Split(NotSo)Wise 💸

Напишете клиент-сървър приложение с функционалност, наподобяваща тази на Splitwise.

Splitwise цели улесняване на поделянето на сметки между приятели и съквартиранти и намаляване на споровете от тип "само аз купувам бира в това общежитие".

Условие

Създайте клиентско конзолно приложение, което приема потребителски команди, изпраща ги за обработка на сървъра, приема отговора му и го предоставя на потребителя в четим формат.

Note: Командите и output-а в условието са примерни, свободни сте да ги преименувате и форматирате. Единственото условие е те да бъдат интуитивни. За улеснение на потребителя, може да имплементирате команда help.

Функционални изисквания

- Регистрация на потребител с username и password; Регистрираните потребители се пазят във файл при сървъра - той служи като база от данни. При спиране и повторно пускане, сървърът може да зареди в паметта си вече регистрираните потребители.
- Login;
- Регистриран потребител може да:
  - добавя вече регистрирани потребители във Friend List на база техния username. Например:

    ```
    $ add-friend <username>
    ```

  - създава група, състояща се от няколко, вече регистрирани, потребители:

    ```
    $ create-group <group_name> <username> <username> ... <username>
    ```

    Групите се създават от един потребител, като всяка група включва трима или повече потребители. Можете да си представяте, че "приятелските" отношения са група от двама човека.

  - добавя сума, платена от него, в задълженията на:
    - друг потребител от friend list-а му:

      ```
      $ split <amount> <username> <reason_for_payment>
      ```

    - група, в която участва:

      ```
      $ split-group <amount> <group_name> <reason_for_payment>
      ```

  - получава своя статус - сумите, които той дължи на приятелите си и в групите си и сумите, които дължат на него. Например:

    ```
    $ get-status
    Friends:
    * Pavel Petrov (pavel97): Owes you 10 LV

    Groups
    * 8thDecember
    ```

```
- Pavel Petrov (pavel97): Owes you 25 LV
- Hristo Hristov (ico_h): Owes you 25 LV
- Harry Gerogiev (harryharry): You owe 5 LV
```

Може да визуализирате всички групи и приятели или само тези, при които има "неуредени сметки".

- Нововъведена сума се дели поравно между всички участници в групата или наполовина, ако се дели с потребител от Friend List-а.
- Когато един потребител А дължи пари на друг потребител В, задължението може да бъде "погасено" (с подходяща команда) само от потребител В.

```
$ payed <amount> <username>
```

Например:

```
$ get-status
Friends:
* Pavel Petrov (pavel97): Owes you 10 LV
* Hristo Hristov (ico_h): You owe 5 LV

$ payed 5 pavel97
Pavel Petrov (pavel97) payed you 5 LV.
Current status: Owes you 5 LV

$ get-status
Friends:
* Pavel Petrov (pavel97): Owes you 5 LV
* Hristo Hristov (ico_h): You owe 5 LV
```

- Когато един потребител А дължи на потребител В сума (например 5$), но преди да ги върне на В добави друга сума, която той е платил (например 5$), тогава дължимите суми и на двамата се преизчисляват (дължимата сума на А ще стане 2.50$, В все още не дължи нищо, но има да взима 2.50$).

```
$ get-status
Friends:
* Pavel Petrov (pavel97): Owes you 10 LV
* Hristo Hristov (ico_h): You owe 5 LV

$ split 5 ico_h limes and oranges
Splitted 5 LV between you and Hristo Hristov.
Current status: You owe 2.50 LV

$ get-status
Friends:
* Pavel Petrov (pavel97): Owes you 5 LV
* Hristo Hristov (ico_h): You owe 2.50 LV
```

- При всяко влизане на потребителя в системата, той получава известия, ако негови приятели са добавяли суми или "погасявали" дългове. Например:

  ```
  $ login alex alexslongpassword
  Successful login!
  No notifications to show.
  ```

  или

  ```
  $ login alex alexslongpassword
  Successful login!
  *** Notifications ***
  Friends:
  Misho approved your payment 10 LV [Mixtape beers].

  Groups:
  * Roomates:
  You owe Gery 20 LV [Tanya Bday Present].

  * Family:
  You owe Alex 150 LV [Surprise trip for mom and dad]
  ```

- Потребителят може да види история на плащанията, извършени от него. Тази история се пази във файл на сървъра.

- (***Бонус***) Сървърът предоставя възможност за currency conversion. Валутата по подразбиране е български лева, като потребителят може да я смени по всяко време на изпълнение на програмата, чрез подходяща команда (например `switch-currency EUR`). Всички суми, които потребителят дължи и дължат на него към този момент, преминават в избраната валута.

  Чрез HTTP заявка до някое публично API (например https://exchangeratesapi.io/) вземете текущите стойности на валутите и обработете response-а.

Нефункционални изисквания

- Сървърът може да обслужва множество потребители паралелно.

Съобщения за грешки

При неправилно използване на програмата, на потребителя да се извеждат подходящи съобщения за грешка.

При възникване програмна грешка, на потребителя да се извежда само уместна за него информация. Техническа информация за самата грешка и stackTraces да се записват във файл на файловата система - няма определен формат за записване на грешката.

Например, нерелевантно е при логин на потребител и възникнал проблем с мрежовата комуникация, да се изписва грешка от вида на "IO exception occurred: connection reset", по-подходящо би било "Unable to connect to the server. Try again later or contact administrator by providing the logs in <path_to_logs_file>".

При възникване на програмна грешка от страна на сървъра, подходящо съобщение се изписва на конзолата и във файл, като освен това, във файла се записва допълнителна информация (например при заявка на кой потребител е възникнала грешката, ако въобще е обвързана с потребителско взаимодействие) и stacktraces.

Уточнения

- Както може би се досещате, това не е система за банкови транзакции и плащането на съответните суми не ни интересува как се случва.

  Нека си представим, че Ана и Ива са съквартирантки. Ана плаща общежитията и на двете и записва това в приложението. После Ива ѝ дава парите, които дължи, а Ана влиза в приложението и записва, че Ива ѝ се е издължила. Ива не може сама да каже "Аз платих моята част".

- Валидацията на потребителския вход е задължителна, т.е. покрийте всички сценарии, за които се сетите с невалиден вход - било то null, грешно форматиране, невалиден тип на данните и т.н.

- Командите и output-а от тях са примерни. Ако не ви допадат или не ги разбирате, сте свободни да използвате други. Единствената им цел тук е да помогнат за разбирането на условието.

- Всякакви допълнителни функционалности, за които се сетите, са добре дошли.

Submission

Качете `.zip` архив на познатите папки `src`, `test` и `resources` (опционално, ако имате допълнителни файлове, които не са .java) в sapera.org. Там няма да има автоматизирани тестове. Проектът ви трябва да е качен в грейдъра не по-късно от 18:00 в деня преди датата на защитата.

Успех!