**[CSE 1310](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/index.html) -** [**Assignments**](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/assignments/index.html) **- Programming Assignment 10**

The assignment will be graded out of 100 points.

All tasks ask you to write code, and specify what name to use for the file where you save that code. You need to use exactly the name that is given (do not change the case, or make any other modification). Remember, the name of the main class must match the filename.

**Task 1 (20 pts.)**

File [task1.java](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/assignments/assignment10/task1.java) contains an incomplete program. The goal of the program is to help users search for statistics of a specific player, stored in file [nba.txt](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/assignments/assignment10/nba.txt). In particular, in the main loop of the program:

1. The user enters part of a player's name (or the entire name).
2. The program looks for all players whose names contain that part. For each such player, the program prints out the stats of that player.

Complete that program, by defining a print\_player\_info function, that satisfies the following specs:

* Function print\_player\_info takes two arguments, called data, player. Argument data is a 2D string array containing the data of the nba.txt spreadsheet. Argument player is a string.
* The function finds all rows in data for which column 0 (in its lower-case version) contains as a substring the string stored in player (also in its lower-case version). For all such rows, it prints the statistics of the corresponding player.

**IMPORTANT: you are only allowed to modify the provided code by writing the print\_player\_info function, and possibly additional auxiliary functions. You are NOT allowed to modify in any way the main function, or any other function that is already provided.**

Hints:

* If you have a string and you want to convert it to lower case, you can use the String.toLowerCase method. For example, the following code sets string s2 equal to "apple".
* String s1 = "AppLE";
* String s2 = s1.toLowerCase();
* If you have a string s1 and a string s2, and you want to see if s2 is a part of s1, in a case-insensitive manner, you can do:
* String y1 = s1.toLowerCase();
* String y2 = s2.toLowerCase();
* if (y2.indexOf(y1) != -1)
* {
* // code here for the case where s2 is part of s1.
* }
* To match the output seen below, allocate 20 spaces to the column name. My printf looks like this:
* System.out.printf("%20s: %s\n", ...
* The function read\_spreadsheet from program nba\_leaders\_2d\_array\_version.java (posted on the course website) can be useful here.
* The names of the columns are stored in the first line of the file. If you are manually typing in things like "minutes per game" in your code, then your solution is way longer than necessary.

An example run of the complete program is shown below.

Enter part of a player's name (or q to quit): bryant

Enter part of a player's name (or q to quit): jord

player: Jordan Hill

games played: 70

minutes per game: 26.8

points: 12.0

field goals made: 5.1

field goal attempts: 11.1

field goal percentage: 45.9

3-pointers made: 0

3-pointers attempted: 0.2

3-pointer percentage: 27.3

free throws made: 1.8

free throws made: 2.4

free throws made: 73.8

rebounds: 2.5

assists: 5.5

steals: 7.9

blocks: 1.5

player: Jordan Clarkson

games played: 59

minutes per game: 25

points: 11.9

field goals made: 4.5

field goal attempts: 10.1

field goal percentage: 44.8

3-pointers made: 0.6

3-pointers attempted: 2.1

3-pointer percentage: 31.4

free throws made: 2.2

free throws made: 2.7

free throws made: 82.9

rebounds: 0.9

assists: 2.3

steals: 3.2

blocks: 3.5

player: DeAndre Jordan

games played: 82

minutes per game: 34.4

points: 11.5

field goals made: 4.6

field goal attempts: 6.5

field goal percentage: 71

3-pointers made: 0

3-pointers attempted: 0

3-pointer percentage: 25

free throws made: 2.3

free throws made: 5.7

free throws made: 39.7

rebounds: 4.8

assists: 10.1

steals: 15

blocks: 0.7

Enter part of a player's name (or q to quit): ebro

player: LeBron James

games played: 69

minutes per game: 36.1

points: 25.3

field goals made: 9

field goal attempts: 18.5

field goal percentage: 48.8

3-pointers made: 1.7

3-pointers attempted: 4.9

3-pointer percentage: 35.4

free throws made: 5.4

free throws made: 7.7

free throws made: 71

rebounds: 6

assists: 7.4

steals: 1.6

blocks: 0.7

Enter part of a player's name (or q to quit): westbrook

player: Russell Westbrook

games played: 67

minutes per game: 34.4

points: 28.1

field goals made: 9.4

field goal attempts: 22

field goal percentage: 42.6

3-pointers made: 1.3

3-pointers attempted: 4.3

3-pointer percentage: 29.9

free throws made: 8.1

free throws made: 9.8

free throws made: 83.5

rebounds: 7.3

assists: 8.6

steals: 2.1

blocks: 0.2

Enter part of a player's name (or q to quit): q

Exiting...

Your program's output should match EXACTLY the format shown above. There should be no deviations, no extra spaces or lines, no extra punctuation in your output. What you see above as uppercase letters should remain uppercase in your output, what you see as lowercase letters should remain as lowercase in your output, what you see as spaces and punctuation should remain exactly as spaces and punctuation in your output.

**Task 2 (20 pts.)**

This task is almost identical to Task 1, except that your function here saves information to a file instead of printing it on the screen.

File [task2.java](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/assignments/assignment10/task2.java) contains an incomplete program. The goal of the program is to help users search for statistics of a specific player, stored in file [nba.txt](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/assignments/assignment10/nba.txt), and save them to an output file called out4.txt. In particular:

1. The user enters part of a player's name (or the entire name).
2. The program looks for all players whose names contain that part. For each such player, the program saves, in file out4.txt, the stats of that player.
3. For example, if the user types in "jord", the program should produce an output file called out4.txt that looks EXACTLY like this [out4.txt](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/assignments/assignment10/out4.txt).

