COP3530 Project 5 – Hash Tables

Due Date: Friday, 12/10/2021 11:59 PM

Turn in:

Submit the zipped Eclipse program including at least Project5.java, HashTable.java, and States5.csv. The zip file should be named <your last name>_Project5.zip (for example, Liu_Project5.zip). The States5.csv file contains information about all 50 States in the USA. For each State, in the CSV are its name, capitol, region, US House seats and population (according to the *2020 Census* found in *Wikipedia* at https://en.wikipedia.org/wiki/List_of_states_and_territories_of_the_United_States_by_population), numbers of COVID-19 total cases and deaths (as of Sept 2, 2021, according to *Centers for Disease Control and Prevention* at https://covid.cdc.gov/covid-data-tracker/), median household income (according to the *2020 World Population Review* at *https://worldpopulationreview.com/state-rankings/median-household-income-by-state*), and violent crime rates (according to the *2019 FBI report* at https://ucr.fbi.gov/crime-in-the-u.s/2019/crime-in-the-u.s.-2019, did not find 2020 FBI data for this).

The program should be well documented in the format of doc comments in Java. Detailed formats are found at http://www.oracle.com/technetwork/articles/java/index-137868.html.

Requirements:

Implement a HashTable class. The hash table will be created from States5.csv. Use the following class definition for a node in the linked lists in the table (put this inside your class and do not change it, except to add comments):

```
private class Node {
    String name;
    long population;
    long deaths;
   Node nextNode;

   public Node(String name, long population, long deaths) {
       this.name = name;
       this.population = population;
       this.deaths = deaths;
   }

   public void printNode() {
      System.out.printf("%-30s %-20.2f\n", name,
(double)deaths/population*100000);
   }
}
```

Your **HashTable** class must be implemented using **Separate Chaining** to deal with collisions. For the array of linked lists in the HashTable class, you must use

double-ended singly-linked list, NOT doubly-linked list! Moreover, insertions will happen at the **END** of the linked list, and searching and deletions will need to go through the list. (As the previous projects, you are to implement these data structures in this project, so you must **NOT** use the *LinkedList* class provided by Java.)

The class must use the following **hash function** to calculate the index value for a state: sum up the Unicode values of all of the characters in the state's name strings (including spaces), and then modulus the result with 101 (This means that you need a hash array of size 101, a prime number slightly bigger than twice of the number of State objects.) Unicode value of a character can be retrieved simply by using a type conversion from char to int. You must **NOT** use any hash function provided by Java libraries to compute hash values of state names.

(1). Create the **HashTable** class that implements the following public methods:

1. A no-arg constructor that creates an empty hash table.
2. The method: **public void insert(String state, long population, long deaths)** that will insert a node into the linked list at the proper position in the hash table based on state name string.
3. The method: **public int find(String state)** that will search the linked list at the proper position in the hash table for the state, and if found will return table index or -1 if not found.
4. The method: **public void delete(String state)** that will find and delete the state of the given name from the hash table.
5. The method: **public void display()** that will traverse the table and will print the hash table as follows:

```
0.   Empty
1.   StateName          DR
     StateName          DR
     StateName          DR
2.   Empty
3.   StateName          DR
4.   Empty
5.   Empty
. . .
100. StateName         DR
```

6. The method: **public void printEmptyAndCollidedCells()** that will print the number of empty cells and the number of collided cells in the hash table array. Note that an empty cell is a cell of no state, and that a collided cell is a cell of multiple states.

There are X empty and Y collided cells in the hash table

(2). Create a class called Project5 that will

1. Prompt user to enter the name of the CSV file, e.g., States5.csv, as input to the system.
2. Read the CSV file to create a hash table by calling the **insert** method.
3. Offer the user the following functionalities:

    1) Print hash table
    2) Delete a state of a given name
    3) Insert a state of a given name
    4) Search and print a state and its DR for a given name.
    5) Print numbers of empty and collided cells
    6) Exit

Provide comments in this form for the **HashTable** classes:

Comments for the class:

```
/**
 * Detailed description of the class.
 *
 * @author <your name>
 * @version <date you last changed the class>
 */
```

Public method comments:

```
/**
 * Description of the purpose of the method, the meaning of the
 * input parameters (if any) and the meaning of the return values
 * (if any).
 *
 * @param  parameter  description of the parameter (one for each)
 * @return description of the return value
 */
```

Provide comments in this form for the **Project5** class.

```
/**
 * COP 3530: Project 5 – Hash Tables
 * <p>
 * Description of the class using as many lines as needed
 * with <p> between paragraphs. Including descriptions of the
 * input required and output generated.
 *
 * @author <your name>
 * @version <the date you last modified the program>
 */
public class Project5
{
```

COP3530 Project 5 – Hash Tables

Example Output:

```
COP3530 Project 5
Hash Tables
Enter the file name: States5.csv

There were 50 state records read into the hash table.

1) Print hash table
2) Delete a state of a given name
3) Insert a state of its name, population, and COVID deaths
4) Search and print a state and its DR for a given name
5) Print numbers of empty and collided cells
6) Exit
Enter your choice: 1

0.   Empty
1.   Empty
…
100. Nebraska       119.25

1) Print hash table
2) Delete a state of a given name
3) Insert a state of its name, population, and COVID deaths
4) Search and print a state and its DR for a given name
5) Print numbers of empty and collided cells
6) Exit
Enter your choice: 2
Enter state name: Alaska

Alaska is deleted from hash table

1) Print hash table
2) Delete a state of a given name
3) Insert a state of its name, population, and COVID deaths
4) Search and print a state and its DR for a given name
5) Print numbers of empty and collided cells
6) Exit
Enter your choice: 2
Enter state name: China

China is not a state

1) Print hash table
2) Delete a state of a given name
3) Insert a state of its name, population, and COVID deaths
4) Search and print a state and its DR for a given name
5) Print numbers of empty and collided cells
6) Exit
Enter your choice: 3
Enter state name: Alaska
Enter state population: 1234567
Enter state COVID deaths: 214987

Alaska is inserted to hash table

1) Print hash table
2) Delete a state of a given name
3) Insert a state of its name, population, and COVID deaths
4) Search and print a state and its DR for a given name
5) Print numbers of empty and collided cells
6) Exit
```

```
Enter your choice: 5

There are X empty and Y collided cells in the hash table

1) Print hash table
2) Delete a state of a given name
3) Insert a state of its name, population, and COVID deaths
4) Search and print a state and its DR for a given name
5) Print numbers of empty and collided cells
6) Exit
Enter your choice: 4
Enter state name: Florida

Florida is found at index 99 with DR of 213.15

1) Print hash table
2) Delete a state of a given name
3) Insert a state of its name, population, and COVID deaths
4) Search and print a state and its DR for a given name
5) Print numbers of empty and collided cells
6) Exit
Enter your choice: 22
Invalid choice enter 1-6: 0
Invalid choice enter 1-6: A
Invalid choice enter 1-6: 6

Have a good day!
```