25/10/2017

Programming project

By: Noel Arteche, Pablo Felipe, Edmund Home

Data Structures and Algorithms 2017

Index

Introduction

The following report aims to serve as context for the midterm assignment of the 2017/2018 Data Structures and Algorithms course's programming project at the Faculty of Computer Science at the University of the Basque Country.

The project we are about to introduce is a very basic prototype of a social network, with a view to analyzing and choosing the proper data structures needed to store its information, users' data, etc.

The code attached to this report implements the first five points from the Initial Description document of the project (attached at the end of this report), as well as some other functions not explicitly requested in that document but definitely useful for testing purposes. At this point, our program can do the following:

1. The application presents an initial menu with the different choices for interacting with the social network (point 1 from the Project Description document).
2. There is a function that takes the data from a person and adds him/her to the network. This function does not read any data from Console. This function receives the data as parameters (point 2 from the Project Description document).
3. Using the function implemented in point 2, the program can upload people's data (people.txt) from a file in our directory and add them to the social network (point 3 from the Project Description document).
4. The program can print to a text file a list of the people in the network (point 4 from the Project Description document).
5. Given a text file with a certain format, it can upload the friendships stored in that file to the network (point 5 from the Project Description document).
6. Given a username, the program can show on the standard output the names of that user's friends.
7. Given a username, the program can show the number of friends that user has.
8. The program can return the number of people registered in the network.

This report does not talk about the class created for generating cliques of people, done in a previous assignment, but the code has been attached to the rest of the project.

The implementations have been written in Java and the project has been developed using the Eclipse environment.

Noel Arteche Echeverría

Pablo Felipe Blanco

Edmund Home

San Sebastian, 24th October 2017

First version of the project

We expect the project to evolve as we implement new features and improve our selection of data structures. However, for the moment the project has been implemented in three Java classes that will be discussed in detail in the following sections.

All the data structures used in the project are from Java language's own implementation; mainly array lists (`java.util.ArrayList`).

Class description and implementation

There are three Java classes: `Menu`, `Person` and `PeopleInTheNetwork`. These classes do the following:

Person

The `Person` class encapsulates the Person ADT, that is, the abstract data type capable of storing all the information of a person. Objects of type `Person` have eleven `String` fields to store the users' data (name, surnames, gender, birthdate, birthplace, home, places of study, places of work, favorite movies and group code). All of this fields are stored as one single `String` each. That means that, even if the `movies` field of that user has two or more, they will all come together in one single `String`. In future versions we will change this so that it stores the different movies, places of study etc. properly.

In addition, it includes one more field, `friends`, which is an `ArrayList` of identifiers (`String`) that stores the identifiers of this person's friends.

```java
public class Person {

        //Fields for the PERSON type:
        /* Some of this fields will have to be changed to ArrayList<String>
         * or String[] in order to store more than one film, workplace, etc.
         * For the time being, we just store the raw information as a String.
         */
        private String identifier;
        private String name;
        private String surnames;
        private String gender;
        private String date;
        private String birthplace;
        private String home;
        private String studiedat;
        private String workedat;
        private String movies;
        private String groupcode;
        private ArrayList<String> friends;
```

This class includes a constructor method with arguments whose implementation will not be included in this report, as it is considered trivial.

It also includes two getter methods, `getIdentifier()` and `getFriends()` that return the `identifier` and `friends` fields, respectively.

The `addFriend(String id)` method, given the ID of a friend, checks if it is already in the list and adds it to the friends field:

```java
/* Given the id (String) of a user, it adds it as a friend */
        public void addFriend(String id) {
                if (!friends.contains(id))
                        friends.add(id);
        }
```

`printUserInfo()` prints a line in console for each field of a person (implementation available in the source code).
`printUserId()` prints a line in console with the identifier of a user.
`printFriends()` prints in console the list of friends of a given user:

```
/* Print the list of identifiers of the user's friends. */
      public void printFriends() {
            if (!friends.isEmpty()) {
                  System.out.println(name+" has the following friends: ");
                  for (String a : friends) {
                        System.out.println(a);
                  }
            } else {
                  System.out.println(name + " has no friends.");
            }
      }
```

Finally, `numberOfFriends()` returns the number of friends of a given user, that is, the size of the `friends` field:

```
/* Returns the number of friends of a given user: */
      public int numberOfFriends() {
            return friends.size();
      }
```

PeopleInTheNetwork

The `PeopleInTheNetwork` class is vital in order to maintain the network running. It has been implemented as any other Java class (private fields, public constructor, some getters and setters…) although, for the time being, only one instance of this class will be created (in the `Menu` class) that will store all the information a network has from the beginning to the end of the program.

