HÁZI FELADAT

Programozás alapjai 3

P3MDB

Élő Zsófia Anna
MAGH42

2023.11.28.

DOKUMENTÁCIÓ

FONTOSABB ALGORITMUSOK

Az alapvető objektumorientált függvényeken kívül (setterek, getterek, konstruktorok stb), amelyekkel minden osztály sajátosan rendelkezik, a P3MDB még ezeket a függvényeket használja:

Film.java és User.java

Filmet és felhasználót megvalósító osztályok, adattagjaik:

Film.Java:	**User.java:**
String title	String username
int releaseYear	String password
int duration	int birthyear
double rating	int numofmovies
int numOfDVDs	String moviesseen
String category	
String description	

Az adattagokon (privát adattagok) kívül ez a két osztály csak konstruktorokkal (paraméter nélküli, paraméteres, másoló) rendelkeznek, valamint getterekkel és setterekkel. Mindkettő osztály szerializálható, a lementés és visszaolvasás kedvéért.

FilmDB

Listába, könnyen kezelhető adatbázisba sorolja a filmeket, összekapcsolja az eltárolt filmeket a megjeleníthető táblázatukkal.

Adattagjai a beolvasott filmek, valamint a filmek táblázata:

- **`List<Film> films`**
- **`FilmTableModel filmTable`**

FilmDB osztály is rendelkezik konstruktorral, setterekkel, getterekkel, valamint:

`public void saveToFile(){…}`

> Szerializálással megnyitja a „films.libdat” nevű fájlt (a projekt mappájában megtalálható), majd lementi ide a filmeket. Ha gond történt volna fájlkezelés közben, arról tájékoztatja a felhasználót.

`public static FilmDB readFromFile(FilmDB filmdb){…}`

> Szerializálással megnyitja a „films.libdat” nevű fájlt (a projekt mappájában megtalálható), majd beolvassa innen a filmeket. Ha gond történt volna fájlkezelés közben, arról tájékoztatja a felhasználót.

`public boolean orderValid(Film f) {…}` és `public boolean order(Film f){…}`

> Dvd rendelést megvalósító függvények,utóbbi meghívja az előbbit, ellenőrzi az előbbi, hogy raktáron van-e még a filmből, majd ha igen, leadja a rendelést.

`public RowSorter<FilmTableModel> filmSearch(String s) {…}`

> Keresést valósítja meg a film címei alapján, RowSorter swing osztállyal.

`public boolean movieAdd(…) {…}`

> Ellenőrzi, hogy helyes adattagokat kapott-e, majd, ha igen, hozzáadja az új filmet az adatbázishoz. Igazzal tér vissza, ha sikeres volt a művelet, hamis, ha meghiúsult.

`public boolean movieEdit(…) {…}`

> Ellenőrzi, hogy helyes adattagokat kapott-e, majd, ha igen, módosítja a szintén paraméterként megkapott filmet az adatbázisban. Igazzal tér vissza, ha sikeres volt a művelet, hamis, ha meghiúsult.

`public boolean movieRemove(Film f) {…}`

> Eltávolítja az adatbázisból a paraméterként átvett filmet. Igazzal tér vissza, ha sikeres volt a művelet, hamis, ha meghiúsult.

public boolean filmFormat(…) {…}

Leellenőrzi, hogy helyesek-e a kapott Film adattagok, például megfelelő intervallumon belül találhatóak-e, értékelés 0 és 10 közötti szám-e, Dvd-k száma nem nulla-e stb. Igaz, ha megfelelőek a formátumok, hamis, ha helytelenek.

UserDB

Listába, könnyen kezelhető adatbázisba sorolja a felhasználókat, összekapcsolja az eltárolt felhasználókat a megjeleníthető táblázatukkal. FilmDB-vel szorosan megegyező, csak felhasználókkal.

