Elokuvateatterin varausjärjestelmä

Työn aiheena on elokuvateatterin varausjärjestelmä, jossa:

1. Ylläpitäjä pystyy luomaan/muokkaamaan/poistamaan saleja, elokuvia ja näytöksiä
2. Käyttäjä pystyy selaamaan näytöksiä ja tekemään varauksia niihin, jos vapaita paikkoja on
3. Tiedot tallennetaan JSON-formaattiin, josta ne ladataan käyttöön sovelluksen käynnistyessä

Käytössä ovat seuraavat luokat (jotka ovat omissa tiedostoissaan):

1. Application (itse sovellus, hallitsee lähinnä kirjautumista)
2. Theater (Pääosa varsinaisesta ohjelmasta, joka hyödyntää muita luokkia)
3. CinemaHall (Salien luokka)
4. Films sekä Film (Elokuvien luokat. Films on luokka, joka sisältää luokan Film objekteja)
5. Settings (Yhteen paikkaan kerättynä ohjelman komennot ja pari muuta juttua. Tämän toteutuksen järkevyydestä en ole aivan varma)

Lisäksi löytyvät seuraavat tiedostot:

1. Input_validation (Lähinnä käyttäjän syötteiden pyytämiseen ja käsittelyyn)
2. Salit.json (tiedontallennusta)
3. Films.json (tiedontallennusta)

Funktioita

Application-luokka

```
def select_user(self):
```

Käytetään kirjautumisessa joko asiakkaana tai omistajana.

```
def __admin_login_passcheck(self):
```

Vain oikealla salasanalla (123) pääsee sisään omistajana. Tämä ei hyödynnä mitään salauksia tai muuta monimutkaisempaa.

```
def exit_admin_mode(self):
```

Määrittää admin-tilan epätodeksi, jolloin silmukka palaa takaisin kirjautumiskohtaan.

```
stop_execution(self):
```

Tämän jälkeen komentoja ei enää kysellä, vaan ohjelman suoritus palaa Applicationin sisältä takaisin main-funktioon, eli käytännössä ohjelma sulkeutuu.

```
def __execute_commands(self):
```
Riippuen millä käyttäjällä kirjauduttiin sisään, tämä funktio lähettää käyttäjän silmukkaan, jossa suoritetaan käyttäjäkohtaisia komentoja (admin/customer).

```
def execute_customer_commands(self):
```
Tämä silmukka suorittaa asiakaskomentoja.

```
def execute_admin_commands(self):
```
Tämä silmukka suorittaa omistajan komentoja.

Theater-luokka

```
def __load_cinema_halls(self, path: str):
```
Ladataan elokuvasalien tiedot JSON-tiedostosta (Tiedoston puuttuessa se luodaan tyhjänä).

```
def __load_json_data_from_file(self, path: str):
```
Avustaa yllä olevaa (__load_cinema_halls) funktiota.

```
def save_file(self):
```
Tallentaa elokuvasalit tiedostoon.

```
def cinema_halls_editing_menu(self):
```
Silmukka salien muokkaamiskomentoja varten.

```
def add_cinema_hall(self, manual: bool = True, name: str = "Ei nimeä", seats: int = 0, shows: list = [], loading_from_file = False):
```
Lisää joko suoraan annetuilla arvoilla (esim. tiedostosta ladatessa) uuden salin. Tai vaihtoehtoisesti sali luodaan pyytämällä käyttäjältä arvot.

```
def edit_cinema_hall(self):
```
Pyytää muokattavan salin numeron ja tarjoaa sille muokkausoperaatioita.

```
remove_cinema_hall(self):
```

