Nutrition Pantry

Documentation and Manual
CSCD 212, Spring 2022

Eric Pratt and Anna Mikhailenko

Contents:

Getting Started

Executing

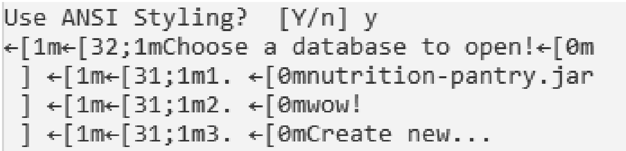
To execute the app, navigate to the directory where `nutrition-pantry.jar` is located using the terminal app of your choice. Assuming `java` is within your terminal's PATH, execute

```
java -jar nutrition-panry.jar
```

To run the command line application. The app will not perform any functions if run via a double-click on some systems that hide the terminal by default, however this method of execution will always display the interface.

Startup

To begin, the user is first asked whether they'd like to enable ANSI styling throughout the program interface. This provides color and text effects that make the program more pleasant and easy-to-use, however, this feature is not supported on some terminals - notably, Windows' cmd.exe and powershell.exe - and will instead produce garbled results.

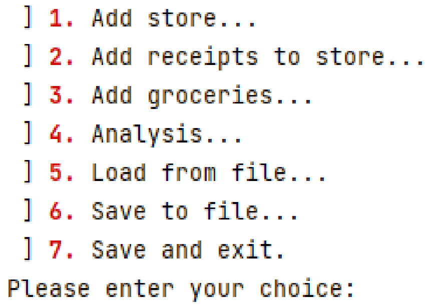


```
Use ANSI Styling?  [Y/n] y
←[1m←[32;1mChoose a database to open!←[0m
 ] ←[1m←[31;1m1. ←[0mnutrition-pantry.jar
 ] ←[1m←[31;1m2. ←[0mwow!
 ] ←[1m←[31;1m3. ←[0mCreate new...
```

Example of unsupported ANSI in cmd.exe - please restart and choose disable!

The user is then prompted to choose a database to open, from the working directory. See *Main Menu.5* for more information on loading databases.

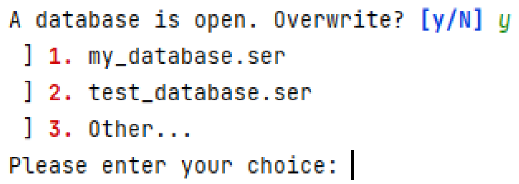
Main Menu



```
  ] 1. Add store...
  ] 2. Add receipts to store...
  ] 3. Add groceries...
  ] 4. Analysis...
  ] 5. Load from file...
  ] 6. Save to file...
  ] 7. Save and exit.
 Please enter your choice:
```

From this menu, all the functions of the program can be accessed. The first three options add data into the database, the fourth option enters the analysis mode, and the final three options load and save the database to disk.

1. **Add a store by name to the database**. It is necessary to perform this step before any receipts may be added.
2. **Add Receipts to the database.** The user must choose the store the receipt is added to and is then prompted to enter a date and a set of groceries. The user must create the groceries before adding them into the receipts. See *Receipts* below.
3. **Add groceries to the database**. The user has a choice to add the groceries either by name and UPC number to automatically download the nutrition information, or the user can manually add the nutrition information of the grocery. See *Groceries* below.
4. **Perform Analysis.** Enters Analysis mode, which allows the user to analyze the nutrition and spending habits. This is the main method by which the database can be queried. See *Analysis* below.
5. **Load database from a file on disk**. If a database is currently open, it will prompt if you wish to discard it before loading. Then, a prompt with the databases in the working directory will be shown, or a custom filepath may be entered by choosing the "Other…" option.

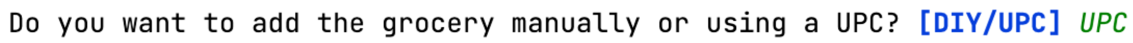


```
A database is open. Overwrite? [y/N] y
 ] 1. my_database.ser
 ] 2. test_database.ser
 ] 3. Other...
Please enter your choice: |
```

6. **Save the working database to disk.** Writes the working database to a file specified by the user. Provides a list of all databases in the working directory to choose from, or provides an "Other…" option to allow saving a new file to a custom filepath. Will prompt a confirmation message before overwriting an existing file.
7. **Save the working database to disk and exit.** Performs the same operation as **6**, but the program exits upon a successful write to disk.