Adattagjai a beolvasott felhasználók, valamint a felhasználók táblázata:

- **List<User> users**
- **UserTableModel userTable**

UserDB osztály is rendelkezik konstruktorral, setterekkel, getterekkel, valamint:

public void saveToFile(){…}

Szerializálással megnyitja a „users.libdat” nevű fájlt (a projekt mappájában megtalálható), majd lementi ide a felhasználókat. Ha gond történt volna fájlkezelés közben, arról tájékoztatja a felhasználót.

public static UserDB readFromFile(Userdb userdb){…}

Szerializálással megnyitja a „users.libdat” nevű fájlt (a projekt mappájában megtalálható), majd beolvassa innen a felhasználókat. Ha gond történt volna fájlkezelés közben, arról tájékoztatja a felhasználót.

public boolean loginSuccess(String usern, String pass) {…}

Bejelentkezést segítő függvény, a paraméterként átvett jelszó-felhasználónév párossal azonosít egy felhasználót, ha sikerül, igazzal tér vissza, hamis, ha nem található ilyen nevű felhasználó ilyen jelszóval.

public RowSorter<UserTableModel> userSearch(String s) {…}

Keresést valósítja meg a felhasználók felhasználónevei alapján, RowSorter swing osztállyal.

public boolean userAdd(…) {…}

Ellenőrzi, hogy helyes adattagokat kapott-e, majd, ha igen, hozzáadja az új felhasználót az adatbázishoz. Igazzal tér vissza, ha sikeres volt a művelet, hamis, ha meghiúsult.

public boolean userEdit(…) {…}

Ellenőrzi, hogy helyes adattagokat kapott-e, majd, ha igen, módosítja a szintén paraméterként megkapott felhasználót az adatbázisban. Igazzal tér vissza, ha sikeres volt a művelet, hamis, ha meghiúsult.

public boolean userRemove(Film f) {…}

Eltávolítja az adatbázisból a paraméterként átvett User-t. Igazzal tér vissza, ha sikeres volt a művelet, hamis, ha meghiúsult.

```
public boolean userFormat(…) {…}
```

Leellenőrzi, hogy helyesek-e a kapott Film adattagok, például megtekintett filmek száma nem negatív stb. Igaz, ha megfelelőek a formátumok, hamis, ha helytelenek.

FilmTableModel és UserTableModel

Mindkét osztály AbstractTableModel-t valósít meg, ezek segítségével jelennek meg a felhasználók és filmek táblázatai, rendelkeznek a táblázatok fejléceivel, valamint az adatbázis felhasználóival/filmeivel. Mivel AbstractTableModel-ek a következő methodok implementálása kötelező: Alapvető konstruktorokon és getter/settereken kívül még a következőek a fontos algoritmusaik: getValueAt(), setValueAt(), getColumnCount(), getRowCount(), isCellEditable(), getColumnClass(), getColumnName().

Mindkét osztályon belül megvalósulnak az egyszerű addFilm / addUser és removeFilm / removeUser függvények, ezek egyszerűen csak elvégzik a táblázat listáján a megfelelő műveletet, majd fireTableDataChanged()-del frissítik a táblát.

```
public RowSorter <FilmTableModel> top10() {…}
```

FilmTableModel függvénye, feladata egy olyan szűrő előállítása, amely a filmek listájából kiszedi a 10 legmasabb értékelésű filmet. Megkapja a sortFilmsByRating() függvénytől a top 10 legmagasabb értékelésű filmet, majd a táblázaton átszűri, hogy csak ezek jelenjenek meg. include-dal. Visszatér a RowSorter szűrővel.

```
private List<Film> sortFilmsByRating() {…}
```

Comparatorral elvégzi az eredeti filmek List-en a rating alapján való szűrést, majd csak az első 10-et hagyja meg, Visszatér ezen filmek listájával.