Pyytää poistettavan salin numeron ja poistaa sen.

```
def confirm_removal_of_hall_with_shows(self) -> bool:
```
Jos poistettavaksi annetussa salissa on näytöksiä, kysytään varmistus poistolle.

```
def __choose_cinema_hall_number(self) -> int:
```
Käytetään esim. yllä olevissa funktioissa määrittämään muokattava tai poistettava sali.

```
def print_cinema_halls(self):
```
Tulostaa kaikki salit.

```
def print_film_selection(self):
```
Tulostaa kaikki elokuvat.

```
def films_editing_menu(self):
```
Siirtyy elokuvien käsittelysilmukkaan.

```
def shows_editing_menu(self):
```
Silmukka, jolla hallitaan näytöksiä.

```
def print_all_shows(self):
```
Tulostetaan kaikki näytökset.

```
def print_halls_and_shows_in_readable_format(self, indexed_list: list):
```
Data on tallessa tiedostoissa (ja muuttujissa) hieman käyttäjäepäystävällisessä muodossa, joten tällä ja parilla apufunktiolla tulostetaan selkokielisesti olennaiset tiedot näytökselle.

```
def edit_shows_in_a_hall(self):
```
Muokkaa tietyn salin näytöksiä.

```
def browse_shows(self):
```
Silmukka asiakkaan tarpeisiin, näytösten selaamista varten.

```
def seat_reservation(self, indexed_list, index):
```
Varaa paikkoja näytöksestä.

```
def reserve_seats_in_show(self, show_info: dict):
```
Silmukka, jossa selvitetään asiakkaan varaustarpeet ja suoritetaan ne, jos vapaiden paikkojen määrä sen mahdollistaa.

```
def get_and_print_indexed_list_and_chosen_index(self, show_list: list) -> list:
```
Ehkä turhan sekava funktio, joka apufunktioita hyödyntäen käsittelee annetun listan niin, että siinä olevat saavat indeksit (joita hyödynnetään komentojen kohdistamisessa) sekä kysyy käyttäjältä numeron (näistä edellämainituista indekseistä), jonka perusteella toiminnot suoritetaan tietylle näytökselle.

```
def choose_show_by_date(self):
```
Käytetään näytösten etsimiseen päivämäärän perusteella.

```
def choose_from_todays_shows(self):
```
Tulostaa kuluvan päivän näytökset ja pyytää valitsemaan niistä esim. yllä olevan get_and_print_indexed_list_jnejnejne –funktion avulla.

```
def choose_from_next_7_days_shows(self):
```
Sama kuin yllä, mutta tulostaa kuluvan päivän + viikon eteenpäin näytökset.

```
def get_user_show_choice(self, largest_index: int) -> int:
```
Kysyy käyttäjältä näytöksen, johon hän haluaa varata paikkoja.

```
def add_indices_to_a_list_of_shows(self, list: list) -> list:
```
Käytetään näytöslistojen indeksointiin muissa funktioissa.

```
def get_shows_from_all_halls(self):
```
Palauttaa listan kaikkien salien kaikista näytöksistä.

```
def sort_list_of_shows(self, show_list):
```
Palauttaa alkamispäivämäärän (ja ajan) mukaan lajitellun kopion annetusta näytöslistasta.

```
def get_sorted_list_limited_by_days(self, days: int) -> list:
```
Aikaisempien funktioiden käyttämä apufunktio tulostettavien näytösten rajaamiseen nykyhetkestä x päivää tulevaisuuteen. Palauttaa listan

```
def get_list_of_shows_for_given_date(self, date: datetime.date):
```
Sama kuin yllä, mutta näytökset rajattu tiettyyn päivään.

```
def limit_list_of_shows_to_chosen_date(self, shows: list, chosen_date: datetime.date)
-> list:
```
Apufunktio yllä olevalle. Tämä tekee sen varsinaisen työn näytösten rajaamisessa päivämäärään.

```
def limit_list_of_shows_to_number_of_days_from_today(self, shows: list, max_days:
int):
```
Sama kuin yllä, mutta päivämäärän sijaan rajataan nykyhetkestä annettu määrä päiviä eteenpäin.

CinemaHall-luokka

```
def form_dictionary_from_self(self): # For JSON format saving in a file
```
Muodostaa CinemaHall objektista sanakirjan, jotta sen saa dumpattua JSON-tiedostoon helposti.

```
def create_new_hall_from_input(self):
```
Luodaan uusi sali käyttäjän antamista arvoista.

```
def change_cinema_hall_name(self):
```
Muuttaa salin nimen.

```
def change_cinema_hall_seats(self):
```
Muuttaa salin istumapaikkojen määrää.

```
add_show(self):
```
Lisää näytöksen saliin käyttäjän antamien arvojen mukaan.

```
def generate_unique_show_id(self):
```
Luo näytökselle uniikin ID:n (salikohtaisesti uniikin).

```
def reserve_seats_in_show(self, number_of_seats: int, show_id: int):
```
Varaa paikkoja näytöksestä.

```
def edit_shows(self):
```
Silmukka näytösten muokkaamiseen.

```
def remove_show(self):
```
Poistaa näytöksen.

```
def get_shows(self):
```
Palauttaa salin näytökset.

```
def print_shows(self):
```
Tulostaa salin näytökset.

```
def get_number_of_available_seats(self, show_reservations: int):
```
Palauttaa näytöksessä olevien vapaiden paikkojen määrän.

```
def get_name(self):
```
Palauttaa salin nimen.

```
def get_number_of_seats(self):
```
Palauttaa salin istuinpaikkojen määrän.