Groceries

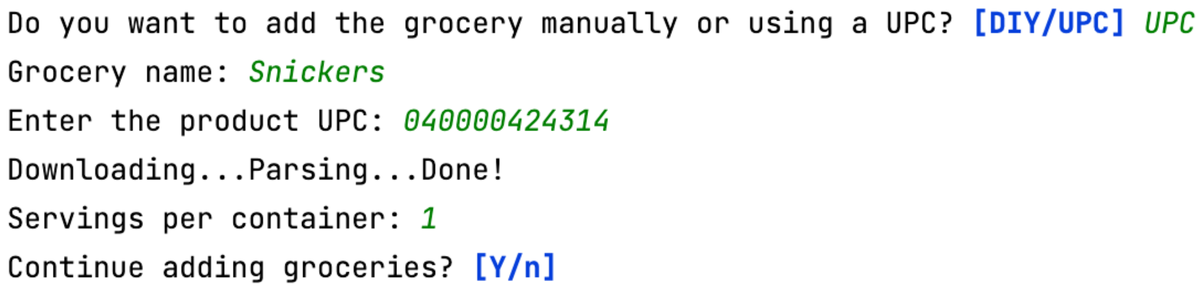
When the user chooses the add groceries mode the user has a choice between using a UPC to add the groceries or manually type in the grocery information. Following this, they are prompted to either continue adding groceries, or exit this mode.



```
Do you want to add the grocery manually or using a UPC? [DIY/UPC] UPC
```

UPC/EAN

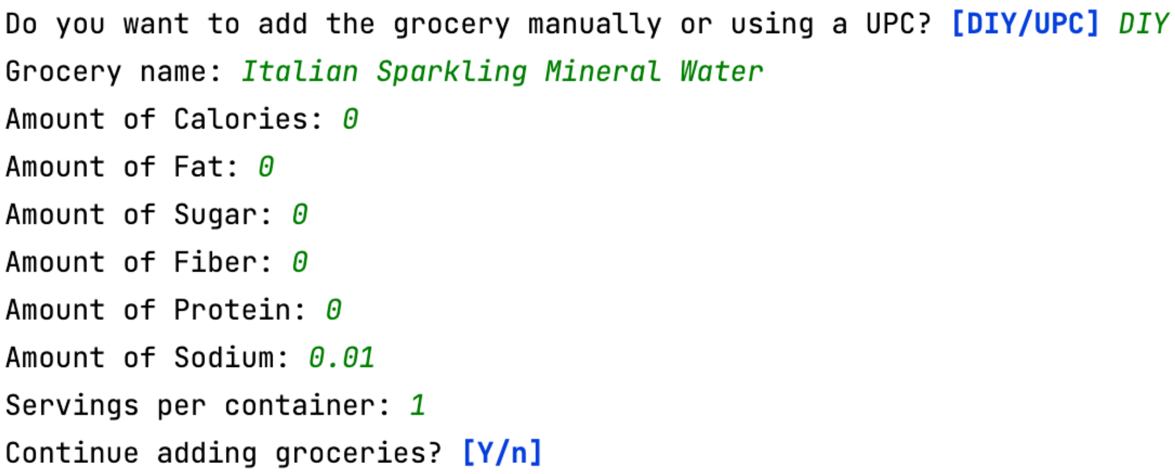
The UPC/EAN is the code displayed underneath the barcode of a grocery, including the small numbers to the side. If the user chooses to add the grocery using the UPC/EAN, then the user is prompted to type in the name of the grocery, the UPC/EAN, and the servings per container. The program will then download and parse the nutrition information, allowing the user to retry if this process fails. Finally, after adding the grocery the user then has a choice to either continue adding groceries or to stop adding groceries.



```
Do you want to add the grocery manually or using a UPC? [DIY/UPC] UPC
Grocery name: Snickers
Enter the product UPC: 040000424314
Downloading...Parsing...Done!
Servings per container: 1
Continue adding groceries? [Y/n]
```