UserFrame, LoginFrame és FilmFrame

Mindhárom osztály extends JPanel, ezekkel az osztályokkal lehet az inputot megvalósítani, akkor hívódnak meg, hogyha egy új filmet vagy felhasználót akarunk hozzáadni/módosítani, vagy bejelentkezni. Itt lehet beírni a megfelelő paramétereket. Mindhárom rendelkezik egy-egy JPanel adattaggal, valamint bevitelnek és attribútumoknak megfelelően JTextField-ekkel.

SearchMovies és SearchUsers

Keresés, kereső mezőt megvalósító JTextField osztályok. A filmek és a felhasználók nézetek rendelkeznek velük. Mindkettő adattagja az adatbázis felhasználói / filmjei. A MainFrame pakolja őket bele a megfelelő nézet JPanelbe, ahol insertUpdate(), removeUpdate() és changedUpdate() függvényekkel dinamikusan lehet keresni a táblázatok sorai között.

MainFrame

Az egész alkalmazás megjelenéséért felelős osztály. Adattagjai a következők:

`JTable filmTableModel` -> a filmek táblázatának kezeléséért felelős adattag

`FilmDB films` -> az adatbázisban található összes film

`JTable userTableModel` -> a felhasználók táblázatának kezeléséért felelős adattag

`UserDB users` -> az adatbázisban található összes felhasználó

`SearchUsers userSea`rch -> users nézet keresőmezője

`SearchMovies movieSearch` -> films nézet keresőmezője

`JPanel topPanel` -> top 10 filmet megjelenítő panel

`JPanel filmPanel` -> összes filmet megjelenítő panel

`JPanel userPanel` -> összes felhasználót megjelenítő panel

Az osztály fontosabb algoritmusai:

`public MainFrame() {…}`

Beállít egy címet megjelenítő panelt, alapvetően induláskor a top 10 film látszódik. Beolvassa a megfelelő algoritmussal a fájlokból a DB-ekbe az elemeket.

`private void newFilm() {…}`

FilmFrame segítségével beolvas egy új filmet, hozzáadja az adatbázishoz, ha minden rendben az inputok formátumával. Try-catch-csel nézi, hogy nem hibás-e a bevitt érték (Int helyett nem String). Throws NumberFormatException, ha helytelen a bevitel.

`private void newUser() {…}`

UserFrame segítségével beolvas egy új felhasználót, hozzáadja az adatbázishoz, ha minden rendben az inputok formátumával. Try-catch-csel nézi, hogy nem hibás-e a bevitt érték (Int helyett nem String). Throws NumberFormatException, ha helytelen a bevitel.

`private void removeFilm() {…} és private void removeUser() {…}`

A táblázat kiválasztott során meghívja az eltávolító függvényeket.

`private void editFilm() {…} és private void editUser() {…}`

Hasonló a kettő működése, a táblázat kiválasztott sorára meghívja FilmFrame és UserFrame segítségével az új beviteli mezőket. Throws NumberFormatException, ha a felhasználó rossz formátumban adja meg az adatokat (szám helyett szöveg).

`private void OrderDvd() {…}`

A filmek táblázatának kiválasztott sorára meghívja a dvd rendelő függvényeket, LoginFrame-mel ellenőrzi a helyes bejelentkezési adatok megadását, ha elfogadja, lefutnak a megfelelő függvények a filmek adatbázisán.

`private void showFilmFrame() {…}`

Láthatóvá állítja a saját panelét, a másik kettőt láthatatlanná. Hozzáadja a filmek pa neljához a keresőmezőt is. Megjelenik az összes film táblázatának panele.

`private void showUserFrame() {…}`

Láthatóvá állítja a saját panelét, a másik kettőt láthatatlanná. Hozzáadja a felhasználók p aneljához a keresőmezőt is. Megjelenik az összes felhasználó táblázatának panele.

`private void showTopFilmFrame() {…}`

Láthatóvá állítja a saját panelét, a másik kettőt láthatatlanná. Megjelenik a top 10 film táblázatának panele.