Complete that program, by defining a save\_player\_info function, that satisfies the following specs:

* Function save\_player\_info takes two arguments, called data, player. Argument data is a 2D string array containing the data of the nba.txt spreadsheet. Argument player is a string.
* The function finds all rows in data for which column 0 (in its lower-case version) contains as a substring the string stored in player (also in its lower-case version). For all such rows, it saves the statistics of the corresponding player in file out4.txt

**IMPORTANT: you are only allowed to modify the provided code by writing the save\_player\_info function, and possibly additional auxiliary functions. You are NOT allowed to modify in any way the main function, or any other function that is already provided.**

**Task 3 (60 pts.)**

File [phonebook.java](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/assignments/assignment10/phonebook.java) contains an incomplete program. The goal of the program is to help users create a phonebook, that contains the phone numbers of various people. The program allows the users to:

1. Print the entire phonebook on the screen.
2. Input a new number.
3. Search the phonebook for entries matching a name (or part of a name).

Complete the program, by writing the functions that are missing. In particular, you have to implement:

* **public static String[][] read\_phonebook()**   
  This function reads a phonebook stored in a file called "phonebook.txt", and returns a 2D String array containing the entries in the phonebook. In that array, entry[i][0] is the name in the i-th row of the file, and entry[i][1] is the number in the i-th row of the file. You can assume that phonebook.txt is a CSV file, where every line contains a name and a number. You can use this example [phonebook.txt](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/assignments/assignment10/phonebook.txt) file.
* **public static void save\_phonebook(String[][] data, String new\_name, String new\_number)**   
  This function writes into a CSV file called "phonebook.txt" the phonebook entries stored in the 2D String array. After that, it writes into the CSV file one more line, that contains the string stored in new\_name, followed by a comma, followed by the string stored in new\_number is a phone number.
* **public static void print\_spreadsheet(String[][] data)**   
  This function prints the phonebook entries stored in data, in the format shown at the sample output further down. Use a width of 4 characters for the line number, and a width of 20 characters for each name.
* **public static void search\_data(String[][] data)**   
  This function asks the user to enter part of a name. Then, it scans the phone entries stored in data, to see if any of the names in those entries contains the string that the user entered. Convert strings to lower case to check if a name contains what the user has entered, and use the indexOf method. For all names that match what the user entered, the function prints out the name and corresponding phone entry, in the format shown at the sample output further down. Use a width of 20 characters for each name.

**IMPORTANT: note that the program should update the phonebook.txt file every time that the user enters a new entry.**

An example run of the complete program is shown below.

0: John, 2-3151

1: Mary, 6-1231

2: Bill, 8-9535

3: Ann, 3-1982

4: Pam, 2-3535

5: Jack, 2-6423

6: Will, 2-6442

7: Jake, 5-5555

1: Print phonebook.

2: Input a new entry.

3: Search by name.

q: Quit program.

Please enter an option: 2

Enter a name: Darth Vader

Enter a number: 43234123

1: Print phonebook.

2: Input a new entry.

3: Search by name.

q: Quit program.

Please enter an option: 2

Enter a name: Han Solo

Enter a number: 533412

1: Print phonebook.

2: Input a new entry.

3: Search by name.

q: Quit program.

Please enter an option: 1

0: John, 2-3151

1: Mary, 6-1231

2: Bill, 8-9535

3: Ann, 3-1982

4: Pam, 2-3535

5: Jack, 2-6423

6: Will, 2-6442

7: Jake, 5-5555

8: Darth Vader, 43234123

9: Han Solo, 533412

1: Print phonebook.

2: Input a new entry.

3: Search by name.

q: Quit program.

Please enter an option: 3

Enter part of the name: ann

Ann: 3-1982

1: Print phonebook.

2: Input a new entry.

3: Search by name.

q: Quit program.

Please enter an option: 3

Enter part of the name: an

Ann: 3-1982

Han Solo: 533412

1: Print phonebook.

2: Input a new entry.

3: Search by name.

q: Quit program.

Please enter an option: q

Your program's output should match EXACTLY the format shown above. There should be no deviations, no extra spaces or lines, no extra punctuation in your output. What you see above as uppercase letters should remain uppercase in your output, what you see as lowercase letters should remain as lowercase in your output, what you see as spaces and punctuation should remain exactly as spaces and punctuation in your output.

**Suggestions**

Pay close attention to all specifications on this page, including file names and submission format. Even in cases where the program works correctly, points will be taken off for non-compliance with the instructions given on this page (such as wrong file names, wrong compression format for the submitted code, and so on). The reason is that non-compliance with the instructions makes the grading process significantly (and unnecessarily) more time consuming. Contact the instructor or TA if you have any questions.

**How to submit**

The assignment should be submitted via [Blackboard](http://elearn.uta.edu). Submit a ZIPPED directory called assignment10.zip (no other forms of compression accepted, contact the instructor or TA if you do not know how to produce .zip files).

To create a zipped directory called assignment10.zip, follow these steps:

1. Create a folder called assignment10.
2. Copy to that folder all your Java files.
3. Zip that folder. On Windows, you can zip a folder by right-clicking on the folder, and then selecting Send to->Compressed (zipped) folder.

Your zip file should contain only 3 files: task1.java, task2.java, and phonebook.java. Do not include any text files.