GroceryDatabase Documentation:

Definition Of The Problem:

Input:	Process:	Output:
Navigation choices Items: Brand (optional) Name Price Quantity	Navigate Sort Items Store Items Search Items Manage user state Add/Edit/Delete items Replace deleted items as to not change index	Prompts Searched + Sorted Inventory

Pseudocode:

1. Initialize constants:
2. PASSWORD = "password123"
3. STARTING_PROMPT = "Welcome to Grocers Database!\n------------------------------\nThis is a program used to keep track of item batches in a grocery store.\n\nWould you like to:"
4. STARTING_OPTIONS = "1). Enter user mode\n2). Enter manager mode\n3). Quit the program"
5. MANAGER_OPTIONS = "1). Edit an existing item listing\n2). Add a new item\n3). Delete an item\n4). Display inventory\n5). Logout"
6. MANAGER_MODE = "User mode:\n\nWould you like to:"
7. USER_MODE = "User mode:\n\nWould you like to:"
8. GENERIC_ITEM_OPTIONS = "1). Item Name\n2). Item quantity\n3). Price"
9. SORT_AND_SEARCH_OPTIONS = "1). Name\n2). ID\n3). Price\n4). Quantity\n5). Brand"
10. SEARCH_ORDER_OPTIONS = "1). A-Z/Highest to Lowest\n2). Z-A/Lowest to Highest"
11. HIGH_TO_LOW = 1
12. LOW_TO_HIGH = 2

```
13.       DISPLAY_FORMAT = "ID: %08d\nItem: %s\n~ ~ ~ ~ ~ ~ ~ ~ ~ ~
   ~ ~ ~ ~ ~ ~ ~\nPrice: $%.2f\nQuantity:
   %d\n--------------------------------"
14.
15.   Initialize variables:
16.       managerChoice = 1
17.       selectChoice, delChoice, editChoice = 0
18.       editItemInput, searchTerm = ""
19.       itemElements = []
20.       itemSet = false
21.
22.   Create an empty inventory list
23.
24.   Function addDefaultItems():
25.       Initialize the inventory list with default grocery items
26.
27.   Function getPassword():
28.       Repeat until user inputs the correct password or chooses
   to quit:
29.           If passwordInput equals PASSWORD:
30.               Reset passwordInput and return MANAGER_MODE
31.           Else If passwordInput equals "Q":
32.               Return STARTING_MENU
33.           Else:
34.               Prompt the user for the password input
35.               Store the user's input in passwordInput
36.               Return INPUTTING_PASSWORD
37.
38.   Function startingMenu():
39.       Display STARTING_PROMPT
40.       Get user's choice from STARTING_OPTIONS
41.       Switch user's choice:
42.           Case 1:
43.               Return USER_MODE
44.           Case 2:
45.               Return INPUTTING_PASSWORD
46.           Default:
47.               Return QUIT
48.
49.   Function prompts() for Manager:
```

```
50.       Display MANAGER_MODE
51.       Get user's choice from MANAGER_OPTIONS
52.       Switch user's choice:
53.           Case 1:
54.               Call editItem()
55.           Case 2:
56.               Call addItem()
57.           Case 3:
58.               Call deleteItem()
59.           Case 4:
60.               Get displayChoices from
   Input.getDisplayChoices(DISPLAY_OPTIONS, SORT_AND_SEARCH_OPTIONS,
   SEARCH_ORDER_OPTIONS)
61.               Call displayItems(displayChoices)
62.           Default:
63.               Return STARTING_MENU
64.       Display "Task performed successfully"
65.       Return MANAGER_MODE
66.
67.   Function prompts() for User:
68.       Display USER_MODE
69.       Get displayChoices from
   Input.getDisplayChoices(DISPLAY_OPTIONS, SORT_AND_SEARCH_OPTIONS,
   SEARCH_ORDER_OPTIONS, "3). Logout")
70.       If displayChoices[0] equals 3:
71.           Return STARTING_MENU
72.       Call displayItems(displayChoices)
73.       Return USER_MODE
74.