DIY

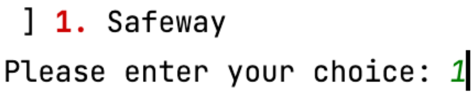
If the user chooses to add the grocery manually, then the user must type in all of the nutrition information manually. All of the nutrition information is in grams.



```
Do you want to add the grocery manually or using a UPC? [DIY/UPC] DIY
Grocery name: Italian Sparkling Mineral Water
Amount of Calories: 0
Amount of Fat: 0
Amount of Sugar: 0
Amount of Fiber: 0
Amount of Protein: 0
Amount of Sodium: 0.01
Servings per container: 1
Continue adding groceries? [Y/n]
```

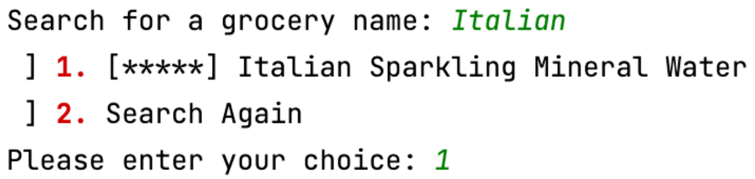
Receipts

When the user chooses the *add receipts to store* mode, the user must first choose the store the receipt is added to.



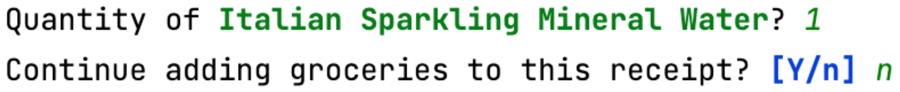
```
 ] 1. Safeway
Please enter your choice: 1
```

After choosing the store, the user then types in the grocery name that is added into the receipt. The user is then given a list of the top five most probable grocery names to choose the correct grocery meant to be added into the receipt. The number of stars represents the strength of the match. This feature was created in case the user misspells a grocery name or doesn't want to type the full name of the grocery.



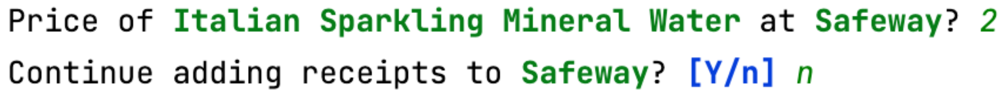
```
Search for a grocery name: Italian
 ] 1. [*****] Italian Sparkling Mineral Water
 ] 2. Search Again
Please enter your choice: 1
```

The user is then prompted for the quantity of the grocery added and whether the user has more groceries to add into the receipt.



```
Quantity of Italian Sparkling Mineral Water? 1
Continue adding groceries to this receipt? [Y/n] n
```

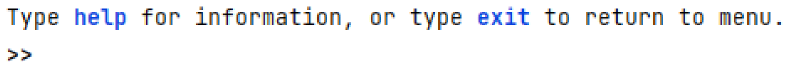
Finally, the user is prompted for the price of any groceries that were added into the receipt that do not already have a price associated with the given store, and whether the user has more receipts to add into the database.



```
Price of Italian Sparkling Mineral Water at Safeway? 2
Continue adding receipts to Safeway? [Y/n] n
```

Analysis

This is the main method by which the database can be queried. Upon entering analysis mode, the user will be shown a command line where custom analysis commands can be entered.



```
Type help for information, or type exit to return to menu.
>>
```

Syntax

In general, the syntax is as follows:

```
<command> [, grocery [filters…]] [, store [filters…]] [, receipt [filters…]]
```

Where [...] indicates optional fields. With this syntax, users may run any arbitrary command on a specified subset of the database information, filtering the groceries, stores, and receipts according to their needs.

