반환한다.

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| (공백 한 칸) | [] | [] |  |
| a | [] | [] |  |
| a2 | [] | [] |  |
| A2 | [‘A2’] | [‘A2’] |  |
| A1~A3 | [‘A1’, ‘A2’, ‘A3’] | index out of range | 배열 범위 오류 |
| A1,A1 | [] | [‘A1’, ’A1’] | 첫 문자열에 대해 제대로 검증하지 못함 |
| A3,A3 | [] | [‘A3’, A3’] |
| A1,A2,A1 | [] | [] |  |
| A1,A3,A2,A2 | [] | [] |  |

오류 분석 : split()을 사용한 부분에서 오류가 나는데, split은 아무 조건이 주어지지 않았을 경우 공백을 기준으로 split함. 이 문자열에는 공백이 없어서 기존 코드대로 진행하면 lastseat[1]에 아무 것도 들어가지 않아서 lastseat[1]에 접근하려고 하면 오류가 발생한다.  
같은 좌석을 구분하는 기능이 동작하긴 하나 첫 좌석과 두번째 좌석이 같은 경우는 제대로 동작하지 않음

# TypeChecker 클래스 4차 설계

* + 1. checkMovieTitle 함수

1. 부분 설계

* ‘.’이 포함되지 않은 문자열에 오류가 발생하여, ‘.’이 없는 경우 바로 False를 반환하도록 설계 수정.

1. 검사 방법

* 3차 설계와 동일

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| 121 | AssertError | AssertError |  |
| 2.2 | AssertError | AssertError |  |
| 121vvv | False | False |  |
| vvvvv | False | False |  |
| (공백 1) | False | False |  |
| (빈 문자열) | False | False |  |
| 1.vvv | True | True |  |
| 1. ee | False | False |  |
| 1. e e | False | False |  |
| 11eee | False | False |  |
| 1.eeeeeeeeeeeeeeeeeee | True | True |  |
| 1.eeeeeeeeeeeeeeeeeeee | False | False |  |
| 1. (공백 1) | False | False |  |

* + 1. checkSeatsList 함수

1. 부분 설계 수정사항

* 굳이 문자열을 나누지 않아도 이후 코드를 진행하는데 문제가 없을 거라 판단, 아래의 코드에 split을 사용하지 않도록 수정하였다.  
  firstseat = list[0].split() -> firstseat = list[0]  
  lastseat = list[1].split() -> lastseat = list[1]  
  또한 첫번째 문자열과 두번째 좌석이 같은 경우를 제대로 구분하지 못해서 find대신 count를 사용하여 테스트하였다.  
  strseat[3:].find(i)>1-> strseat.count(i)>0

1. 검사 방법

* 3차 설계와 동일하고, 각 좌석에 대한 형식 검증은 checkOneSeat함수가 처리하므로 좌석 입력 자체가 틀린 경우에 관한 검사를 여러 번 진행하지는 않았다

1. Test Cases 결과 및 정리

|  |  |  |  |
| --- | --- | --- | --- |
| 입력(호출) | 예상 결과 | 실제 결과 | 비고 |
| A1 | [‘A1’] | [‘A1’] |  |
| A2 | [‘A2’] | [‘A2’] |  |
| A3 | [‘A3’] | [‘A3’] |  |
| A1~A5 | [‘A1’, ‘A2’, ‘A3’, ‘A4’, ‘A5’] | [‘A1’, ‘A2’, ‘A3’, ‘A4’, ‘A5’] |  |
| A5~A2 | [] | [] |  |
| A9~A7 | [] | [] |  |
| A7,B9 | [‘A7’, ‘B9’] | [‘A7’, ‘B9’] |  |
| A3,A3,B5 | [] | [] |  |
| A3,A3,A5, | [] | [] | 문자열 뒤의 ‘,’를 구분하는지 |
| A3~A4~A5 | [] | [] |  |
| A4~A5,B5 | [] | [] |  |
| A1,A1 | [] | [] |  |
| N5,N5 | [] | [] |  |
| N9~N10 | [] | [] |  |
| N1~N9 | [‘N1’, ‘N9’] | [‘N1’, ‘N9’] |  |
| e3 | [] | [] |  |
| e1~e3 | [] | [] |  |
| ,,, | [] | [] |  |
| , E6 | [] | [] |  |
| ,E9 | [] | [] |  |
| E7, | [] | [] |  |
| E7,E | [] | [] |  |
| E9,E9 | [] | [] |  |

