Data Structures and Algorithms
Faculty of Computer Science
UPV/EHU

Programming Project

Social Network

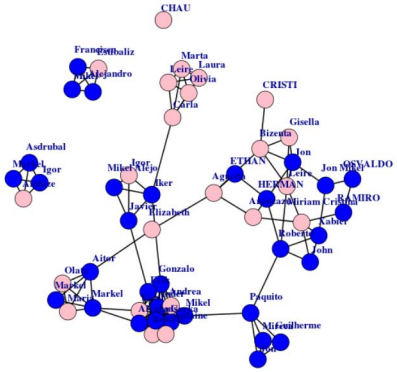
Iyán Álvarez
Davy Wellinger

October 18, 2020

Contents

1. Introduction

This is a short introduction to the **Programming Project** made by Iyán Álvarez and Davy Wellinger.



The goal of this project is the creation a functional **Social Network** that coordinates people and the relations between them. Implementing not only different basic functions, such as **loading** people and relations into the network or **downloading** it's contents into a ".txt" file, but also utilitarian functions, like simple **searching and sorting.**

In order for the user to interact with the network, we have implemented a simple **console application** that gives the user various interaction options. At the end of the project, we might implement a basic **GUI** to replace this system.

From the programming point of view, we're interested in implementing as many types of **data structures** as possible, in order to visualise their different strengths and weaknesses and improve the running cost of the program.

To further understand the data structures implemented in the final version of the project, we decided to first implement everything using **java.util's ArrayList.** This way, upon trying out the different possible data structure, we will be able to see where they excels over each other.

In the end, this Programming Project will be implemented in the most suitable way for the usage of the diverse utilitarian functions: **Searching and Sorting**. And focus on making those processes run smooth and fast.

2. First version of the Project

2.1 Classes design and method implementations

Our project is based on **four classes:**

- **Person:** Person has a lot of data/information and can have relations with other persons on the Social Network.

  Person class has **11 variables**, 3 of them are arrays, to store all the specific information of each person:

  - **identifier:** User ID's for the Social Network.
  - **name:** Person's name.
  - **surname:** Person's surname.
  - **birthdate:** Person's birthdate.
  - **gender:** Person's gender.
  - **birthplace:** Person's birthplace.
  - **home:** Person's home.
  - **studydata:** Person's study data.
  - **workdata:** Person's work data.
  - **movies:** Person's favourite movies.
  - **groupcode:** Groupcode.

  Person class implements some **methods** too:

  - **Getters:** Gets the value stored in a variable.
  - **equals():** Compares a Person with the given Object and returns true if the ID, name and surname are the same.
  - **toString():** Returns a String with the information of the person in the format: idperson,name,lastname,birthdate,gender,birthplace,home,studiedat,workplaces, films,groupcode.

- **Relation:** Relation defines the relation of two people, this relation is mutual.

  Relation class has **2 variables**, to store a relation between two people:

  - **person1:** Person related to person2.
  - **person2:** Person related to person1.

  Relation class implements some **methods** too:

  - **Getters:** Gets the value stored in a variable.
  - **equals():** Compares a relation with the given Object and returns true if the relations have the same meaning.
  - **toString():** Returns a String with the information of the relation in the format: person1,person2.

- **SocialNetwork:** SocialNetwork contains people information and the relations between them if exist.
  SocialNetwork class has **4 variables**, to save person and relations data, a scanner to read user's input and to check that only one instance exists:

  - **personList:** ArrayList that stores the Person instances of the SocialNetwork.
  - **relationList:** ArrayList that stores the Relation instances of the SocialNetwork.
  - **instance:** Controls the existance of an instance of this variable; based on the Singleton concept.
  - **sc:** Scanner to read user's input from console.