75.   Function editItem():
76.       Get selectChoice as the user's choice for the item to edit
77.       Display "What would you like to edit?"
78.       If inventory[selectChoice - 1] is an instance of
   BrandedItem:
79.           Get editChoice from user using GENERIC_ITEM_OPTIONS
   and "4). Brand" (max choice 4)
80.       Else:
81.           Get editChoice from user using GENERIC_ITEM_OPTIONS
   (max choice 3)
82.       Display "What would you like to change that to?"
```

```
83.       Get editItemInput from the user
84.       Call inventory[selectChoice - 1].edit(editItemInput,
   editChoice)
85.
86.   Function addItem():
87.       Get itemElements from user by splitting a string input
   using ", "
88.       Initialize newItem as GenericItem or BrandedItem based on
   the length of itemElements
89.       Repeat for each index i from 0 to the size of inventory:
90.           If inventory[i] is null:
91.               Set inventory[i] to newItem
92.               Call inventory[i].changeId(i + 1)
93.               Set itemSet to true
94.               Break the loop
95.       If itemSet is false:
96.           Add newItem to inventory
97.
98.   Function deleteItem():
99.       Get delChoice as the user's choice for the item to delete
100.      Set inventory[delChoice - 1] to null
101.
102.  Function displayItems(displayChoices):
103.      Set displayChoice, searchChoice, sortChoice, sortOrder
   from displayChoices
104.      If displayChoice equals 2:
105.          Display "What is your search term?"
106.          Get searchTerm from the user
107.      Else:
108.          Set searchTerm to "No Search Term"
109.      Set searchedAndSortedItems to the result of calling
   sortAndSearch(inventory, searchTerm, searchChoice, displayChoice
   equals 1, sortOrder, sortChoice)
110.      For each item in searchedAndSortedItems:
111.          Call item.display()
112.
113.  Function sortAndSearch(inventory, searchedTerm,
   searchedVariable, displayAll, searchOrder, sortedVariable):
114.      Set inventoryOutput to the result of calling
   prepare(inventory, searchedVariable, sortedVariable)
```

```
    Sort inventoryOutput based on searchOrder and sortOrder using the merge sort algorithm
    If not displayAll:
        Filter inventoryOutput to contain only items with searchVariable containing searchedTerm
    Return inventoryOutput

Function search(inventory, searchedTerm):
    Create an empty temporaryInventory list
    For each item in inventory:
        If item.searchVariable contains searchedTerm (case-insensitive):
            Add item to temporaryInventory
    Convert temporaryInventory to an array and return it

Function prepare(inventory, searchedVariable, sortedVariable):
    Create an empty temporaryInventory list
    For each item in inventory:
        If item is not null:
            Add item to temporaryInventory
            Call item.prepareForSearch(searchedVariable)
            Call item.prepareForSort(sortedVariable)
    Convert temporaryInventory to an array and return it

Function merge(toSort, leftMost, middle, rightMost, sortOrder):
    // Merge two subarrays of toSort
    // Sort order: HIGH_TO_LOW or LOW_TO_HIGH
    // ...
    // (Implementation of the merge step in merge sort algorithm)

Function sort(toSort, leftMost, rightMost, sortOrder):
    // Recursively sort an array using merge sort
    // Sort order: HIGH_TO_LOW or LOW_TO_HIGH
    // ...
    // (Implementation of the merge sort algorithm)

Class GenericItem:
```

```
149.      Fields: itemType, id, quantity, price, searchVariable,
  sortVariable
150.      Method prepareForSearch(varSelect):
151.          Set searchVariable based on varSelect
152.      Method prepareForSort(varSelect):
153.          Set sortVariable based on varSelect
154.      Method edit(edit, varSelect):
155.          Update the item based on varSelect
156.      Method changeId(changeTo):
157.          Update the item's ID to changeTo
158.      Method display():
159.          Display the item's details using DISPLAY_FORMAT
160.
