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

* 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

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.

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)

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. 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 |  |
| Menu4311(“123”) | AssertionError | AssertionError |  |
| self.\_\_now\_time = “201913111200”  Menu4311(“20191230”) | AssertionError | AssertionError |  |
| self.\_\_now\_time = “000012111200”  Menu4311(“20191230”) | AssertionError없음 | AssertionError없음 |  |
| self.\_\_now\_time = 123  Menu4311(“20191230”) | AssertionError | AssertionError |  |
| self.\_\_now\_time = “000012111200”  Menu4311(123) | AssertionError | AssertionError |  |
| self.\_\_now\_time = “2019121112000”  Menu4311(“20191230”) | AssertionError | AssertionError |  |
| 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.  Menu4311(“20191110”) | 부분설계의 2번 결과 | 부분설계의 2번 결과 |  |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재함.  Menu4311(“20191110”) | 부분설계의 2번 결과 | 부분설계의 2번 결과 |  |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재함.  Menu4311(“20191230”) | 부분 설계의 4번 결과 | 부분 설계의 4번 결과 |  |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재하지 않음.  Menu4311(“20191230”) | 부분 설계의 3번 결과 | 부분 설계의 3번 결과 |  |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재하지 않음.  Menu4311(“20191111”) | 부분 설계의 3번 결과 | 부분 설계의 3번 결과 |  |
| self.\_\_now\_time = “201911111200”  해당 날짜의 영화 리스트 존재함.  Menu4311(“20191111”) | 부분 설계의 4번 결과 | 부분 설계의 4번 결과 |  |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재하지 않음.  Menu4311(“20181229”) | 부분 설계의 2번 결과 | 부분 설계의 2번 결과 |  |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  Menu4311(“20181229”) | 부분 설계의 2번 결과 | 부분 설계의 2번 결과 |  |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재하지 않음.  Menu4311(“20190101”) | 부분 설계의 3번 결과 | 부분 설계의 3번 결과 |  |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  Menu4311(“20190101”) | 부분 설계의 4번 결과 | 부분 설계의 4번 결과 |  |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  Menu4311(“201901010”) | 부분 설계의 1번 결과 | 부분 설계의 1번 결과 |  |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  Menu4311(“2019-01-01”) | 부분 설계의 1번 결과 | 부분 설계의 1번 결과 |  |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  Menu4311(“abcdefg”) | 부분 설계의 1번 결과 | 부분 설계의 1번 결과 |  |
| self.\_\_now\_time = “201901010000”  해당 날짜의 영화 리스트 존재함.  Menu4311(“0.123456”) | 부분 설계의 1번 결과 | 부분 설계의 1번 결과 |  |

* + 1. menu4312 함수

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

# TypeChecker 클래스

# 2.5 FileManager 클래스