COMP 202– Spring 2016 - Project #1(At-home part)
Due: March 1, 2016

Submission:

- *F drive, under HOMEWORK\Project1*

- *Do the project individually. Otherwise, you risk being penalized of cheating.*
- *Your at-home submissions will be checked against plagiarism*
- *In two cases you will NOT be allowed to attend the in-lab part and your grades will be zero:*
  - *If you submit your projects (at-home part) after the due time.*
  - *If you plagiarize in at-home part*

In this project, you will implement an Address book by the use of binary search trees, BSTs. The project has two parts: 1) Reading a database and making the BSTs 2) Applying a list of queries on the created BSTs.

1) Reading a database and making the BSTs:

In this part, you are given a file named "database.txt" which contains a set of address records i.e. names, surnames and phone numbers with the following format:

<name><space><surname><space><phone number>

Each line contains a single record. Name, surname and phone number have been separated by a single space (**<space>** stands for a single space). The following is an example of database.txt content. The BSTs created according to the file's content have also been provided in Figures 1, 2 and 3.

```
Jim Carrey 0954924809
Anne Hathaway 0648572102
David Beckham 0222301329
Lionel Messi 0159921548
Antonio Banderas 0396408083
```

You should read the file content line by line and make 3 different BSTs corresponding to names, surnames and phone numbers. While reading the records and inserting them into the BSTs, you should consider the order of records in the file i.e. first insert the first record of the file and then the second record and so on. **You need to implement your own BST data structure and its corresponding methods**.

