1. 개요

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

1. 단위 검사

# main(Ticketer?) 클래스

* + 1. main 함수

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 일 때 – 아무 것도 진행하지 않고 입력을 대기한다.

2-2. MM.getMI().getwhere != 0 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 일 때 – 아무 것도 진행하지 않고 입력을 대기한다.

3-2. MM.getMI().getwhere == 2 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 | ‘’ |

# menuInfo 클래스

* + 1. \_\_init\_\_ 함수

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. getmenuNum, getisMember, getwhere 함수

단순히 return만을 수행하는 getter 함수이기 때문에 검사하지 않음.

* + 1. setMI 함수

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 |  |

# 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 실행할 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”) | 출력x | 출력x | 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. \_\_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. 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. 한 좌석 당 가격을 7000으로 계산하여 “결제하실 금액은 총 ㅁㅁ원 입니다.” 을 출력함.

3-2. 한 좌석 당 가격을 10000으로 계산하여 “결제하실 금액은 총 ㅁㅁ원 입니다.” 을 출력함.

4-1. “회원님의 마일리지 잔액은 ㅁㅁ원 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.

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 : ValueError

실제 결과 : 부분 설계 3-2번 결과, 부분 설계 4-1번 결과

비고 : print\_seat 함수 수정 필요

* + 1. menu43141 함수

인자로 전달받은 input의 자료형을 str로 assert문을 걸어준 상태.

1. 부분 설계
2. 검사 방법
3. Test Cases 및 결과 정리
   * 1. menu43142 함수
4. 부분 설계
5. 검사 방법
6. Test Cases 및 결과 정리
   * 1. print\_10days 함수
7. 부분 설계
8. 검사 방법
9. Test Cases 및 결과 정리
   * 1. printday\_movie 함수
10. 부분 설계
11. 검사 방법
12. Test Cases 및 결과 정리
    * 1. print\_seat 함수
13. 부분 설계
14. 검사 방법
15. Test Cases 및 결과 정리
    * 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

# FileManager 클래스

1. 통합 검사

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

사용자가 사용할 수 있는 기능(현재시간 입력, 로그인, 회원 가입, 비회원 로그인, 영화 예매, 예매 내역 조회, 영화 시간표 조회)별로 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

# 예매 내역 조회 기능(회원) – 조윤직

# 영화 시간표 조회 기능(회원) – 송재원

# 영화 예매 기능(비회원) – 박재영

# 3.8 예매 내역 조회 기능(비회원) – 조윤직

# 3.9 영화 시간표 조회 기능(비회원) -송재원