`private void menu() {…}`

A MainFrame felső sorába beleteszi a menüpontokat, azokon belül elhelyez gombokat, mindegyik más-más funkcióval. A funkciók a következők:

Users: felhasználókkal kapcsolatos műveleteknek (hozzáadás, törlés, módosítás) hívja meg a megfelelő függvényét.

Movies: filmekkel kapcsolatos műveleteknek (hozzáadás, törlés, módosítás) hívja meg a megfelelő függvényét.

Save: lehetőséget kínál elmenti mindkettő vagy csak az egyik adatbázist, annak az osztálynak a saveToFile() függvényének a meghívásával.

Exit: System.exit(0) meghívása, tetszés szerint mentéssel.

Views: show…Frame() függvény meghívása gombnyomásra.

Tesztesetek

`FilmTest.java() {…}`

Az osztály felelős a JUnit tesztjeinek lebonyolításáért.

Következő eseteket teszteli:

1. Felhasználó keresés nem hibás adatbázison hajtódik-e végre?

```
@Test
public void userSearchInitTest() {
    SearchUsers us = new SearchUsers(null);
    Assert.assertEquals("USERSEARCH INICIALIZALAS FAIL",null, us.getUsers());
}
```

2. Film keresés nem hibás adatbázison hajtódik-e végre?

```
@Test
public void filmSearchInitTest() {
    SearchMovies ms = new SearchMovies(null);
    Assert.assertEquals("MOVIESEARCH INICIALIZALAS FAIL",null, ms.getMovies());
}
```

3. Felhasználó keresés adattagjai megfelelően inicializálódnak-e?

```
@Test
public void userSearchGetSetTest() {
    UserDB udb = new UserDB();
    udb = UserDB.readFromFile(udb);
    SearchUsers us = new SearchUsers(udb.getUsers());
    us.setUsers(udb.getUsers());
    Assert.assertEquals("USERSEARCH GET/SET FAIL",udb.getUsers(), us.getUsers());
}
```

4. Film keresés adattagjai megfelelően inicializálódnak-e?

```
@Test
public void filmSearchGetSetTest() {
    FilmDB fdb = new FilmDB();
    fdb = FilmDB.readFromFile(fdb);
    SearchMovies ms = new SearchMovies(fdb.getFilms());
    ms.setMovies(fdb.getFilms());
    Assert.assertEquals("MOVIESEARCH GET/SET FAIL",fdb.getFilms(), ms.getMovies());
}
```

5. Új film valóban hozzáadódik a FilmDB-hez?

```
@Test
public void filmdbAddTest() {
    FilmDB fdb = new FilmDB();
    Film f = new Film();
    fdb.movieAdd(f.getTitle(), f.getReleaseYear(), f.getDuration(), f.getRating(), f.getDVDs(), f.getCategory(), f.getDescription());
    Assert.assertTrue("FILMDB ADD FAIL", fdb.getFilms().contains(f));
}
```

6. Új felhasználó hozzáadása után a user megtalálható UserDB-ben.

```
@Test
public void userdbAddTest() {
    UserDB udb = new UserDB();
    User u = new User();
    udb.userAdd(u.getUserName(), u.getPassword(), u.getBirthYear(), u.getNumOfMovies(), u.getMoviesSeen());
    Assert.assertTrue("USERDB ADD FAIL", udb.getUsers().contains(u));
}
```

7. Eltávolításra került film valóban nincs a FilmDB-ben.

```
@Test
public void filmdbRemoveTest() {
    FilmDB fdb = new FilmDB();
    Film f = new Film();
    fdb.movieAdd(f.getTitle(), f.getReleaseYear(), f.getDuration(), f.getRating(), f.getDVDs(), f.getCategory(), f.getDescription());
    fdb.movieRemove(f);
    Assert.assertFalse("FILMDB REMOVE FAIL", fdb.getFilms().contains(f));
}
```