# FileManager 클래스

* + 1. \_init\_함수

1. 부분 설계  
   파일 클래스의 객체를 만들면 Cardlist.csv, MovieList.csv ,ResecationList.csv, UserList.csv에서 데이터를 불러와 self.cardlist, self.userlist, self..movielist, self.reservationlist 에 데이터를 저장 해둠  
   -cardlist: 딕셔너리 형식으로 카드번호가 키 ,등록 여부가 value  
   -userlist: 딕셔너리 형식으로 유저ID가 키 [유저id, 유저 패스워드, 등록카드번호, 마일리지]의 딕셔너리가 valuie  
   -movielist: 딕셔너리 형식으로 예매코드형식에서 자석 부분을 제외한 것이 키 [상영 날짜, 영화코드, 영화이름, 시작 시간, 끝나는 시간, 관 정보, 좌석형태, A,B,C,,,열 정보 ]의 리스트가 value  
   -reservationlist: 2차원 리스트 형식으로 여러 리스트가 [회원여부,ID,예매코드,좌석정보,취소 여부]로 저장되어 있다.
2. 검사 방법:  
   각 리스트, 딕셔너리에 값이 의도한 바와 같이 담기는지 검사  
   각 csv의 값을 한 행마다 여러 경우의 열 값을 준 뒤 리스트의 담기는 값들을 확인 파일에서 불러올 때 모든 값들을 문자열로 담아 가져오므로 숫자로만 이루어 진 값도 문자열로 테스팅,  
    검사: csv를 불러 올 때 행 마다 맨 뒤에 데이터 2개의 값을 불러올 것을 그래서 행마다 인자가 더 많을 시 뒤에 두 값만 가져옴 예상
3. 검사
   * 1. Cardlist

|  |  |
| --- | --- |
| 입력, 예상 결과 | 실제 결과 |
| CardList.csv  cardnum,regist  testnum1,test1,testnum1,test1-1  예상결과:  {testnum1:test1-1} |  |
| CardList.csv  cardnum,regist  test1  testnum3,test2  예상결과:  에러 예상, 첫 줄의 값이 하나 이므로 | 값이 부족할경우 nan으로 들어가는 것을 확인 |
| CardList.csv  cardnum,regist  test1,testnum3,test2  예상결과:  {testnum3:test2} |  |
| CardList.csv  cardnum,regist  test1  testnum3,test2,testnum2  예상결과:  {test1:nan,test2:testnum2} | 에러 발생, 이전 행이 값의 개수가 부족 하였을 때, 다음행은 ?.. |
| CardList.csv  cardnum,regist  test1  testnum3,test2,testnum2,test02  예상결과:  에러 | 이전 행이 값의 개수가 부족 하였을 때는 다음행의 값이 꼭 2개 여야함 |
| CardList.csv:  cardnum,regist  test1  testnum3  예상 결과:  에러. | 이전 행의 개수가 부족 했을 때 다음행도 부족하면 둘다 nan으로 들어감 |
| CardList.csv:  cardnum,regist  test1  test2  test3  test4  test5  test6  예상 결과:  각 값의 value nan으로 들어감 |  |
| CardList.csv:  cardnum,regist  test1  testnum3  testnum2,test02  예상 결과:  {test1':nan,'test3':nan,'testnum2':test02} |  |
|  |  |
| -테스트에 계속 쓰일 CardList.csv 원본 | |
|  | |

* + 1. userlist

|  |  |
| --- | --- |
| 입력, 예상 결과 | 실제 결과 |
| userID,userpassword,registcard,mileage  u8s0e9r  입력값이 적을 때 1  예상 결과  {u8s0e9r:{userID:u8s0e9r,userpassword:nan,registcard:nan,mileage:nan}} |  |
| userID,userpassword,registcard,mileage  u8s0e9r,asdf  입력값이 적을 때 2  예상 결과  {u8s0e9r:{userID:u8s0e9r,userpassword:asdf,registcard:nan,mileage:nan}} |  |
| userID,userpassword,registcard,mileage  u8s0e9r,asdf,12312312444,12  예상결과  {u8s0e9r:{userID:u8s0e9r,userpassword:asdf,registcard:12312312444,mileage:12}} |  |
| userID,userpassword,registcard,mileage  u8s0e9r,asdf,12312312444,12,11244  123124,12aedf,12122,aad,ddd,ddd |  |
| userID,userpassword,registcard,mileage  uasdf,aaaa,qwer,asdf  u8s0e9r,asdf,12312312444,12,11244,2222  예상결과  에러: |  |
|  |  |
| 원본 UserList.csv | |
|  | |

