Обектно-ориентирано програмиране, летен семестър 2017/2018

Домашно 1: Структури и файлове - поправка

Честито! Вие сте посетител номер 1000000 на тази страница! На вас се пада уникалната възможност да се включите в създаването на най-успешния проект, включващ не-block-chain технологията! От вас се иска да направите програма, която ще съхранява и обработва данни за потребители на новата не-анонима, не-крипто-валута FMICoin, само за студенти!

От вашата програма ще се изисква да може да създава портфейли при подаване на команда `add-wallet fiatMoney name`, където **fiatMoney** е началният баланс (инвестиция) на портфейла в истински пари (левове), а **name** е името на притежателя на портфейла, което не надхвърля 255 символа. При създаването на портфейл, трябва да му генерирате уникално **id** (което е цяло *положително* число, по малко от 2^32), за идентификация на портфейла! Данните за създадените портфейли трябва да съхранявате в бинарен файл с името *wallets.dat*, и при стартиране на вашата програма ако съществува такъв файл, трябва да зареждате от него вече запазената информация.

Трябва да поддържате и списък с извършени транзакции между различните портфейли, запазени във вашата програма, като запазвате информацията за тези транзакции в бинарен файл с име *transactions.dat*. Данните за една транзакция са **senderId** - уникалният номер на изпращача, **receiverId** - уникалният номер на получателя, **fmiCoins** - количеството трансферирани FMICoin-и с тип реално число и **time** - цяло число с големина 8 байта, което е броят секунди изминали от 01.01.1970г 00:00ч. За да стимулираме развитието на иновативната валута, при регистрацията на всеки нов потребител ще захранваме неговата сметка с начално количество FMICoin-и, равни на стойността на парите, които той инвестира (количеството **fiatMoney,** подадено на програмата при създаване на портфейл) разделени на 375 (число, генерирано от честен генератор на случайни числа, и същевременно определящо константния курс на FMICoin монетите, спрямо истинските пари). Тоест, при създаване на портфейл, трябва и да създадем една транзакция, която има получател новосъздадения портфейл и количеството изчислени монети, а за изпращач ще записваме номера на системния портфейл - 4294967295 (инцидентно, този номер не е валиден, за който и да е създаден портфейл).

Няма как да направим успешна не-крипто-валута без начин за търгуване. За това вашата програма трябва да може да съхранява информация за нареждания за покупка и продажба на монети, направени от потребителя. Информацията за едно нареждане съдържа три полета: **type** - тип на нареждането, а именно **SELL** или **BUY**, **walletId** номер на портфейла, който създава нареждането и **fmiCoins** - количество на монетите, които са предложени за продажба или покупка. Трябва да можете да създавате нареждания с команда `make-order type fmiCoins walletId`, където **type** е едно от SELL и BUY, **fmiCoins** е реално число, представляващо количество монети и **walletId** е номерът на наредителя. Информацията за всички създадени нареждания трябва да съхранявате в бинарен файл с име *orders.dat*.

Имате следните условия при създаване на нареждане за *продажба* на монети:

- Съществува портфейл за подадения номер;
- Наредеителят (**walletId**) трябва да има поне толкова монети колкото иска да продаде;

    - За да разберете какво количество монети притежава един портфейл то трябва да разгледате извършените транзакции, които афектират него (като имате предвид че началното му състояние от монети идва от транзакцията от системният портфейл с номер 4294967295).
- За да изпълни текущата заявка за продажба, програмата трябва да провери дали има нареждания за купуване на монети и да удовлетвори някои или всички от тях в реда в който са били направени. Ако се случи така, че има недостатъчно нареждания за покупка за да се удовлетвори текущото нареждане, то остатъка от текущото нареждане се записва във файл с гореспоменатия формат. Използваните нареждания за покупка се премахват от списъка с нареждания, и за всяко едно от тях се създава транзакция във файла за транзакции *transactions.dat* със съответните изпращач и получател и монети.
- За всяка изпълнена транзакция трябва да отразите и промяната на **fiatMoney** в портфейлите на участниците в транзакцията, като използвате гореспоменатият курс за монети спрямо **fiatMonet**.

Условията за създаване на нареждане за *покупка* на монети са следните:

- Съществува портфейл за подадения номер
- Наредителят трябва да има достатъчно истински пари за да закупи желания брой монети, изчислен при гореспоменатия курс. За да изпълни текущата заявка, програмата трябва да провери дали има нареждания за продажба на монети и да удовлетвори някои или всички от тях, в реда в който са били направени. Изпълняват се заявки за продажба на монети, докато се изпълни цялата текуща заявка или докато няма повече подходящи заявки. Ако се окаже че няма достатъчно заявки за да може текущата да се удовлетвори напълно, то тогава записваме текущата заявка във файла за заявки *orders.dat*, но със съответно приспаднато количество монети. За всяка от извършените транзакции в този процес трябва да направите запис за транзакция със съответните получател, изпращач и монети. Също всяка изпълнена заявка се премахва от списъка заявки.
- За всяка изпълнена транзакция трябва да отразите и промяната на **fiatMoney** в портфейлите на участниците в транзакцията, като използвате гореспоменатият курс за монети спрямо **fiatMonet**.

