Project 2 - App Hash Table

Delaware Technical Community College

Program Specifications:

One of the most interesting implementations of the dictionary ADT is using a hash table, which, on average, can be very, very fast. Due to the O(1) average insert, remove, and search, hash tables are one of the most popular data structures to store information.

First, you will implement a dictionary using a simple hash function using the sum of the ASCII values of each character in the string key. This program does NOT need to deal with collisions.

Secondly, you will write a simple program that will import a CSV file containing apps from the Apple App Store. Each app will be inserted into the dictionary using the title as the key and an App object as the value. Your program should have a simple interface that can search for an App by title and display information about the app.

Finally, write a brief reflection on the efficiency of this implementation answering these questions. Why were there were so many collisions? What would be the best array size for the hash table for this program?

Notes:

- Your output does NOT need to match the samples below exactly. The sample run output is given for reference only.
- The CSV can be downloaded directly from D2L or https://www.kaggle.com/tristan581/17k-apple-app-store-strategy-games/downloads/17k-apple-app-store-strategy-games.zip/3
- The CSV has over 17,000 entries.
- The values of each character of the word “Sudoku” are:
    - S = 83
    - u = 117
    - d = 100
    - o = 111
    - k = 107
    - u = 117
- The hash function will simply be the sum of each character based on their ASCII values.
    - For example, “Sudoku” should be inserted into the index 635 since (83+117+100+111+107+117).

- Your dictionary's constructor should take one parameter - array size of the hash table.
    - If the array size for the dictionary is less than 635, then the "Sudoku" app should be inserted into 635%array_size. For example, if the array size were 400, then it would instead be inserted into 635%400 or 235.

To Do:

- Design and implement the dictionary using a hash table implementation.
- Design and implement a driver class to import all the apps into the dictionary and allow a user to search for an app. The search result should be, at minimum, the name of the app and its rating.
- The program should display how many collisions took place during the importing process.
- The program will return an App that matches the hash value even if it is the incorrect result; please see the sample output for an example of this.
- Write a brief reflection answering the questions in the Program Specifications section.
    - Please include the reflection as a comment on the top of your driver class

Sample Code:

```
Dictionary<String, App> appList = new HashTable<>(400);

String userInput = "";
do {
   System.out.println("Please enter an app's name to view (hit enter to exit): ");
   userInput = scan.nextLine();
   System.out.println("App name: User Rating");
   System.out.println(appList.search(userInput));
}while(!userInput.isEmpty());
```

Sample Output:

** Notice in the output below that the search for "StoneCross" and "Castleparts" yielded the same result incorrectly. This is due to collisions not being handled correctly.

```
##############################################
Finished processing csv file...
Total number of apps: 17007
Total number of collisions: 16607
##############################################
Please enter an app's name to view(hit enter to exit):
Sudoku
App name: User Rating
Sudoku: 4.0
```

```
Please enter an app's name to view(hit enter to exit):
StoneCross
App name: User Rating
StoneCross: 0.0
Please enter an app's name to view(hit enter to exit):
Castleparts
App name: User Rating
StoneCross: 0.0
Please enter an app's name to view(hit enter to exit):
```