* + 1. movielist
    2. reservationlist
    3. Init 함수 재설계  
       기존의 방식에서 데이터 값을 불러 올 때 각행의 열수가 정해진 형식과 다르면 제대로 파일형식이 맞지 않습니다 를 출력하고 종료하게  
       이 부분에서 파일을 불러올 때 header값을 주면 데이터 프레임의 모양이 고정이 되어 버버 불러올 때 header를 생략해주고 불러온 불러온 열의 개수가 의도한 형식이랑 맞는지 비교한 뒤 다시 헤더를 지정해주는 방식으로 재설계 데이터 자체가 어떠한 형식을 갖추어야 한다 라는 것은 정의 되어있지 않아 데이터 열수 체크는 들어가 있지만 데이터 가 어떠한 값을 가져야 한다는 의미의 형식 체크는 구현되어 있지 않음

1. 검사 목적 ,값이 부족하거나 값이 더 많을 형식에 맞지 않다며 제대로 출력 하는지
2. 검사방식: 각 파일의 맨 위의 header 부분은 삭제 예정이라 검증에 포함 되지 않고 검사에서는 첫 줄은 무조건 생략하고 검증
3. 검사

|  |  |
| --- | --- |
| Cardlist, 예상 결과 | 실제 결과 |
| cardnum,regist  204023801186,1  예상결과:cardlist  {204023801186:1} |  |
| cardnum,regist  예상결과:종료  파일 형식이 맞지 않습니다. |  |
| Cardnum,regist  112312312  예상결과: 종료  파일 형식이 맞지 않습니다. |  |
| cardnum,regist  testnum1,testnum2,asdf  예상결과: 종료  파일 형식이 맞지 않습니다 |  |
| cardnum,regist  testnum1,testnum2,asdf,asdfas  예상결과: 종료  파일 형식이 맞지 않습니다. |  |
| cardnum,regist  testnum1,testnum2,asdf,asdfas,asfdad,12312,asdfasd,asdasd,asdad22  예상 결과: 종료  파일 형식이 맞지 않습니다. |  |
| cardnum,regist  testnum1,testnum2  asdf,asdfas  asfdad,12312  2312,asdfasd  asdasd,asdad22  예상 결과:  {'testnum1': 'testnum2', 'asdf': 'asdfas', 'asfdad': '12312', '2312': 'asdfasd', 'asdasd': 'asdad22 |  |
| cardnum,regist  testnum1,testnum2  asdf,asdfas  asfdad  2312,asdfasd  asdasd,asdad22  예상 결과:  {'testnum1': 'testnum2', 'asdf': 'asdfas', 'asfdad':nan, '2312': 'asdfasd', 'asdasd': 'asdad22 |  |
| Cardlist | |
|  | |
| Userlist 예상결과 | 실제 결과 |
| userID,userpassword,registcard,mileage  예상결과  파일 형식이 맞지 않습니다. |  |
| userID,userpassword,registcard,mileage  aaa  예상결과  파일 형식이 맞지 않습니다. |  |
| userID,userpassword,registcard,mileage  aaa,1112,2222  예상결과  파일 형식이 맞지 않습니다. |  |
| userID,userpassword,registcard,mileage  userid,pass,123124124,1000  예상결과:  {'userid':{'userID':'userid','userpassword': 'pass','registcard':'123124124', 'mileage': '1000'}} |  |
| userID,userpassword,registcard,mileage  userid,pass,123124124,1000,12,11  asdaa,ddd,123124545,1111  예상결과  파일 형식이 맞지 않습니다 |  |
| userID,userpassword,registcard,mileage  userid,pass,123124124,1000  asdaa,ddd,123124545,1111,123123,11111,1111,22,22221,222,12 |  |
| Userlist 파일 원본 - | |
|  | |
| MovieList.csv 예상결과 | 실제 결과 |
| day,moviecode,moviename,starttime,finishtime,screen,seat,A,B,C,D,E,F,G,H,I,J  예상결과  파일 형식이 맞지 않습니다 | |
| day,moviecode,moviename,starttime,finishtime,screen,seat,A,B,C,D,E,F,G,H,I,J  20191202,AA,masdf  예상결과  파일 형식이 맞지 않습니다 | |
| day,moviecode,moviename,starttime,finishtime,screen,seat,A,B,C,D,E,F,G,H,I,J  20191202,AA,masdf,asdfa,asdfasd,12s,444,a,b,c,d,e,f,g,h,i,j  예상 결과  {'20191202AAasdfa': ['20191202', 'AA', 'masdf', 'asdfa', 'asdfasd', '12s', '444', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']} | |
| day,moviecode,moviename,starttime,finishtime,screen,seat,A,B,C,D,E,F,G,H,I,J  20191202,AA,masdf,asdfa,asdfasd,12s,444,a,b,c,d,e,f,g,h,i,j,1,2  예상 결과 : 열수가 더 많아서 에러  파일형식이 맞지 않습니다. | |
| MovieList.csv    예상 결과:  {20191020AAxxxx:[20191020,AA,movie1,0410m0550,screen1,8x8,0000,000,0000,0,00,00,000,00000,000,00000]이런식의 딕셔너리가 행의 개수만큼 담겨 있다  결과 | |
| ReservationList.csv | |
| member,userID,reservationcode,seats,cancel  1,aa  예상 결과 : 열수가 부족해서 에러.  파일형식이 맞지 않습니다. | |
| member,userID,reservationcode,seats,cancel  1,user11,202011AAdf012  예상 결과 : 열수가 부족해서 에러.  파일형식이 맞지 않습니다. | |
| member,userID,reservationcode,seats,cancel  1,user11,202011AAdf012,A1,0  예상 결과:  [['1', 'user11', '202011AAdf012', 'A1', '0']] | |
| member,userID,reservationcode,seats,cancel  1,user11,202011AAdf012,A1,0,11  예상 결과: 열수가 많아서  파일 형식이 맞지 않습니다. | |
| member,userID,reservationcode,seats,cancel  1,user11,202011AAdf012,A1,0,11,2,2,4,1,1,2,2  예상 결과: 열수가 많아서  파일 형식이 맞지 않습니다. | |
|  | |
| 4파일 모두 양식(열의 개수)가 맞아야 정상적으로 객체가 생성되는 것을 확인 | |