При създаване на нареждане трябва да генерирате текстов файл с информация за всички изпълнени заявки, които са били предизвикани от създаването на това нареждане. В този текстов файл трябва да има по един ред за всяка транзакция, в които се включват имената на получателя, изпращача и количеството трансферирани монети. Накрая на файла трябва да се запише броят извършени транзакции и себестойността на транзакциите в истински пари. Името на този файл трябва да съдържа идентификационния номер на портфейла, който задава нареждането и времето в което се е случило нареждането.

Програмата ви трябва да поддържа и следните две команди:

- `quit` спиране на програмата и записване на всички данни в съответните файлове.
- `wallet-info walletId` - дава информация за портфейл с въведен номер, като показва името и количеството пари в него, а също така количеството монети, които притежава. Количеството монети, които притежава този портфейл, може да бъде изчислено от транзакциите, в които участва портфейлът.
- `attract-investors` - дава информация за 10-те най богати (спрямо количеството монети) потребителя в системата. За всеки от тях показва количеството монети, броя извършени заявки, времето на първата и последната заявка извършени от тях.

Забележки и пояснения

- При пускане на програмата бинарните файлове, споменати по-горе, може да съществуват от предишно изпълнение. При това положение, при изпълнението на всяка команда, трябва да се съобразявате с данните вече записани във файловете.
- Няма ограничение кои данни може да съхранявате в паметта на вашата програма или да държите само в файловете, но имайте предвид, че този проект ще бъде много успешен, което означава, че ще има множество записи от всеки тип (портфейли, заявки и транзакции). Както **всеки** програмист, ценящ труда си, трябва да се погрижите всяка команда да се изпълнява максимално бързо и ефикасно.

Отделът за развойна дейност прилага и следния откъс от *c++* код, който да ви помогне в започване на задачата:

```
struct Wallet {
    char owner[256];
    unsigned id;
    double fiatMoney;
};

struct Transaction {
    long long time;
    unsigned senderId;
    unsigned receiverId;
    double fmiCoins;
};

struct Order {
    enum Type { SELL, BUY } type;
    unsigned walletId;
    double fmiCoins;
};
```

Пример 1:

Приемаме че има създадени 3 портфейла с ид-та (100, 101 и 102), файлът с нарежданията е празен и изискванията за следните команди (налични пари и монети са изпълнени):

```
        make-order BUY 3 100 // създаваме нареждане с тип BUY за 3 моенти от портфейл 100
        make-order BUY 5 101 // създаваме нареждане с тип BUY за 5 моенти от портфейл 101
        make-order SELL 10 102 // това нареждане 'консумира' вече съществуващите 2 но остават още 2 за пр
одаване
```

След изпълнението на командите в файлът с нарежданията (orders.dat) има едно нареждане от тип SELL, на стойност 2 монети от walletId 102. А във файлът с транзакциите са добавени 2 транзакции:

```
Transaction { time(), 102, 100, 3 }
Transaction { time(), 102, 101, 5 }
```

Пример 2:

Приемаме че има създадени 2 портфейла с ид-та (100 и 101), файлът с нарежданията е празен и изискванията за следните команди (налични пари и монети са изпълнени):

```
        make-order SELL 10 100 // създаваме нареждане с тип SELL за 10 моенти от портфейл 100
        make-order BUY 4 101 // това нареждане 'консумира' само част от съществуващото
```

След изпълнението на командите в файлът с нарежданията (orders.dat) има едно нареждане от тип SELL, на стойност 4 монети от walletId 100. А във файлът с транзакциите е добавена 1 транзакция:

```
Transaction { time(), 100, 101, 4 }
```

Състояние на заданието

Състояние на заданието	Няма опити
Състояние на оценяването	Неоценена
Краен срок	сряда, 15 август 2018, 23:55
Оставащо време	Предаването на работата закъснява с 47 дни 17 часа след срока
Последна промяна	-
Коментари към заданието	▶ Коментари (0)

◀ Предаване на проекти - поправка

Отиди на ...

Домашно 2: Класове и обекти - поправка ▶

НАВИГАЦИЯ

Начална страница
Моето табло
Страници от сайта
Моите курсове
Проекти
Учебни материали
Архив
Бакалаври, летен семестър 2017/2018
Избираеми дисциплини
И
Обектно-ориентирано програмиране, летен семестър 2...
Участници
Значки
Компетенции
Оценки
Главна
Поправителна сесия
Предаване на проекти - поправка
Домашно 1: Структури и файлове - поправка
Домашно 2: Класове и обекти - поправка
Домашно 3: Наследяване и полиморфизъм - поправка
Домашно 4: Шаблони и предефиниране на оператори - ...
Домашно 5 - задължително за поравката
Запазване на оценка от редовния изпит
Подготовка на мястото за работа за практическия изпит
Практически изпит 25.08.2018 г.
Теоретичен изпит 25.08.2018 г.
Крайна оценка - поправителна сесия

Полезни ресурси

От лекции

Проекти

Числени методи, летен семестър 2017/2018

Логическо програмиране, летен семестър 2017/2018

- Още...

Курсове

НАСТРОЙКИ

Администриране на курс

Вие сте влезли в системата като Тея Андреева (Изход)
С402619
Get the mobile app