It consists of only one field, `list`, which is an `ArrayList` of `Person` objects. This way, we will store in it all the users in the network.

```
public class PeopleInTheNetwork {

      //Fields of the objects PeopleInTheNetwork:
      private ArrayList<Person> list;
```

There is a public constructor without parameters (trivial implementation, not included here).
The `getNumberOfUsers()` method returns the size of the list, that is, the total number of users registered in the network:

```
public int getNumberOfUsers() {
      return list.size();
}
```

`idInTheNetwork(String id)` returns true iff the person with the given identifier is indeed in the list:

```
public boolean idInTheNetwork(String id) {
      int i = 0;
      boolean found = false;
      while (i < list.size() && !found) {
            if (list.get(i).getIdentifier().equals(id)) {
                    found = true;
                }
                i++;
            }
            return found;
        }
```

inTheNetwork(Person p) does the same thing, but takes the whole object Person as a parameter instead of the identifier.

```
        public boolean inTheNetwork(Person p) {
            return idInTheNetwork(p.getIdentifier());
        }
```

indexOfId(String id), given a user's ID, returns an integer int that represents its index in the ArrayList. This method is private, as it is related to the class's internal implementation.

```
        /* Given the id (String) of a user, it tells us the index
         * at which we can find that user in the list. -1 if not in the list.
         * ATTENTION: It is private.
         */
        private int indexOfId(String id) {
            int i = 0;
            while (i < list.size()) {
                if (list.get(i).getIdentifier().equals(id)) {
                    return i;
                }
                i++;
            }
            return -1;
        }
```

Given an identifier, getUser(String id) returns the whole Person object that has that identifier from the list.

```
        /* Given the id (String) of a user, it returns
         * the whole user's Person object.
         */
        public Person getUser(String id) {
            return list.get(indexOfId(id));
        }
```

Finally, we have the two most important methods of this class, used by the social network's menu: addPeopleToTheNetwork(String filePath) and addFriendsToTheNetwork(String filePath).

The first one is used to import the people from a text file:

```
        /* Given a file path (String), the method reads the people
         * in the file and adds them to the network as new users.
         */
        public void addPeopleToTheNetwork(String filePath) throws FileNotFoundException{
            //We open the file:
            File fileInstance = new File(filePath);
            Scanner f = new Scanner(fileInstance);
            f.nextLine();
            f.useDelimiter(",");
            //We read the file and create people of type Person.
            //When that user is not in our list, we add it:
            while (f.hasNextLine()) {
                    Person p = new Person(f.next(), f.next(), f.next(), f.next(),
                    f.next(), f.next(), f.next(), f.next(), f.next(), f.next(),
                    f.nextLine());
                    if (!inTheNetwork(p))
                            list.add(p);
            }
            f.close();
        }
```

The second one receives as a parameter the path of a text file with friendships and stores them in the network:

```
/* Given a file path (String), the method reads the identifiers
 * in the file and adds them friend relationships to the network.
 */
public void addFriendsToTheNetwork(String filePath) throw FileNotFoundException {
        File fileInstance = new File(filePath);
        Scanner f = new Scanner(fileInstance);
        f.nextLine();
        f.useDelimiter(",");
        String f1, f2;
        while (f.hasNextLine()) {
                f1 = f.next();
                f2 = f.nextLine();
                f2 = f2.substring(1, f2.length());
                if (idInTheNetwork(f1) && idInTheNetwork(f2)) {
                        list.get(indexOfId(f1)).addFriend(f2);
                        list.get(indexOfId(f2)).addFriend(f1);
                }
        }
        f.close();
}
```

