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

1. 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. Savefile.함수
   1. 부분 설계  
      파일을 저장하는 함수 데이터를 저장. 하는 함수
   2. 검사 방식:  
      이 함수는 입력 값,인자 값이 없으므로 통합검사에서 파일이 제대로 저장이 되어지는 지를 확인
2. dupli\_checkID 함수
   1. 부분 설계  
      id형식에 맞는 문자열이 인자로 들어와 이 id가 유저 리스트에 있는지 없는지 를 bool형식으로 출력
   2. 검사 방식:  
      인자 값을 주어
3. dupli\_checkCARDNUM함수
   1. 부분설계  
      card번호 형식에 맞는 문자열이 인자로 들어와 이 카드번호가 존재하지 않는지(1리턴) 카드번호가 이미 등록되어 있는 카드인지(‘2’리턴) 존재하지만 등록되어 있지 않은 카드는 ‘1’으로 한다.
   2. 검사 방식
4. join\_user함수
   1. 부분설계  
      이미 회원 가입 할수 있다고 검증된 id,password,card번호를 입력하면 userlist에그 유저 정보를 추가, card리스트에도 등록 처리 해줌
   2. 검사방식
5. getuser함수
   1. 부분설계  
      다른 함수에서 사용하기 위해 유저의 id와 비밀번호를 입력하면 그 유저의 정보가 담긴 리스트가 리턴
   2. 검사 방식
6. day\_movielist
   1. 부분설계  
      날짜와 시간을 문자열 인자로 주면 그 날짜에 하는 아직 지나지않은 영화를 리스트에 담아 리턴해줌
   2. 검사방식
7. seats\_to\_list
   1. 부분설계  
      문자열로 좌석의 정보를 받아 그것을 좌석을 하나씩 담은 리스트로 바꿔줌
   2. 검사방식
8. bookmovie함수
   1. 부분설계  
      isuser,id,choice\_movie,strseat를 인자로 받음 isuser:지금 이 사용자가 회원인지(1), 비회원 인지(0),id: 유저의 id, choice\_movie: 고른 지금 영화의 정보가 담긴 리스트 이값이 변하면,movielist도 변함, movie 리스트에 이 좌석이 예매됬음을 수정, 예약리스트에 이 정보를 추가
   2. 검사방식
9. getReservation함수
   1. 부분설계  
      유저의 id나 예매코드를 인자로 입력하면 해당 예약정보의 reservationlist안에 인덱스를 리스트에 담아 리턴해줌
   2. 검사방식
10. book\_cancel함수
    1. 부분 설계  
       예매 코드를 입력받으면 영화 내역을 취소해주는 함수
    2. 검사 방식