1. 통합 검사

검사 방법 : Ticketer.py 를 실행시키고 각 단계마다 한 번씩 입력해가며 테스트를 진행함.

사용자가 사용할 수 있는 기능별로 기능을 실행해가며 중간 과정 및 출력물을 검사함.

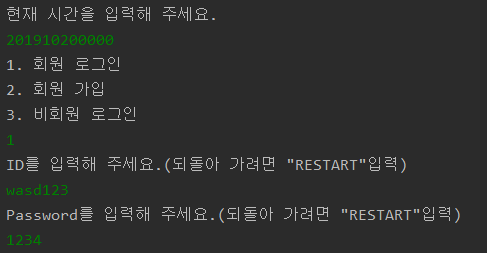
# 현재 시간 입력 기능 – 하정수

# 회원 가입 기능 – 하정수

# “EXIT”, ”RESTART”, ”BACK” 기능 – 하정수

# 영화 예매 기능(회원) -이재승

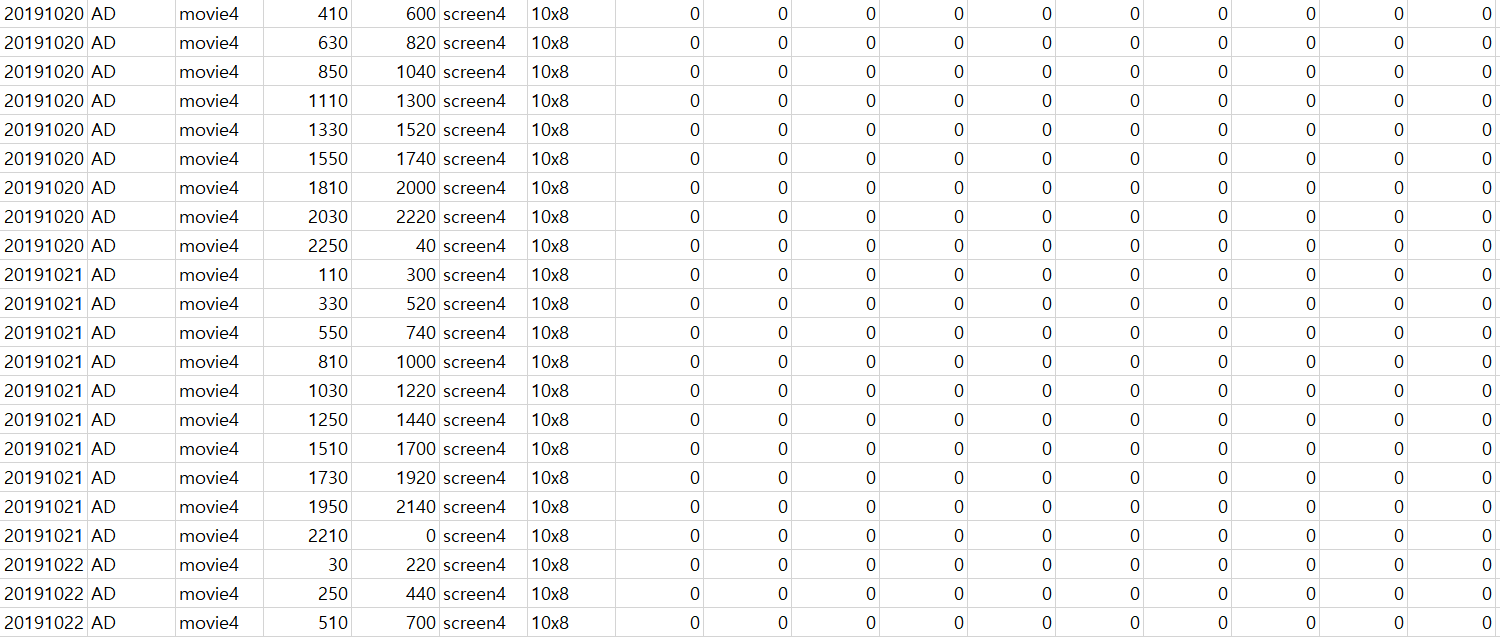
테스트 환경





마일리지 5000원

MovieList.csv

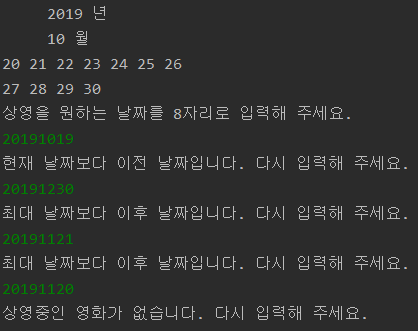
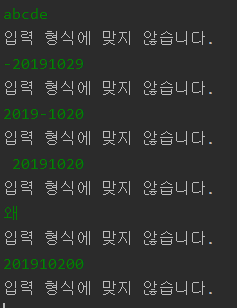