Films-luokka

```
def __load_films(self, path: str):
```
Lataa elokuvat tiedostosta.

```
def save_file(self):
```
Tallentaa elokuvat tiedostoon.

```
def add_film(self, manual: bool = True, name:str = "Ei nimeä", director: str = "Ei ohjaajaa", runtime: int = 0, required_age: int = 0, id: int = 0, loading_from_file = False):
```
Lisää elokuvan joko käyttäjän antamilla arvoilla tai sitten funktiokutsun mukana tulevilla arvoilla (esim. tiedostosta ladatessa)

```
def edit_films(self):
```
Elokuvien muokkaussilmukka.

```
def edit_film(self):
```
Silmukka yksittäisen elokuvan muokkaamiseen.

```
def remove_film(self):
```
Poistaa elokuvan.

```
def get_all_film_ids(self) -> list:
```
Palauttaa listan kaikista elokuvien ID:istä.

```
def get_film_name_from_id(self, id: int) -> str:
```
Palauttaa annetulla ID:llä olevan elokuvan nimen.

```
def film_index_exists(self, film_index) -> bool:
```
Tarkistaa onko elokuva annetulla indeksillä olemassa (esim. poistamista ja muokkausta varten)

```
def print_films(self):
```
Tulostaa kaikki elokuvat.

```
def select_film_for_show(self):
```
Apufunktio elokuvan lisäämiseksi näytökselle.

Film-luokka

```
def get_age_limit(self):
```
Palauttaa ikärajan.

```
def get_name(self):
```
Palauttaa elokuvan nimen.

```
def get_id(self):
```
Palauttaa elokuvan ID:n

```
def create_unique_id(self):
```
Luo uniikin ID:n elokuvalle, pohjautuen jo olemassa oleviin.

```
def change_name(self):
```
Muuttaa elokuvan nimeä.

```
def change_director(self):
```
Muuttaa ohjaajan nimeä.

```
def change_runtime(self):
```
Muuttaa elokuvan kestoa.

```
def change_required_age(self):
```
Muuttaa ikärajaa.

```
def get_runtime(self):
```

Palauttaa elokuvan keston.

```
def edit_film(self):
```
Silmukka elokuvan muokkaamiseen.

```
def form_dictionary_from_self(self): # For JSON
```
Jälleen muutetaan datan muotoa JSON-yhteensopivaksi.

Settings-luokka

Tämä tiedosto sisältää lähinnä admin-salasanan, päiväämäärille annettavan maksimivuoden, tiedostopolut JSONeille sekä tulosteita käyttäjälle ja sanakirjat, joissa numero ja funktion nimi.

Esimerkkinä:

```
self.__customer_commands = {
                                "0": "self.stop_execution",
                                "1": "self.theater.print_cinema_halls",
                                "2": "self.theater.browse_shows",
                                "9": "self.select_user"
        }
```
Kun käyttäjä antaa Application-luokan asiakaskomentojen silmukassa arvon 0, kutsutaan funktiota self.stop_execution(), joka muuttaa Application-objektin self.__execution_on muuttujan Falseksi, joka taasen katkaisee kyseisen while-silmukan seuraavan kierroksen alussa.

Input_validation.py

Tämä tiedosto sisältää apufunktioita käyttäjältä arvojen pyytämiseksi ja niiden sisällön tarkistamiseksi sekä muutaman muun funktion, joita käytetään matkan varrella, esim. Datetime-objektien muuntamiseksi merkkijonoiksi ja toisin päin.

En usko, että olen onnistunut näillä funktioilla estämään kaikkia ohjelman kaatavia virheellisiä arvoja, joita käyttäjä voi tuputtaa sisään, mutta edes osan.

Kirjastot

Ohjelma käyttää kirjastoja datetime ja json, jotka kuuluvat Pythonin oletuskirjastoon eli ulkoisia riippuvaisuuksia ei sen suuremmin ole.