Last but not least, `printListOfUsers(String filePath)` creates a file with all users' identifiers printed on it:

```
/*Given a file path (String), the method creates a file in which
 * it lists the identifiers of all the users in the network:
 */
public void printListOfUsers(String filePath) throws FileNotFoundException {
        File fileInstance = new File(filePath);
        PrintWriter f = new PrintWriter(fileInstance);
        for (Person p : list) {
                f.println(p.getIdentifier());
        }
        f.close();
}
```

Menu

The third class, `Menu`, is the one in charge of creating the menu the user interacts with. The method that creates it, `menu(PeopleInTheNetwork people)`, takes an object of type `PeopleInTheNetwork` that will have been created before. We may even have different objects of this type and therefore open and close menus with different "networks". For the moment, the `main()` method, also included in this `Menu` class, creates just one `PeopleInTheNetwork` object and then calls the `menu(…)` method with that object as a parameter.

The implementation of `menu(PeopleInTheNetwork people)` is the following:

```
public static void menu(PeopleInTheNetwork people) {
        int choice;
        Scanner input = new Scanner(System.in);

        System.out.println("Select one of the following options:");
        System.out.println("1. Load a file of people.");
        System.out.println("2. Load a file of friends.");
        System.out.println("3. Get the number of user in the network.");
        System.out.println("4. Get the list of friends of a given user.");
        System.out.println("5. Get the number of friends of a given user.");
        System.out.println("6. Print a file with the IDs of all users.");
        System.out.println("7. QUIT");

        choice = -5;
        while (choice != 7) {
                choice = input.nextInt();
                switch (choice) {
                case 1:
                        try {
                                System.out.println("Introduce a path for the file
                                with people:");
                                String path = input.next();
                                people.addPeopleToTheNetwork(path);
                                System.out.println("File successfully added!");
                        } catch (FileNotFoundException e) {
                                System.out.println("The given path is invalid or
                                the file does not exist. Try again.");
                                break;
                        }
                        break;
                case 2:
                        try {
                                System.out.println("Introduce a path for the file
                                with friends:");
                                String path = input.next();
                                people.addFriendsToTheNetwork(path);
                                System.out.println("File successfully added!");
                        } catch (FileNotFoundException e) {
                                System.out.println("The given path is invalid or
                                the file does not exist. Try again.");
                                break;
                        }
                        break;
                case 3:
                        System.out.println("There are "+
                        people.getNumberOfUsers() + " users registered in the
                        social network.");
                        break;
                case 4:
                        System.out.println("Introduce the nickname of a user:");
                        String id = input.next();
                        if (people.idInTheNetwork(id) ) {
                                people.getUser(id).printFriends();
                        } else {
                              System.out.println("The given nickname is invalid
                              or not in the network. Please, try again:");
                        }
                        break;
                  case 5:
                        System.out.println("Introduce the nickname of a user:");
                        String ident = input.next();
                        if (people.idInTheNetwork(ident))
                              System.out.println(ident + " has " +
                              people.getUser(ident).numberOfFriends() + "
                              friends.");
                        else
                              System.out.println("The given nickname is invalid
                              or not in the network. Please, try again:");
                        break;
                  case 6:
                        try {
                              System.out.println("Introduce a path for the
                              file:");
                              String path = input.next();
                              people.printListOfUsers(path);
                              System.out.println("File successfully printed!");
                        } catch (FileNotFoundException e) {
                              System.out.println("The given path is invalid or
                              the file does not exist. Try again.");
                              break;
                        }
                        break;
                  case 7:
                        System.out.println("END OF THE PROGRAM");
                        break;
                  default:
                        System.out.println("Invalid selection. Please enter
                        another option:");
                        break;
                  }
            }
            input.close();
      }
```

The main method

Finally, we will have a look at our proposed main method, included for the moment in the `Menu` class. The `main(String[] args)` method prints some generic information about the program on the standard input and then creates an object of type `PeopleInTheNetwork`, called `database`. It then invokes `menu(database)` and stays there until the menu subprogram ends. It then prints some more generic information and, finally, the program stops.