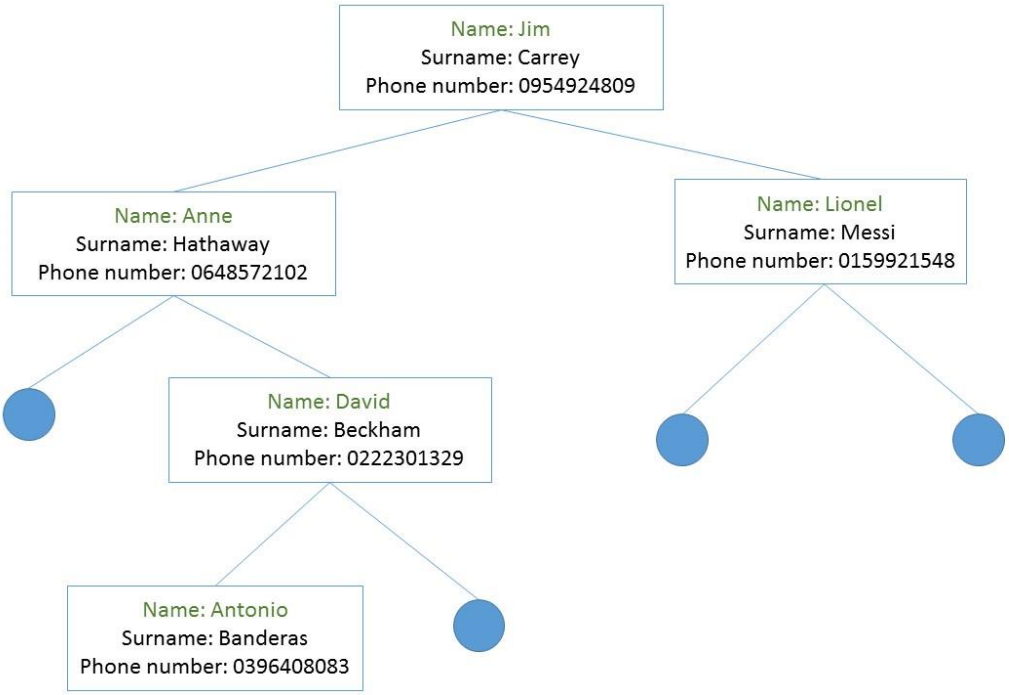

Figure 1) BST#1 based on names

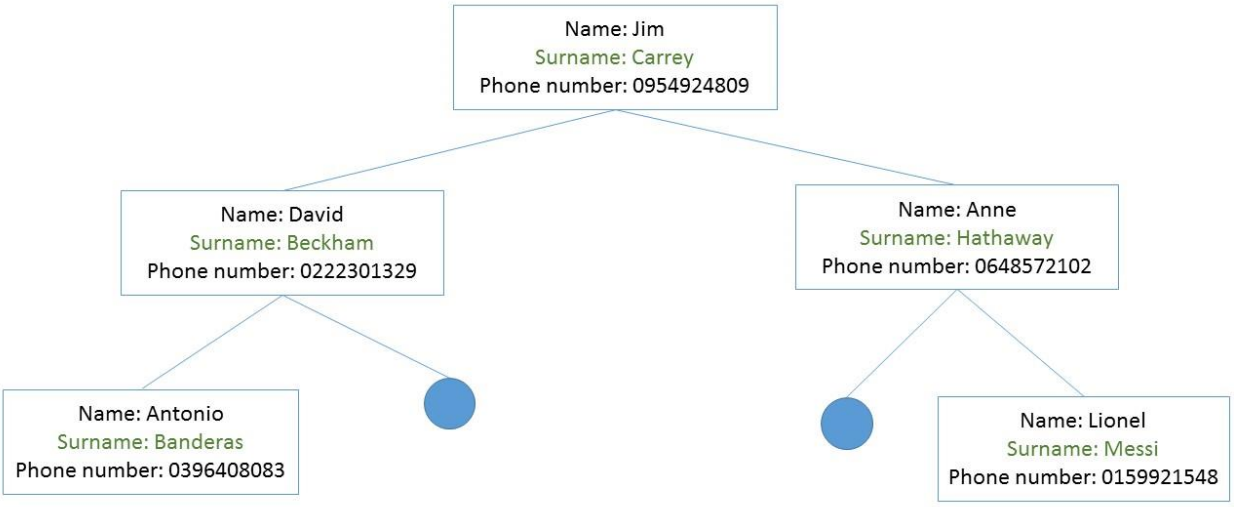

Figure 2) BST#2 based on surnames

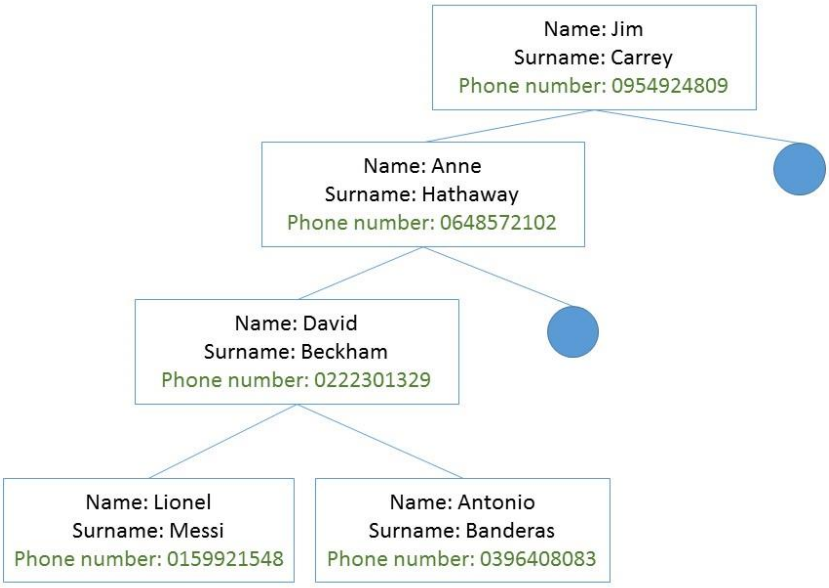

Figure 3) BST#3 based on phone numbers

2) Applying queries on the created BSTs:

In this part, you need to apply a bunch of queries on your BSTs. There is a file named query.txt which specifies a list of queries, one query per line. The queries have one of the following formats:

1. **<Delete><space><name><space><surname><space><phone>**
   Which means, delete the record having the given **name**, **surname** and **phone number**. The record must be deleted from all the trees.
   <Insert><space><name><space><surname><space><phone number>
   Means that insert a node having the given **name**, **surname** and **phone number**. The node must be inserted to all the BSTs.

