Документация на проект: Социална мрежа

- ## Резюме на проекта:

Проектът представлява социална мрежа, в която може да се добавят нови потребители, да се трият съществуващи потребители, да се създават приятелства между потребители, да се премахват приятелства, да се забрани приятелство между двама потребителя, да се намери даден потребител, да се препоръчват на даден потребител нови потребители за приятели.

- ## Подход на решение:

- **Архитектура на проекта**

Проекта е разделен на Input, Parser, резултат от Parser, Output, команди, Data Storage, Engine

Цикъл на програмата: Старт на програмата->Зареждане на базата данни(файл)->Вход->Разчитане на Входа->Ако входът е бил невалиден се изписва съответна грешка->Ако входът е валиден се изпълнява съответната команда и се изписва съобщение след изпълнението на команда->Вход

- **Разделение на стъпки**

1. Имплементиране на Input/Output, Parser, резултат от Parser
2. Имплементиране на командите
3. Имплементиране на DataStorage
4. Имплементиране на Engine
5. Тестване на workflow на програмата

- ## Класове и тяхното предназначе

- **Input**

Input е класът, който чете входа, въведен от ползвача на програмата.

- **Parser**

Parser-ът се грижи да валидира входа и да създаде обект от ParseCommandResult класа, в зависимост от какъв е бил входа.

- **ParseCommandResult**

ParseCommandResult е клас с член данни команда и съобщение на грешка и метод, който проверява дали обектът е грешка(ако не е значи е команда)

- **Output**

Output класа се грижи за това да принтира на конзолата правилния изход(ако резултат от Pareser-а е команда изписва съобщението от изпълнението на командата, а ако резултатът от Parser-а е грешка, то изписва нейното съобщение).

- **Command**

Command е абстрактен клас с виртуален(абстрактен) метод execute, който

се изпълнява по предефинирания за съответната имплементация на командата.

- **CreateCommand**

Имплементация на Command, която създава потребител по подадено име и години и го добавя в „базата данни“ (map-а от user-и(потребители)).

- **LinkCommand**

Имплементация на command, която по подадени две имена на съществуващи потребители и тип на приятелство ги прави приятели, ако все още не са.

- **DelinkCommand**

Имплементация на command, която по подадени две имена на съществуващи потребители, премахва тяхното приятелство, ако има такова.

- **DeleteCommand**

Имплементация на command, която по подадено име на съществуващ потребител го трие от базата данни.

- **FindCommand**

Имплементация на command, която по подадено име изписва информация за потребителя, ако такъв съществува.

- **BanCommand**

Имплементация на command, която по подадени две имена на два потребителя добавя втория потребител в set-а от потребители с които не иска да има нищо общо първия потребител.

- **RecommendCommand**

Имплементация на command, която по подадено име на съществуващ потребител му препоръчва нови потребители за приятели

- **User**

User е класът, който пресдставлява потребител с член данни име, години, set от приятелтва, set от потребители с които не иска да има нищо общо. Той има методи за добавяне на приятелство в set-а му от приятелства, добавяне на потребител в set-а от потребители с които не иска да има нищо общо и гетъри.

- **Friendship**

Friendship е класът, който репрезентира приятелство, съдържа име и тип на приятелство, предефиниран оператор < и гетъри

- **DataStorage**

DataStorage е класът който репрезетира базата от данни за проекта. Той има, като член данни map с ключ име на потребител и стойност самия потребител обект от тип fstream, който ще служи за четене и писане във файл и методи за композиране на потребител(създаване на string от информацията за потребител който string ще се запише във файлът), парсване на потребител(създаване на потребител от string с информацията на потребителя), метод за писане във файл и метод за зареждане(прочитане) на целия файл и гетър за map-а.

- **Acquaintance**

Acquaintance е клас който ще създава обекти , с член данни коефицент и име, които обекти представляват познанство на потребител и пазят името на даден потребител и коефицента му на познанство. Този клас спомага за намирането на приятели за препоръчване на потребител, като всеки обект от класа всъщност пази коефицента(колкото-по висок толкова по възможно да се познава с даден потребител, на който искаме да препоръчаме нови приятели). Има предефиниран оператор< и гетъри.

- **Engine**

Engine е класът, който има като член данни обекти от класовете Input, Output, Parser, ParseCommandResult, DataStorage. Той представлява взаимодействието между всички класове за правилен workflow на програмата. Има един единствен метод start, който вкарва програмата в безкраен цикъл и изпълнява workflow-а описан в първата страница от документацията.

- ## Начин на ползване на програмата

Команди:

create име_на_потребител години

delete име_на_потребител

find име_на_потребител

link име_на_потребител име_на_потребител тип_на_приятелство

delink име_на_потребител име_на_потребител

ban име_на_потребител име_на_потребител

recommend име_на_потребител

Където име_на_потребител е едно единствено име, което започва с главна латинска буква, а всички останали са малки латиснки и не се допускат други знаци и е съставено от поне две букви. Години са число между 12 и 120. Тип_на_приятелство е bestie, relative или normal.

Програмата работи като изпълнява съответната команда всичко е въведено правилно или изписва грешка за съответно несъответсвие във входа.

Правила:

Никой потребител не може да е приятел със себе си.

Никой потребител не може да стане пряител с друг потребител,който е в списъкът му от забранени потребители.

Особености на имплементации

Recommend командата намира 30-те най-потенциални приятели за препоръка, ако не успее да намери 30, ги допълва до 30 с потребителите с най-много приятели. Намирането на потенциални приятели за препоръка става по следния начин:обхождат се приятелите на приятелите на потребителя , на който искаме да му препоръчаме нови приятели и се оценяват с коефицент съответно с 3 за bestie,2 за relative и 1 за нормал в зависимост от типа на приятелството им с приятеля на нашия потребител. Като не се разглеждат потребители, които вече са приятели с потребителя или са в списъка му със забранени потребители. Намират се първите 30 и се връщат сортирани в низходящ ред относно коефицента на познанство. Ако потребителя, на който искаме да препоръчаме нови приятели няма никакви приятели му препоръчваме хората с най-много приятели подредени в низходящ ред по брой приятели, като не му препоръчваме повече от 30 потребителя.

При всяко изпълнение на команда пишем цялото съдържание на map-а от потребитеи във файла „user-data.txt”.

При първо стартиране на програмата зареждаме информацията за потребителите от файла „user-data.txt”.

Предефинирания оператор< в класовете Acquaintance и Friedship ни позволяват да ги пазим в set сортирани в зависимост от предефиницията.

Почти всичко се пази сортирано лексикографски(приятелите на даден потребител,самите потребители), което ни дава възможност да намираме потребители или приятели със сложност O(logn).

Средна сложност на почти всикчки методи е O(logn).

Идеи за бъдещи подобрения

- Изнасяне на абстракцията на Input и Output по високо ниво, т.е. да работят не само с текстов файл и конзолата, а всякакъв поток.
- Използване на enum за типовете приятелства
- Добавяне на нови команди (премахване на ban)

- Подобряване на бързодействието където е възможно
- Използване на повече структури с цел бързодействие