  SocialNetwork class implements several **methods** too:

  - **getInstance():** Gets instance of the Social Network, if it does not exist it creates a new one, that is going to be unique and returns it.
  - **initialMenu():** Presents an initial menu with the different choices for interacting with the social network.
  - **printInitialMenu():** Prints the choices of the initial menu.
  - **printAddPersonPeople():** Prints the choices of Add person or people.
  - **printAddRelations():** Prints the choices of Add relations.
  - **printPrintOut():** Prints the choices of Print out people.
  - **printFind():** Prints the choices of Find.

- **selectionInitialMenu():** Request user to select an option and do what was specified in the menu.
- **askForSelectionInput():** Request user for an input, it has to be a number and treats InputMismatchException.
- **addPersonPeopleSelected():** Prints Add person option and performs task.
- **addRelationsSelected():** Prints Add relation option and performs task.
- **printOutSelected():** Prints Print out option and performs task.
- **printFindSelected():** Prints Find option and performs task.
- **addPerson(data):** Adds a person to the SocialNetwork.
- **addPeopleFromFile(filename):** Adds all the people from the file to the Social Network.
- **printPeopleToConsole():** Prints all the people at the SocialNetwork to the console.
- **printPeopleToFile(filename):** Prints all the people at the SocialNetwork to a file.
- **addRelation():** Adds a relation of 2 people.
- **addRelationsFromFile():** Adds all the relations that the specified file contains.
- **existsInSocialNetwork():** Checks if one person exists in the social network given the ID (identifier).

- **SocialNetworkSimulation:** SocialNetworkSimulation simulates how the SocialNetwork works.

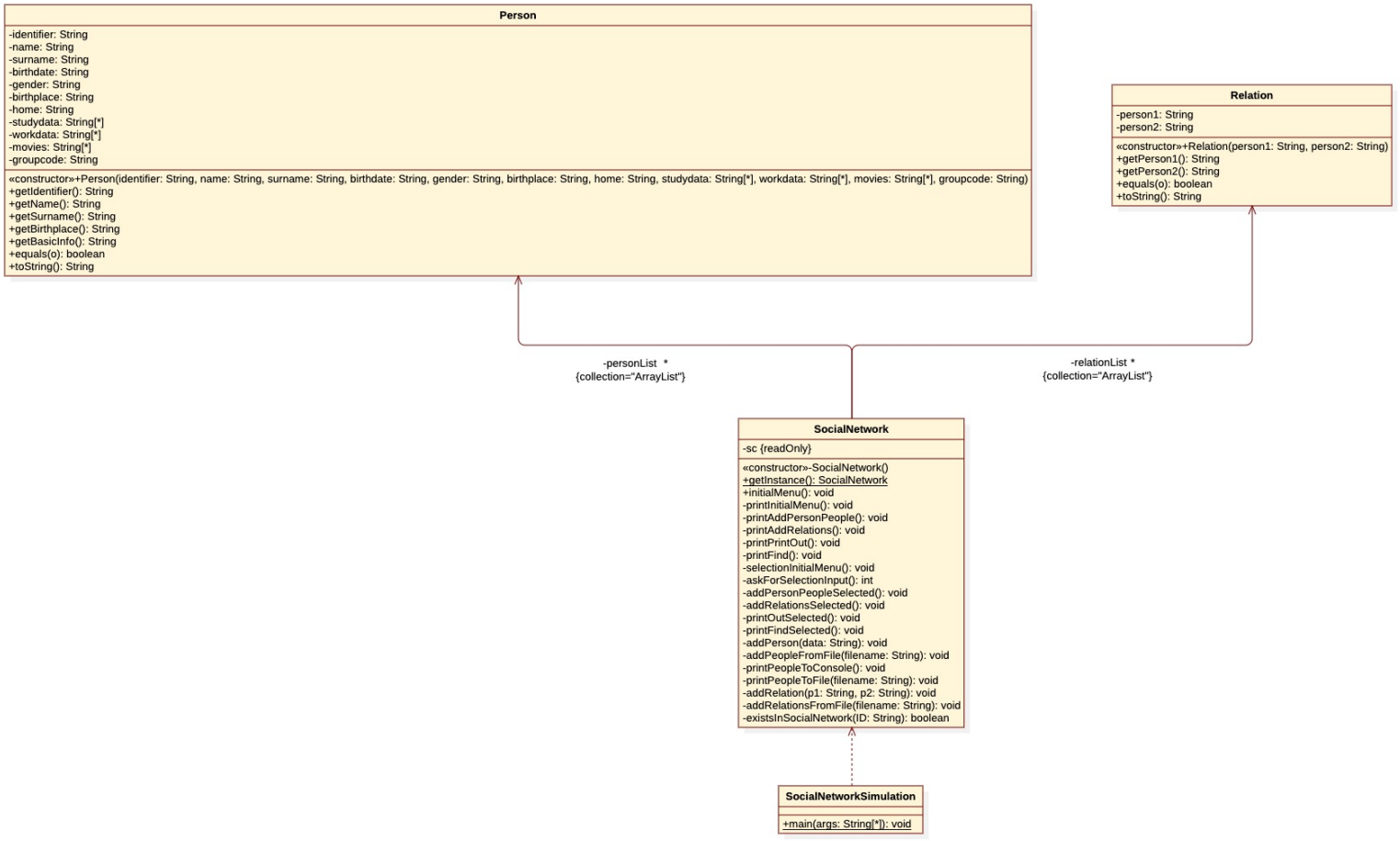
  SocialNetworkSimulation class has **no variables** and has **a main method**.