161.  Class GrocersDatabase:
162.      Constants: STARTING_MENU, INPUTTING_PASSWORD,
  MANAGER_MODE, USER_MODE, QUIT
163.      Main Function:
164.          Initialize userState to STARTING_MENU
165.          Call addDefaultItems() to populate the inventory
166.          Repeat while userState is not QUIT:
167.              Switch userState:
168.                  Case MANAGER_MODE:
169.                      Set userState to the result of calling
  prompts() in
170.
171.  Class Input:
172.      Fields: stdIn (a Scanner object)
173.      Method getString():
174.          Return a string input from the user
175.
176.      Method getListChoice(list, listLength):
177.          Initialize input as "no input yet"
178.          Initialize output as 0
179.          Display list
180.          Get input from the user
181.          Repeat until the input is a valid choice:
182.              While input is empty or contains non-digit
  characters:
183.                  Display an error message
184.                  Get input again
```

```
185.             Convert input to an integer and store it in output
186.             If output is less than 1 or greater than
  listLength:
187.                 Set input to "invalid"
188.         Return output
189.
190.     Method getDisplayChoices(display, itemPrompts,
  searchOrder):
191.         Initialize returns as an array of 4 integers
192.         Set returns[0] to the result of calling
  getListChoice(display, 2)
193.         If returns[0] equals 2:
194.             Display "What would you like to search by?"
195.             Set returns[1] to the result of calling
  getListChoice(itemPrompts, 5)
196.         Display "What would you like to sort by?"
197.         Set returns[2] to the result of calling
  getListChoice(itemPrompts, 5)
198.         Display "What order should the sort be in?"
199.         Set returns[3] to the result of calling
  getListChoice(searchOrder, 2)
200.         Return returns
201.
202.     Method getDisplayChoices(display, itemPrompts,
  searchOrder, extraOptions):
203.         Initialize returns as an array of 4 integers
204.         Set returns[0] to the result of calling
  getListChoice(display + extraOptions, 3)
205.         If returns[0] is greater than 2:
206.             Return returns
207.         Else If returns[0] equals 2:
208.             Display "What would you like to search by?"
209.             Set returns[1] to the result of calling
  getListChoice(itemPrompts, 5)
210.         Display "What would you like to sort by?"
211.         Set returns[2] to the result of calling
  getListChoice(itemPrompts, 5)
212.         Display "What order should the sort be in?"
213.         Set returns[3] to the result of calling
  getListChoice(searchOrder, 2)
```

```
214.          Return returns
215.
216.      Method validItem(newItem):
217.          Try to split newItem into testedStrings
218.          Set correctLength based on the length of testedStrings
219.          Try to parse price and quantity from testedStrings
220.          If any exceptions are thrown:
221.              Return false
222.          Return correctLength
223.
224.      Method splitNewItem():
225.          Set PROMPT as the input prompt for a new item
226.          Display PROMPT
227.          Get newItem from the user
228.          Repeat until newItem is a valid item:
229.              Display an error message
230.              Get newItem again
231.          Split newItem using ", " and return the result as an
   array of strings
232.
233.  Class BrandedItem extends GenericItem:
234.      Field: DISPLAY_FORMAT, brand
235.
236.      Constructor BrandedItem(brand, itemType, price, quantity,
   id):
237.          Call the superclass constructor with itemType, price,
   quantity, and id
238.          Set the brand of the branded item
239.
240.      Method prepareForSearch(varSelect):
241.          Call the superclass method prepareForSearch(varSelect)
242.          If varSelect is 5:
243.              Set searchVariable to the brand
244.
245.      Method prepareForSort(varSelect):
246.          Call the superclass method prepareForSort(varSelect)
247.          If varSelect is 5:
248.              Set sortVariable to the brand
249.
250.      Method edit(edit, varSelect):
```

```
251.          Call the superclass method edit(edit, varSelect)
252.          If varSelect is 5:
253.              Set the brand to edit
254.
255.      Method display():
256.          Display the branded item's details using
   DISPLAY_FORMAT with brand
257.
```