8. Ha egy user-t kitörlünk, eltűnik a UserDB-ből is?

```
@Test
public void userdbRemoveTest() {
    UserDB udb = new UserDB();
    User u = new User();
    udb.userAdd(u.getUserName(), u.getPassword(), u.getBirthYear(), u.getNumOfMovies(), u.getMoviesSeen());
    udb.userRemove(u);
    Assert.assertFalse("USERDB REMOVE FAIL", udb.getUsers().contains(u));
}
```

9. FilmDB Edit valóban átírja az adattagokat.

```
@Test
public void filmdbEditTest() {
    FilmDB fdb = new FilmDB();
    Film f = new Film();
    fdb.movieAdd(f.getTitle(), f.getReleaseYear(), f.getDuration(), f.getRating(), f.getDVDs(), f.getCategory(), f.getDescription());
    fdb.movieEdit(f, "title test", 2023, 30, 5, 0, "category test", "description test");
    Film testFilm = new Film("title test", 2023, 30, 5, 0, "category test", "description test");
    Assert.assertEquals("FILMDB EDIT FAIL", testFilm.getTitle(), f.getTitle());
}
```

10. UserDB Edit után mások lesznek a user tulajdonságai.

```
@Test
public void userdbEditTest() {
    UserDB udb = new UserDB();
    User u = new User();
    udb.userAdd(u.getUserName(), u.getPassword(), u.getBirthYear(), u.getNumOfMovies(), u.getMoviesSeen());
    udb.userEdit(u, "username test", "password test", 2023, 0, "seen movies test");
    User testUser = new User("username test", "password test", 2023, 0, "seen movies test");
    Assert.assertEquals("USERDB EDIT FAIL", testUser.getUserName(), u.getUserName());
}
```

11. Ha van legalább 1 dvd-je raktáron a filmnek, megrendelhető az állapota.

```
@Test
public void filmdbOrderTest1() {
    FilmDB fdb = new FilmDB();
    Film f = new Film("title test", 2023, 30, 5, 1, "category test", "description test");
    fdb.movieAdd(f.getTitle(), f.getReleaseYear(), f.getDuration(), f.getRating(), f.getDVDs(), f.getCategory(), f.getDescription());
    Assert.assertTrue("FILMDB ORDER FAIL 1", fdb.orderValid(f));
}
```

12. Ha nincsen dvd raktáron az adott filmből, nem engedi megrendelni.

```
@Test
public void filmdbOrderTest2() {
    FilmDB fdb = new FilmDB();
    Film f = new Film("title test", 2023, 30, 5, 0, "category test", "description test");
    fdb.movieAdd(f.getTitle(), f.getReleaseYear(), f.getDuration(), f.getRating(), f.getDVDs(), f.getCategory(), f.getDescription());
    Assert.assertFalse("FILMDB ORDER FAIL 2", fdb.orderValid(f));
}
```

13. Miután megrendeltünk egy filmet, eggyel kevesebb dvd-je lesz raktáron.

```
@Test
public void filmdbOrderTest3() {
    FilmDB fdb = new FilmDB();
    Film f = new Film("title test", 2023, 30, 5, 1, "category test", "description test");
    fdb.movieAdd(f.getTitle(), f.getReleaseYear(), f.getDuration(), f.getRating(), f.getDVDs(), f.getCategory(), f.getDescription());
    int dvdNum = f.getDVDs();
    fdb.order(f);
    Assert.assertEquals("FILMDB ORDER FAIL NOT SUBTRACTING", dvdNum-1, f.getDVDs());
}
```

14. Helyes adatok megadása után érvényes a bejelentkezés

```
@Test
public void userdbLoginTest1() {
    UserDB udb = new UserDB();
    User u = new User("tesztelek", "pass", 2023, 1,"film");
    udb.userAdd(u.getUserName(), u.getPassword(), u.getBirthYear(), u.getNumOfMovies(), u.getMoviesSeen());
    Assert.assertTrue("USERDB LOGIN FAIL 1", udb.loginSuccess("tesztelek", "pass"));
}
```