  - **main(args):** Launches SocialNetwork's initial menu and user can interact with it.

In addition, we have implemented **three exceptions.** All the exceptions extend from Java's RuntimeException, because this errors can be given on the runtime of the program, sometimes depending on the data introduced.

- **PersonAlreadyAtSocialNetwork:** If the person's that wants to be added already exists in the SocialNetwork.
- **PersonNotFoundException:** If at least one person of a relation does not exist in the SocialNetwork.
- **RelationAlreadyAtSocialNetwork:** If the relation's that wants to be added already exists in the SocialNetwork.

2.2 UML design of Social Network



2.3 Description of the data structures used in the project

As mentioned in the introductory description, up to this point, the **Milestone 1**, we have focused on creating a basic framework upon which we will build and expand all functions.

ArrayList
To make things simpler, we decided implementing first only **java.util's ArrayList** and later trying out different data structures, to easily visualize the advantages and disadvantages for our project.

The **ArrayList** class included in java's library **java.util** is easy to use and to keep track of. Being a auto-expanding data structure, we must not worry about running out of space. Similarly, the structure is easy to iterate over and its elements can be inserted anywhere and be identified and deleted easily.

3. Notes of the Meetings

3.1 1st milestone

For the 1st milestone of the project we held 4 meetings:

- **First meeting**
  In our first meeting, we organised the points that we had to work at on different issues at GitHub, and we set up all the environment and the repository on GitHub to work simultaneously on the programming project. We discussed the possible implementation structures, how to organise people, relations and how to implement the menu and some other methods. Apart from that, we decided to develop a individual approach of the Social Network and the classes design and to implement it with our personally preferred approach, to later combine our ideas if needed.

- **Second meeting**
  In the second meeting we compared both of our implementations. We analyzed each other's approaches in some of the topics of the Social Network, and discussed which ideas were the best, combining everything into one project and decided to implement one of them to the master branch of the programming project. We decided to make a fully robust Social Network, treating any exceptions that might be thrown at run-time. This meeting was very productive since we learnt a lot and saw each other's approach.

- **Third meeting**
  In our third meeting, after finishing the implementation of the first milestone of the programming project, we started to work on the documentation and made some experiments with the simple GUI mentioned above. We finally decided to remodel the menu, and let the GUI be an option for a later implementation of the project.

- **Fourth meeting**
  In the fourth meeting we made a final review of the project, and finished the documentation of the 1st milestone of the programming project. We have discussed different options for next implementations and uploaded all the material to eGela.

4. Conclusions

Concrete conclusions will be made after finishing the programming project.

5. Bibliography

- **Data Structures and Algorithms - eGela (UPV/EHU)**