* Test Cases 및 결과 정리

Case 1.

영화 선택까지의 입력 ( 2019-10-20 ~2019-11-20 만 가능 )

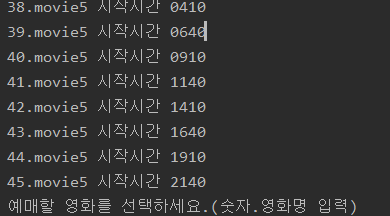
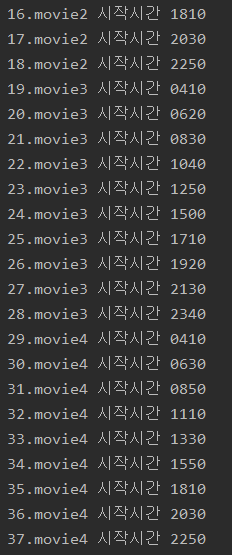
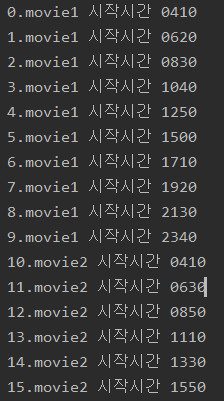
시간 입력 예외 처리

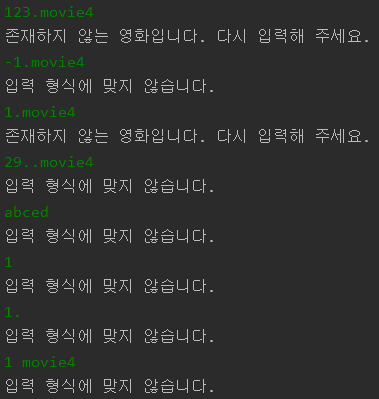
입력



영화 리스트



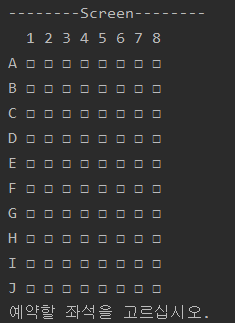
영화 선택 예외 처리



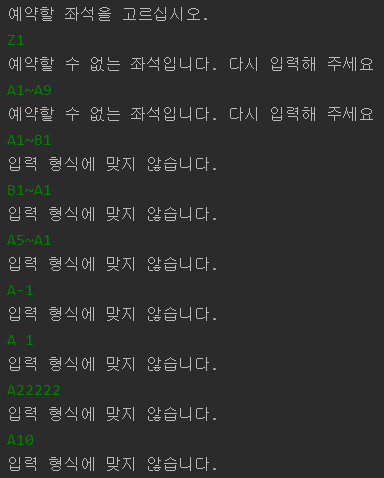
입력



좌석 표 출력 ( 10x8 )