<space> stands for a single space.

The following is an example of query.txt content (it has two queries but there might be more).

```
delete Anne Hathaway 0648572102
Insert Taylor swift 0614334734
```

Figures 4, 5 and 6 show the results after first query execution i.e. "delete Anne Hathaway 0648572102", and Figures 7, 8 and 9 represent the final BSTs after applying both first and second query i.e. "delete Anne Hathaway 0648572102" and "Insert Taylor swift 0614334734".

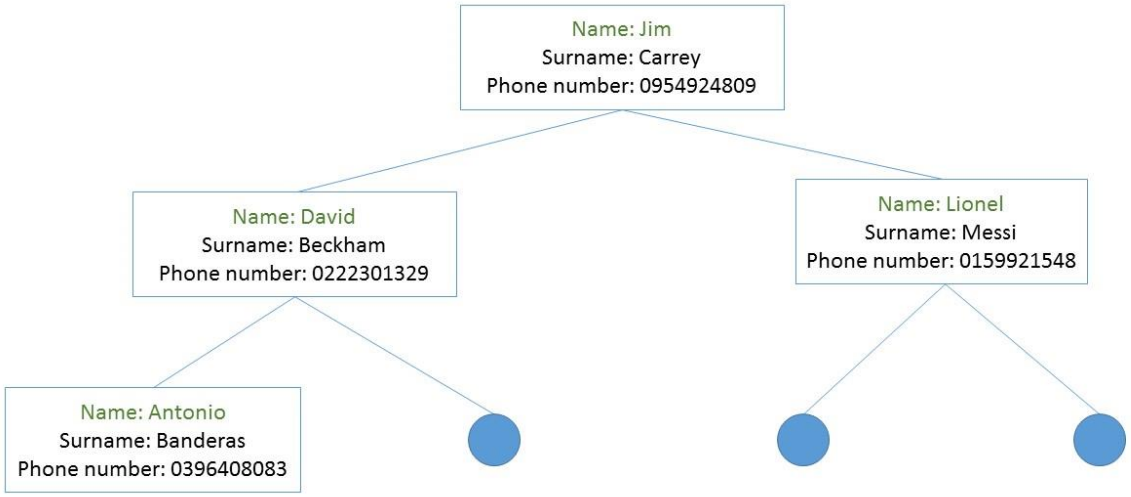

Figure 4) BST#1 after first query

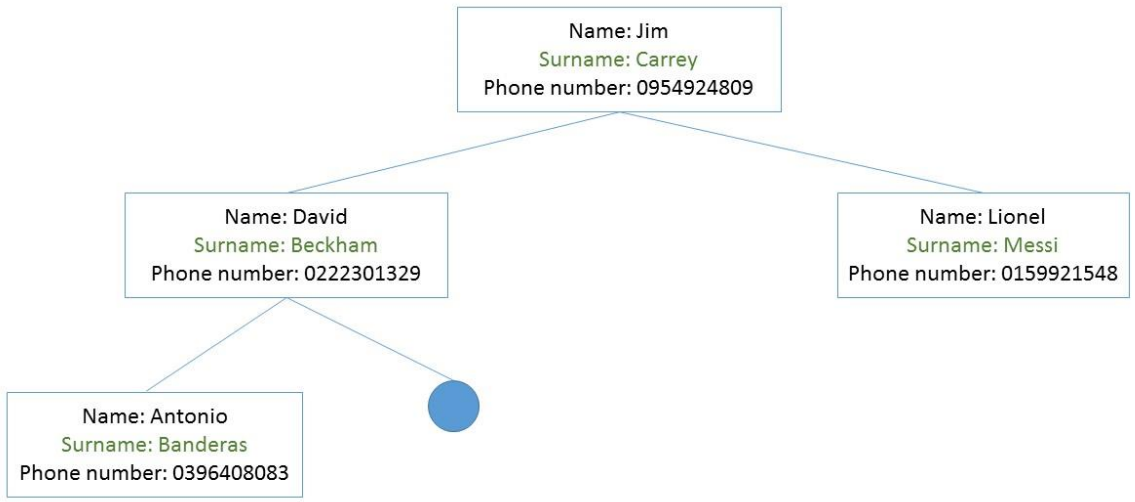

Figure 5) BST#2 after first query

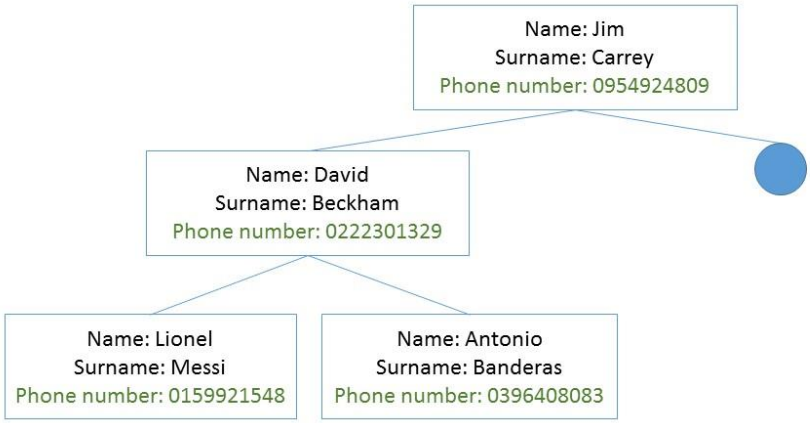

Figure 6) BST#3 after first query

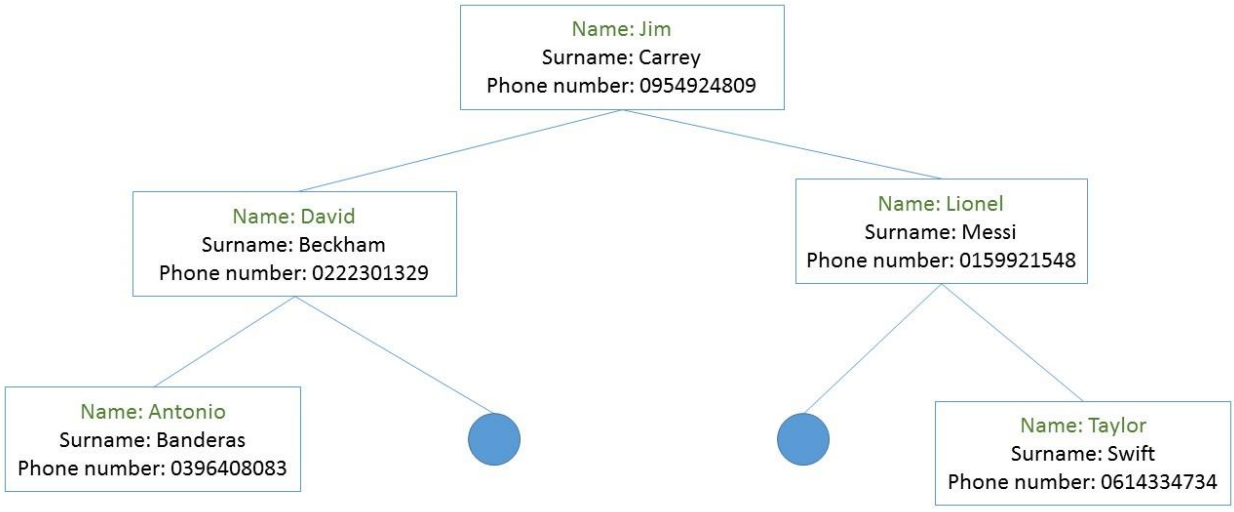

Figure 7) BST#1 after first and second query

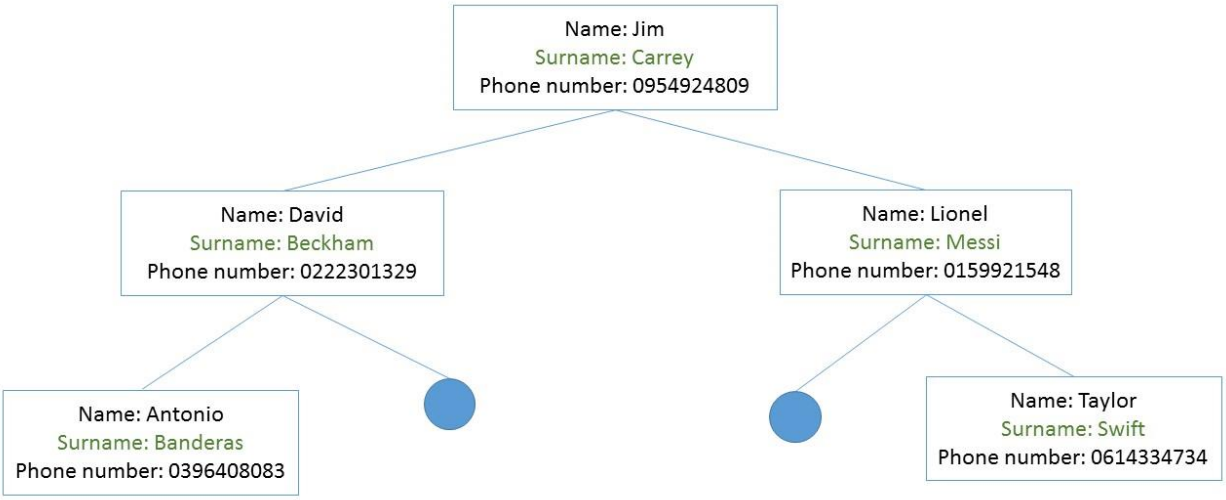

Figure 8) BST#2 after first and second query

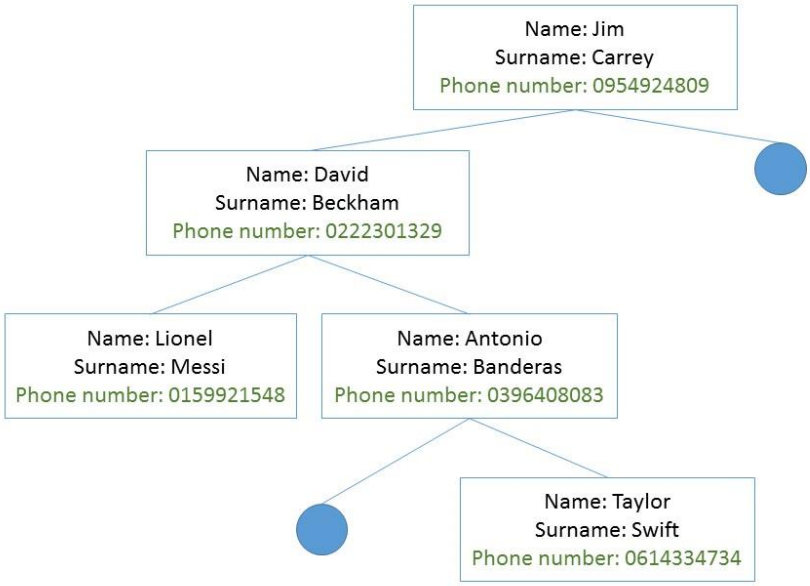

Figure 9) BST#3 after first and second query

Consider the followings while you are writing your program:

- Your code must be written in **Java**
- **Implement your own tree data structure and methods**
- Your code should not be case sensitive
- Your code must be able to read database.txt and query.txt located in your project's root directory
- You must check against possible error conditions and throw an exception if error is encountered