The code is the following:

```
`     public static void main(String[] args) {
            System.out.println("-------------------------------------------------------");
            System.out.println("------ SOCIAL NETWORK - DSA Programing Project 2017-18 ------");
            System.out.println("-------------------------------------------------------");

            PeopleInTheNetwork database = new PeopleInTheNetwork();
```

```
        menu(database);

        System.out.println("================================================================");
        System.out.println("--- Developed by Noel Arteche, Edmund Home and Pablo Felipe ---");
        System.out.println("================================================================");
    }
```

Appendix A – Project Description Document

Programming Project of Data Structures and Algorithms:

Managing a Social Network

Preface

The aim of this programming project is to advance in the study of the design and implementation of data structures and algorithms throughout the course. The development of the project will require, most probably, to rethink program designs and programming alternatives previously made, but those activities are considered part of course goals. All the design decisions made, the project management and the program code will be documented in a *Project Report* which will record all activities and tasks performed. The report will be used to assess the task completed at any time. The Project Report will be structured according to a specification described in another document. Whenever possible use Junit for testing your code.

Requirements and project description

The software application to be developed will have to manage a social network. This social network is formed by people that may be linked among each other if there is a friendship relationship among them. For each person the following items will be recorded:

- identifier (unique)
- name
- surname(s)
- birthdate
- birthplace
- home
- studiedat (he/she could have studied at many places)
- workedat (he/she could have worked at many places)
- movies
- groupcode

New people can be added to the network or deleted from it, besides other actions that may be of interest for the project.

At least, the following actions must be implemented:

1. The application must present an initial menu with the different choices for interacting with the social network.
2. On the social network: take the data from a person and add him/her to the network. This function <u>does not</u> read any data from Console. This function receives the data as parameters.
3. On the social network: upload people's data (*people.txt*) from a file in our directory and add them to the social network. Use the function developed in the previous point.
   The file ***people.txt*** describes the collection of people in the network. Only the people appearing in that file are considered to belong to the social network. Each person identifier has only one occurrence in the file, but the information for that person may be incomplete. Only the identifier is compulsory.

4. On the social network: print out a listing to a text file of the people on the network.

5. The file named ***friends.txt*** (see below) has in each row two person identifiers that represent a friendship relation. The task is to implement an operation that relates them in the network, that is, the relationship has to be stored in the network. Example of the content of the file *friends.txt* (note: friendship relations are reciprocal).

Jon232,Mikel34

Mikel34,Xabi112

Roberto22,Xabi112

Miriam21,Xabi112

Leire1,Miriam21 ...

The file friends.txt describes the friendship relation among people as pairs of identifiers. A pair is not taken into account if either of its identifiers does not appear in the file people.txt. If an identifier appears in the file people.txt but it is not mentioned in the file friends.txt, that person does not have friends in the social network.

6. On the social network: given a person surname, retrieve his/her friends.

7. On the social network: given a city, retrieve all people who were born there.

8. On the social network: retrieve the people who were born between dates D1 and D2, sorted by *birthplace, surname, name*. The order relationship will be implemented according to the lexicographic order (dictionary's order) of the strings used for the attributes.

9. Given a set of identifiers in a file named ***residential.tx***t, recover the values of the attributes *name, surname, birthplace* and *studiedat* of the people on the network whose *birthplace* matches the *hometown* of the people who are described in *residential.txt*. People whose *birthplace/hometown* is unknown do not affect the result of this operation.

10.Two users have the same profile if they match the same collection of favorite movies. Your task is to split the users into classes with the same profile and to build a list of those classes.

11. Six degrees of separation is the theory that everyone on Earth is six or fewer stepsaway, by way of introduction, from any other person in the world, so that a chain of “a friend of a friend" statements can be made to connect any two people in a maximum of six steps (or five intermediaries).
On the social network: given two people of the network, your task is to retrieve the shortest chain that relates them.

12. On the social network: given two people, recover the largest chain of differentpeople linking them (duplicate intermediaries are not allowed)

13. On the social network: retrieve all the cliques of friends (crews) with more than 4friends. A clique is a group of friends in which each person has friendship with each other.

14. To be defined.