15. Helytelen adatok megadásakor nem enged bejelentkezni az alkalmazás.

```
@Test
public void userdbLoginTest2() {
    UserDB udb = new UserDB();
    User u = new User("tesztelek", "pass", 2023, 1,"film");
    udb.userAdd(u.getUserName(), u.getPassword(), u.getBirthYear(), u.getNumOfMovies(), u.getMoviesSeen());
    Assert.assertFalse("USERDB LOGIN FAIL 2", udb.loginSuccess("ajjajj", "rossz lesz"));
}
```

6. FELHASZNÁLÓI DOKUMENTÁCIÓ

A P3MDB egy felhasználó által könnyedén irányítható nyilvántartó program. Elinduláskor a program kiírja a felhasználónak, hogy sikerült-e beolvasnia filmeket/felhasználókat. Ha futás közben a felhasználó bármelyik pontban is hibás adatot adna meg (pl. értékelésnél túl nagy számot, születési évnél szöveget), akkor az alkalmazás warning ablakkal értesíti erről a felhasználót és meghiúsul az éppen zajló művelet. Ha a felhasználó csak bezárja az ablakot, az becsukódik mindenféle mentés nélkül.

Majd a top 10 film listája fogadja legelőször a felhasználót. Az alkalmazás címe alatt található egy sorban a menü. A menü gombjainak lenyomásával tudja a felhasználó irányítani a programot. Az egyes menüpontok és funkcióik a következőek:

Users menüpont

1. Register new user: a menüpont kiválasztása után a program feldob egy beviteli mezőkkel rendelkező ablakot, ahol a felhasználó beírhatja az új user adatait. Ezután a felhasználónak megerősíteni vagy elvetni kell a műveletet.
2. Delete a user: először is a felhasználók táblázatából ki kell választani egy sort (ezt kéken kiemelve mutatja az alkalmazás), majd megerősíteni vagy cancel-özni kell a döntést.
3. Edit a user: a felhasználók táblázatából kiválasztott user adatai megjelennek egy újabb ablakon, ahol kedvünkre új értékeket adhatunk meg. Ismételten itt is meg kell erősíteni a műveleteket.

Movies menüpont

1. Add a movie: a menüpont kiválasztása után a program feldob egy beviteli mezőkkel rendelkező ablakot, ahol a felhasználó beírhatja az új film adatait. Ezután a felhasználónak megerősíteni vagy elvetni kell a műveletet.
2. Delete a movie: először is a filmek táblázatából ki kell választani egy sort (ezt kéken kiemelve mutatja az alkalmazás), majd megerősíteni vagy cancel-özni kell a döntést.
3. Edit a movie: a filmek táblázatából kiválasztott film adatai megjelennek egy újabb ablakon, ahol kedvünkre új értékeket adhatunk meg. Ismételten itt is meg kell erősíteni a műveleteket.
4. Order Dvd: a felugró ablakban a felhasználónak be kell írnia érvényes felhasználónév-jelszó párost, és ha elérhető a dvd a filmhez, akkor sikeres a művelet. Ha nem tudna befejeződni, warning-gal figyelmezteti a felhasználót.

Save menüpont

1. Save all: mind a felhasználók, mind a filmek táblázatán végrehajtott műveletek elmentődnek a fájlokba.
2. Save movies: csak a filmek táblázata kerül elmentésre.
3. Save users: csak a userek táblázata kerül elmentésre.

Exit menüpont

1. Exit without save: a program bezárul, semmi más művelet nem történik.
2. Save and exit: a program magától lementi a táblázatokat, majd becsukódik.

View… menüpont

1. View top 10 movies: megjeleníti a top 10 legjobb filmet
2. View movies: megjeleníti a filmek táblázatát

3. View users: megjeleníti a felhasználók táblázatát