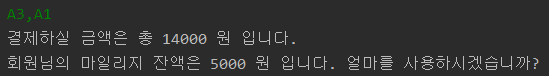


좌석 예외 처리

Z행 존재x

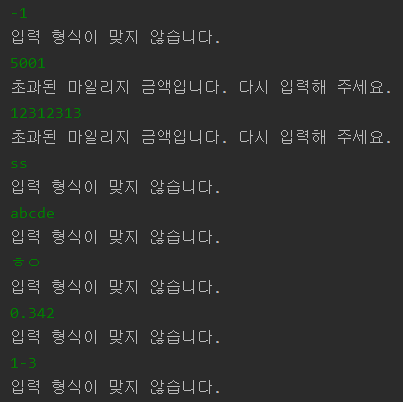
A9 존재x

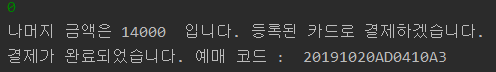
열 개수는 최대 9

입력  


12시 이전 영화이므로 7000x2 = 14000 원

마일리지 예외 처리





마일리지 : 5000 + 14000/10 = 6400

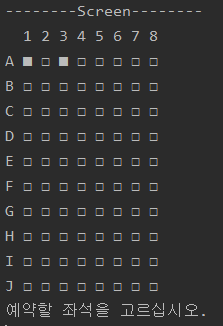


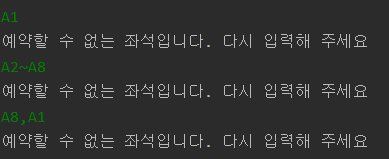
예매 내역

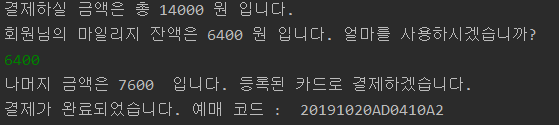


Case 2.

Case 1. 과 동일한 영화 선택



이미 예약된 좌석이 포함.



결제 금액 14000 – 6400 = 7600

마일리지 = 0 + 14000/10



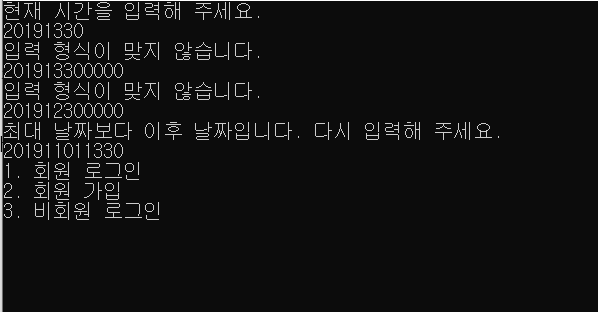
예매 내역

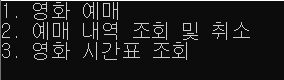
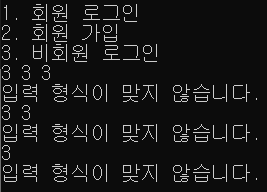


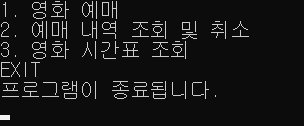
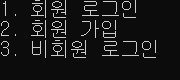
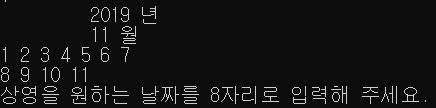
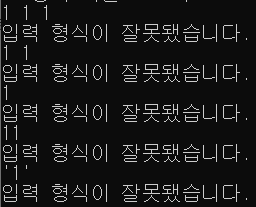
# 예매 내역 조회 기능(회원) – 조윤직

# 영화 시간표 조회 기능(회원) – 송재원

# 영화 예매 기능(비회원) – 박재영

목적 : 시간 입력이 제대로 이루어지는지 확인  
입력 : 20191330  
 201913300000  
 201911011330  
예상 결과 : 입력 형식이 맞지 않습니다.  
 최대 날짜보다 이후 날짜입니다. 다시 입력해주세요  
 최대 날짜보다 이전 날짜입니다. 다시 입력해주세요.  
 1. 회원 로그인  
 2. 회원 가입  
 3. 비회원 로그인   
실제 결과  


