PofaLexikon

FELADAT ISMERTETÉSE:

Nagyházinak idén egy közösségi oldalt valósítok meg. Az oldalon regisztrált felhasználók kereshetnek és követhetnek ismerősöket, valamint kommunikálhatnak egymással. Az oldalnak van egy feed- je, ahol a felhasználók az általuk követett ismerősök posztjait láthatják. Ismerősökre lehet név, email cím valamint lakcím alapján keresni. A közösségi oldal grafikus felhasználói felülettel rendelkezik, amelyet a felhasználó tud kezelni. Egy gyors regisztráció után már használhatja is a programot. A felhasználó keresése közben akár az egy településen lakó embereket is ki tudja listázni a program. Felhasználót a belépés után a feed fogadja, a menügombok segítségével navigál a programon belül.

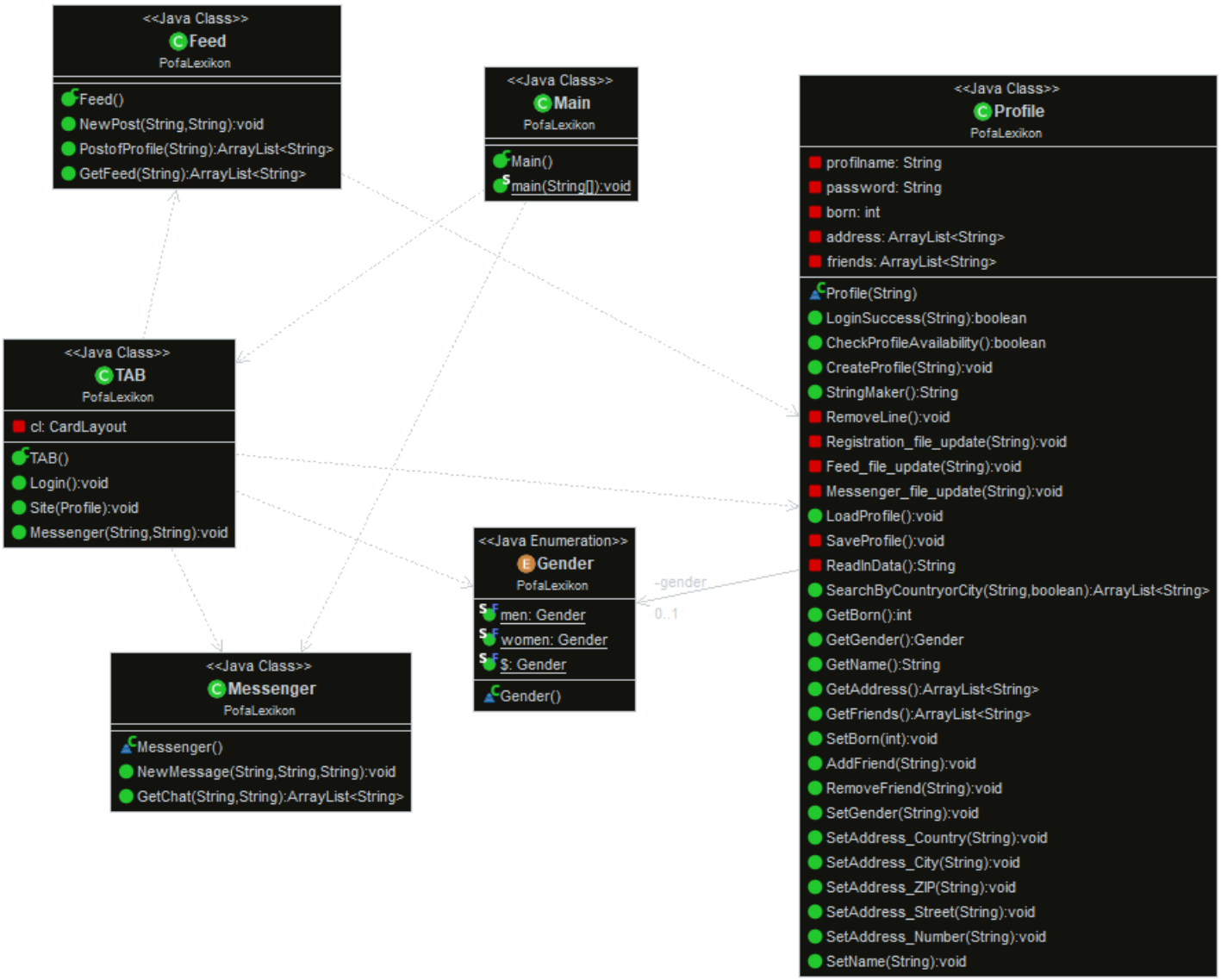
USE-CASES:

Menügombok: ▪ FEED : Itt nincsenek beállítások ▪ SEARCH : Erre kattintva tudunk felhasználóra keresni ▪ MESSENGER : Itt tudjuk megnézni az aktív beszélgetéseinket valamint újat kezdeményezni ▪ SETTINGS : Itt van lehetőség módosítani profilunkat, valamint itt tudunk kijelentkezni.

Felhasználói kézikönyv:

Miután elindítja a programot a felhasználó, lehetősége nyílik regisztrálni, esetleg ha korábban megtette ezt bejelentkezni az adataival (profilnév, jelszó). Sikeres bejelentkezést követően a fő oldalon látható a „feed" ahol a bejelentkezett felhasználó látja saját, valamint az általa követett felhasználók posztjait. Ezen az oldalon maradva, lehetőség van új posztot írni is. A program felső részében található tabokkal navigálhat a programon belül. Search fülre kattintva tud keresni felhasználókat nevük szerint, valamint egy adott országban, illetve településen lakó felhasználókat is ki lehet listázni. Ha van sikeres találat akkor itt az adott illetőt be tudja a felhasználó követni, tehát a jövőben az ő posztjait is láthatja. Az üzenetek fülön belül, kilistázza a program, kiket követünk, de bárki valós profilnak tudunk írni üzenetet. Ha begépeljük az illetőnek a nevét akivel beszélgetni szeretnénk, és rákattintunk az open chat gombra, akkor egy új ablakban megnyílik az adott felhasználóval való beszélgetésünk. Itt láthatjuk korábbi üzenetváltásainkat, valamint új üzenetet tudunk írni. Ha visszamegyünk a fő oldalra akkor van még egy fül, ez a settings, itt tudunk adatokat módosítani, mint például a születési dátumunk, nemünk, lakcímünk, valamint a nevünket is tudjuk módosítani, ha az új név nem használt már. Az megfelelő mezőket átírjuk, majd az apply gombra kattintva érvényesítjük a módosítást. A logout gombra kattintva tudunk kijelentkezni, ekkor visszajutunk a login oldalra. Bezáródik a site ablak és megnyílik a login ablak. Itt újra be tudunk jelentkezni, regisztrálni vagy kilépni a programból.

CLASS DIAGRAM:



CLASSOK:

I. PROFILE

Class leírás:
Az osztály tárolja egy felhasználó adatait, amelyeket tagfüggvényei segítségével le lehet kérdezni valamint lehet módosítani. Az osztály kezeli a felhasználó adatainak fájlba írását, valamint beolvasását. Az alább részletezem a tagfüggvényeket.

Class változók:

```
private String profilname;
private String password;
private Gender gender;
private int born;
private ArrayList<String> address = new ArrayList<String>();
private ArrayList<String> friends = new ArrayList<String>();
```

-A változó nevek egyértelműen azonosítják mit tartalmaznak az adott változók.

Class függvények:

public boolean LoginSuccess(String pass)
-ha jók a bejelentkezési adatok akkor igazzal tér vissza ellenkező esetben hamissal

public boolean CheckProfileAvailability()
-ha felhasználónév elérhető, akkor igazzal tér vissza

public void CreateProfile(String pass)
-létrehozza a profilt, tehát a Registration.txt, Profiles.txt fájl-ba beírja a felhasználó adatait

public String StringMaker()
-felhasználó adataiból létrehoz egy Stringet, majd ezt adja vissza

private void RemoveLine()
-felhasználó sorát törli a Profiles.txt – ből

private void Registration_file_update(String name)
-névmódosítás esetén frissíti a Registration.txt fájlt az új névre

