7.10.18

Readme - Stocks portfolios project

Welcome to Stocks portfolios project. This project allows you to create and manage a stock portfolio for your customers.
Note: you can create stocks portfolios in parallel.

1. **Server API:**
    a. **Register** - register new customer which consists of unique id and stocks (names and amounts).
    b. **Update** - updating exist customer portfolio by id. It's updating specific stocks.
    c. **Replace** - updating exist Customer portfolio by id. It's setting a new portfolio.
    d. **Value** - calculating the portfolio value of the customer.
2. you need to have **Maven** installed
3. When I'm saying: "**resources library**", I mean to the library in the relative path: "src/main/resources/".
4. - Prepare input files:
    - [MUST] File for building the **initial stocks database**, which called "stocks_states.txt", in the following format:
        - All the stocks names separated by single space, and after the last name, put space, $, space, and initial stock value (will be to all the stocks).
        - In the other lines, write stock state changes – name, space and amount.
        - You can see the *"stocks_states.txt"* file for example, in resources library.
    - [MUST] For **registering** new stocks portfolio (=new customer):
      - File in the format: line contain (existing) stock name, space, (integer) amount of the stock.
      You can see two examples at the files *"client1_registerPortfolio.txt"* and *"client2_registerPortfolio.txt"* for two register requests (in resources library).
      Prepare such a file per register request.
      - File which contains the names of the above register request input files – they will register in parallel. You can see the *"registerFilesNames.txt"* file, as an example (in resources library).
    - For **updating** existing customer - **update** existing stocks:
      - File in the format: line contain (existing) stock name, space, (integer) amount of the stock.
      You can see two examples at the files:
      "client1_newPortfolio.txt" and "client2_updatePortfolio.txt",
      for two update-updating requests (in resources library).
    - For **replacing** portfolio of an existing customer- set a new portfolio:

prepare file in above format.
You can see two examples at the files:
"client1_newPortfolio.txt" and "client2_newPortfolio.txt", for
two replace requests (in resources library).

- For both requests – update and replace, you need to prepare:
  - File which contains the names of the above update request input files. You can see the *"newPortfolioFilesNames.txt"* file for replace requests, and "updatePortfolioFilesNames.txt" for update requests, as examples (in resources library).
  - File with indexes of ids on "ids.txt" file (in resources library). For example, if you want to update the portfolio with the first id, write 0, for second write 1 etc. You can see the "idsForUpdate.txt" file as an example (in resources library).
- Similarly, for calculate the portfolio **value** of customer, prepare file with indexes of the portfolios ids which you want to calculate their value.
  You can see the "idsForValue.txt" file as an example (in resources library).

5. I used call to the resources library path (see above) **basePath**. I defined the default basePath to be "src/main/resources" at the pom.xml file. If you want to change the location, change the basePath accordingly when you run the program (by maven, recommended, see below).
   You need to define the basePath (i.e "src/main/resources" in default) as parameter to the WebServer and to the CLIRunner Classes.
6. Running the project:
   Open the terminal / command-line from the folder of the project, and run the project with **Maven** by running the following commands:
   ```
   mvn clean install
   ```
   - After the above comment the **server is up** (the server class is called "WebServer").
   - Now, whenever you want to run **client**, you need to run the **runner**, which let you run multiple clients (the clients number is according to the input files which you prepared, see above), by the command:
   ```
   mvn exec:java@second-execution
   ```

   (If you want to change the default resources library, you need to run the following commands:
   ```
   mvn clean install "-DbasePath=Another path to the resources folder"
   mvn exec:java@second-execution "-DbasePath=Another path to the resources folder")
   ```
   Now follow the instructions for inputs which will show on the screen.

7. **Important notes:**
   1. It is important to write .txt at the end of the file names when you enter the inputs for the programs.
   2. indexes start from 0.
   3. The files I prepared for example will work without print errors, if we first run
   4. run operation 3 (replace) and then operation 2, we will not get print errors. But of course, you can build matching examples so that you can run operation 2 and then operation 3 without print errors. Anyway, the program not crashing.
   5. I prepare files for 2 requests (threads) at the same time. You can prepare any quantity you want (line for each request) and it will work. Limit: 10,000.
   6. Recommendation: Do not prepare the input text files through Windows Notepad, it creates problems. Please use a different text generator such as notepad++.
8. <u>Examples for runs:</u>

   - **Register** request (two inputs):

   1
   registerFilesNames.txt

   - **Update** request (three inputs):

   2
   idsForUpdate.txt
   updatePortfolioFilesNames.txt

   - **Replace** request (three inputs):

   3
   idsForReplace.txt
   newPortfolioFilesNames.txt

   - **Value** request (two inputs):

   4
   idsForValue.txt