목적 : 비회원 예매 기능에 제대로 진입하는지 확인  
입력 : 333  
3 3 3  
3 3  
3 (공백 1)  
3  
예상 결과 :  
  
실제 결과 :  


목적 : 영화 예매 기능으로 제대로 진입하는지 확인  
입력 :  
1 1 1  
1 1  
1 (공백 1)  
11  
‘1’  
EXIT  
RESTART  
1  
예상 결과  
  
실제 결과:  


목적 : 시간 입력과 비회원 설정과 기능 입력은 제대로 되었지만 예매 날짜 입력이 제대로 되지 않았을 때, 오류 메시지를 정상적으로 출력하는지 확인  
입력 및 예상 결과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 입력 1 | 입력 2 | 입력 3 | 입력 4 |
| 시간 입력 | 201910251330 | 201910251340 | 201910251640 | 201910261750 |
| 비회원 설정을 위한 입력 | 3 | 3 | 3 | 3 |
| 기능 입력 | 1 | 1 | 1 | 1 |
| 예매 날짜 입력 | 20191330 | 20191024 | 20191127 | 20191190 |
| 예상 결과 | 입력 형식 오류. | 날짜 오류 : 현재 날짜 이전 | 날짜 오류 :  프로그램 정상 작동 날짜 이후 | 입력 형식 오류. |
| 이후 입력 | RESTART 🡪 EXIT | RESTART 🡪 EXIT | RESTART 🡪 EXIT | RESTART 🡪 EXIT |

실제 결과 :

|  |  |  |
| --- | --- | --- |
| 실제 결과 | | 비고 |
| 입력 1 |  | 정상 출력 정상 종료 |
| 입력 2 |  | 정상 출력 정상 종료 |
| 입력 3 |  | 정상 출력  정상 종료 |
| 입력 4 |  | 정상 출력  정상 종료 |

목적 : 시간 입력과 비회원 설정과 기능 입력과 예매 날짜 입력이 제대로 되었을 때 영화 제목이 제대로 입력되지 않았을 경우 오류 메시지를 정상 출력하는지 확인

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 입력 1 | 입력 2 | 입력 3 | 입력 4 | 입력 5 | 입력 6 |
| 시간 입력 | 201910251750 | 201910231730 | 201911010230 | 201911071230 | 201911011130 | 201910261220 |
| 비회원 설정 | 3 | 3 | 3 | 3 | 3 | 3 |
| 기능 입력 | 1 | 1 | 1 | 1 | 1 | 1 |
| 예매 날짜 | 20191119 | 20191101 | 20191103 | 20191109 | 20191107 | 20191030 |
| 영화 제목 | 2222 | 22.%$ | 60.ee | 41.movie42 | 41.moviee eeeeee eeeeee eeeeeee | 41. movie4 |
| 예상 결과 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. | 존재하지 않는 영화입니다. | 존재하지 않는 영화입니다. | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. |
| 이후 입력 | RESTART🡪EXIT | RESTART🡪EXIT | RESTART🡪EXIT | RESTART🡪EXIT | RESTART🡪EXIT | RESTART🡪EXIT |

실제 결과

|  |  |  |
| --- | --- | --- |
|  | 결과 | 비고 |
| 입력 1 |  | 정상 출력  정상 종료 |
| 입력 2 |  | 정상 출력  정상 종료 |
| 입력 3 | 강제 종료됨 | 강제 종료됨 |
| 입력 4 |  | 정상 출력  정상 종료 |
| 입력 5 |  | 정상 출력  정상 종료 |
| 입력 6 |  | 정상 출력  정상 종료 |

목적 : 시간 입력부터 예매할 영화 제목 입력까지 모든 과정에서 입력이 제대로 이루어졌으나 좌석 입력이 잘못되었을 경우 오류 메시지를 정상 출력하는지 확인

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 입력 1 | 입력 2 | 입력 3 | 입력 4 | 입력 5 |
| 시간 입력 | 201911030930 | 201911020920 | 201911010810 | 201910310710 | 201910300650 |
| 비회원 설정 | 3 | 3 | 3 | 3 | 3 |
| 기능 입력 | 1 | 1 | 1 | 1 | 1 |
| 예매 날짜 | 20191105 | 20191106 | 20191107 | 20191108 | 20191109 |
| 영화 제목 | 21.movie2 | 22.movie3 | 23.movie3 | 24.movie3 | 25.movie3 |
| 좌석 입력 | AA | 11 | A1~A3~A5 | A4,A6, | A4,B1~B5 |
| 예상 결과 | 입력 형식에 맞지 않습니다. | | | | |
| 이후 입력 | RESTART 🡪 EXIT | RESTART 🡪 EXIT | RESTART 🡪 EXIT | RESTART 🡪 EXIT | RESTART 🡪 EXIT |