Commands

The list of valid analysis commands and their aliases are as follows:

- **Totals**
  - `total-nutrition, nutrition`
  - `total-servings,  servings`
  - `total-quantity,  quantity`
  - `total-price,     price`
  - Calculates the total of the given parameter over items that pass the given filters. Useful for determining cumulative data, such as how much has been spent on groceries over a certain period of time, or how many servings have been bought.
- **Average by price**
  - `average-nutrition-per-price, nutrition/price, nutrition/$`
  - `average-servings-per-price,  servings/price,  servings/$`
  - `average-quantity-per-price,  quantity/price,  quantity/$`
  - Calculates the average of the given parameter over items that pass the given filters, when divided by the total price. Helps the user determine how relatively expensive food items are within the database via three different metrics.
- **Average by day**
  - `average-nutrition-per-day, nutrition/day`
  - `average-servings-per-day,  servings/day`
  - `average-quantity-per-day,  quantity/day`
  - `average-price-per-day,     price/day,      $/day`
  - Calculates the average of the given parameter, when divided by the total number of days spanned by the receipts that pass the given filters. Allows the user to get an idea of their food consumption per day, as well as the price of food on a daily basis.
- **Lists**
  - `list-groceries`
  - `list-stores`
  - Lists all unique items of the given type that pass the given filters. Useful for searching for existing items in the database, querying what items come from what stores,, or seeing which groceries and stores pass the given filters.
- **Meta**
  - **`exit`**
  - Leaves analysis mode and returns to the main menu
  - **`help`**
  - Prints out a handy cheat-sheet of the command syntax, analysis types, and filter types.

Filters

The filter specification is order-agnostic, so filters and filter sections may be input in any order. The list of valid filters are as follows:

- **Store**
  - `name <underscore_separated_name>`
  - Matches any store whose name contains the case-insensitive string, where underscores represent spaces. For example, *`market`* will match both *Yoke's Fresh Market* and *Union Market*, whereas *`mitchell's_harvest_foods`* will match only *Mitchell's Harvest Foods*.
- **Receipt**
  - `between <start-date> <end-date>`
  - Matches any receipt whose dates fall between the specified endpoints, in the yyyy-mm-dd format.
- **Grocery**
  - `name <underscore_separated_name>`
  - Matches any grocery whose names contain the given case-insensitive string. Follows the same rules as *Store name* above.

When multiple filters are specified in one section, or multiple sections are used, the items will pass only if *all* other filters pass as well. Specifying multiple filters is thus a way to "and" many filters together.

The structure of our code is built such that adding new filters is trivial, and would mesh perfectly with previous filters. Even with this set of relatively few filters however, our app is feature complete and allows for complex analysis as seen below.

Example analyses

```
>> nutrition, store name Safeway, receipt between 2022-01-01 2022-05-01
<< Calculate the total nutrition of all food purchased from Safeway between Jan
1st and May 1st
<< Nutrition{Calories: 13540.7, Fat: 1231.0, Sugar: 1891.78, Fiber: 102.5,
Protein: 2315.0, Sodium: 5.052}

>> list-groceries, grocery name Bob's_Red_Mill
<< List all groceries in the database made by "Bob's Red Mill"
<< * Bob's Red Mill Golden Flax Seed
<< * Bob's Red Mill Vital Wheat Gluten
<< * Bob's Red Mill Couscous

>> quantity, grocery name Garbanzo
<< List the number of groceries matching "Garbanzo" that have been bought
<< 17.0 groceries.
```

```
>> servings/price, grocery name Tortilla, store name Safeway
>> servings/price, grocery name Tortilla, store name Grocery_Outlet
<< Compare the relative unit price of groceries matching “Tortilla” between
Safeway and Grocery Outlet (servings per dollar, higher is better!)
<< 5.0 servings.
<< 7.0 servings.

>> nutrition/day
<< Get the average nutrition purchased per day. This can be used as an estimate
of nutrition intake, assuming the database is complete
<< Nutrition{Calories: 1987.7, Fat: 31.1, Sugar: 25.45, Fiber: 30.33, Protein:
57.9, Sodium: 0.152}

>> list-stores, grocery name Apple
<< List every store from which an apple has been purchased
<< * Safeway
<< * Union Market
<< * Mitchell’s Harvest Foods
```