private void Feed_file_update(String name)
-névmódosítás esetén frissíti a FEED.txt fájlt az új névre

private void Messenger_file_update(String name)
-névmódosítás esetén frissíti a Messenger.txt fájlt az új névre

public void LoadProfile()
-felhasználó adatait betölti a Profiles.txt – fájlból

```
private void SaveProfile()
```
-felhasználó adatait menti a Profiles.txt – fájlba

```
private String ReadInData()
```
-visszaadja a felhasználó adatait egy Stringben a Profiles.txt ből
```
public ArrayList<String> SearchByCountryorCity(String countryOrCity, boolean country)
```
-Azon felhasználó neveit adja vissza akik az adott országban avagy városban élnek.

```
public int GetBorn(){return born;}
public Gender GetGender(){return gender;}
public String GetName(){return profilname;}
public ArrayList<String> GetAddress(){ return address; }
public ArrayList<String> GetFriends() throws IOException { return friends; }
```
-Getterek

```
public void SetBorn(int n) throws IOException {RemoveLine(); born=n; SaveProfile();}
public void AddFriend(String friend) throws IOException {RemoveLine(); friends.add(friend); SaveProfile();}
public void RemoveFriend(String friend) throws IOException {RemoveLine(); friends.remove(friend); SavePro-
file();}
public void SetGender(String g) throws IOException {RemoveLine(); gender = Gender.valueOf(g);
System.out.println(gender); SaveProfile();}
public void SetAddress_Country(String s) throws IOException {RemoveLine(); address.set(0,s); SaveProfile();}
public void SetAddress_City(String s) throws IOException {RemoveLine(); address.set(1,s); SaveProfile();}
public void SetAddress_ZIP(String s) throws IOException {RemoveLine(); address.set(2,s);SaveProfile();}
public void SetAddress_Street(String s) throws IOException {RemoveLine(); address.set(3,s);SaveProfile();}
public void SetAddress_Number(String num) throws IOException {RemoveLine(); address.set(4,num);SavePro-
file();}
public void SetName(String s)
```
-Setterek, ezek gondoskodnak, nem csak az adott változók megváltozásáról, de a fájlok frissítéséről is.

II. Messenger

Class leírás:
Az üzenetekért felelős osztály. Messenger.txt fileba írja az üzeneteket

Class változók:
-

Class függvények:

```
public void NewMessage(String from, String to, String message)
```
-Messenger.txt fájlba írja az üzenetet, abban a formában : from \t to \t message \n

```
public ArrayList<String> GetChat(String Person1, String Person2)
```
-2 ember közötti üzenetváltásokat adja vissza a Messenger.txt-ből kiolvasva. a Person2 által küldött üzenet elé be van szúrva egy „$” így lehet megkülönböztetni ki küldte kinek az üzenetet.

II. Feed

Class leírás:
Az feed-ért felelős osztály. Feed.txt fileba írja az üzeneteket

Class változók:
-
Class függvények:
public void NewPost(String person, String post)
-Egy posztot ír a FEED.txt fileba, ilyen formátumban: person \t post
public ArrayList<String> PostofProfile(String person)
-Egy adott felhasználó által írt posztokat adja vissza

public ArrayList<String> GetFeed(String Person1)
-Egy adott felhasználó Feed-jét adja vissza, tehát saját, és azon emberek posztjait akiket követ az illető.

III. TAB

Class leírás:
Az üzenetekért felelős osztály. Messenger.txt fileba írja az üzeneteket

Class változók:
-

Class függvények:
public void Login()
-A bejelentkezés illetve regisztrációért felelő függvény, sikeres regisztráció esetén meghívja a Site függvényt.

public void Site(Profile profile)
-A program felhasználói felülete, sikeres bejelentkezés esetén ez a függvény hívódik meg.

public void Messenger(String me, String friend)
-A chat grafikus megjelenítéséért felelős függvény. Chat a bejelentkezett profil és friend között. A Site() függvény hívhatja meg.

IV. Main

Class leírás:
Main osztály, ez a program belépési pontja, itt van létrehozva a TAB osztály

Class változók:
-

Class függvények:
public static void main(String[] args)