실제 결과

|  |  |  |
| --- | --- | --- |
|  | 결과 | 비고 |
| 입력 1 |  | 정상 출력  정상 종료 |
| 입력 2 |  | 정상 출력  정상 종료 |
| 입력 3 |  | 정상 출력  정상 종료 |
| 입력 4 |  | 정상 출력  정상 종료 |
| 입력 5 |  | 정상 출력  정상 종료 |

목적 : 시간 입력부터 좌석 입력까지 모든 과정에서 입력이 제대로 이루어졌으나 카드 번호 입력이 잘못되었을 경우 오류 메시지를 정상 출력하는지 확인

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 입력 1 | 입력 2 | 입력 3 | 입력 4 | 입력 5 | 입력 6 |
| 시간 입력 | 201910212130 | 201910222130 | 201910220930 | 201910231020 | 201910241240 | 201911121450 |
| 비회원 설정 | 3 | 3 | 3 | 3 | 3 | 3 |
| 기능 입력 | 1 | 1 | 1 | 1 | 1 | 1 |
| 예매 날짜 | 20191024 | 20191025 | 20191026 | 20191027 | 20191028 | 20191114 |
| 영화 제목 | 30.movie3 | 31.movie3 | 32.movie3 | 33.movie4 | 34.movie4 | 11.movie2 |
| 좌석 | A1 | A2.B3 | A1~A5 | A7 | G8 | G7 |
| 카드번호 | 2040158842532 | aaaaaaa | 204015884 53 | 20401588425 | 123412341234 | 204047613901 |
| 예상 결과 | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. | 입력 형식에 맞지 않습니다. | 존재하지 않는 카드입니다. | 이미 등록된 카드번호입니다. |
| 이후 입력 | RESTART 🡪 EXIT | RESTART 🡪 EXIT | RESTART 🡪 EXIT | RESTART 🡪 EXIT | RESTART 🡪 EXIT |  |

실제 결과

|  |  |  |
| --- | --- | --- |
|  | 결과 | 비고 |
| 입력 1 |  | 정상 출력  정상 종료 |
| 입력 2 |  | 정상 출력  정상 종료 |
| 입력 3 |  | 정상 출력  정상 종료 |
| 입력 4 |  | 정상 출력  정상 종료 |
| 입력 5 |  | 정상 출력  정상 종료 |
| 입력 6 |  | 정상 출력  정상 종료 |

목적 : 시간 입력부터 좌석 입력까지 비회원 예매의 모든 과정이 정상적으로 이루어졌을 때 기능 선택 메뉴로 돌아오며, ReservationList.csv파일에 예매 내역이 정상적으로 저장되는지 확인

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 입력 1 | 입력 2 | 입력 3 | 입력 4 | 입력 5 |
| 시간 입력 | 201911101310 | (없음) | 201911121450 | (없음) | 201911142230 |
| 비회원 설정 | 3 | 3 | 3 |
| 기능 입력 | 1 | | | | |
| 예매 날짜 | 20191113 | 20191113 | 20191115 | 20191116 | 20191119 |
| 영화 제목 | 9.movie1 | 10.movie1 | 8.movie1 | 12.movie2 | 13.movie2 |
| 좌석 | A1~A5 | B1,B5 | H1 | C1~C4 | D1,E1 |
| 카드번호 | 204026724978 | 204045744463 | 204019914162 | 204010021351 | 204042113002 |
| 예상 결과 | ReservationList.csv에 저장됨. | | | | |
| 이후 입력 | 다음 좌석 예매 | RESTART 🡪 EXIT | 다음 좌석 예매 | RESTART 🡪 EXIT | RESTART 🡪 EXIT |

실제 결과

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 입력 1 | 입력 2 | 입력 3 | 입력 4 | 입력 5 |
| 결과 |  | | | | |
| 비고 | 예매코드 정상 출력 | 예매코드 정상 출력 정상 종료 | 예매코드 정상 출력 | 예매코드 정상 출력  정상 종료 | 예매코드 정상 출력  정상 종료 |

# 3.8 예매 내역 조회 기능(비회원) – 조윤직

# 3.9 영화 시간표 조회 기능(비회원) -송재원