|  |
| --- |
|  |
| Social Network |
| Subject: Data Structures and Algorithms  Degree: Informatics Engineering  Academic Year: 2016/2017   |  | | --- | | Andrea López Rodríguez  Andrea López Figueroa  Ruben Naranjo de las Heras |   School: DIF/FISS |
|  |

**Contents**

**Abstract 3**

1. **Introduction 3**
2. **First version of the project 4**
   1. **Classes 4**
   2. **Description of the data structures used 8**

**Abstract**

This project consists in implementing a functional social network using the data structures we see and learn in class.

In the first version of our project, we’ve tried to implement the first five compulsory methods that are asked in the project’s heading, followed by all the classes, programs and input/output files that are necessary to complete the project in an efficient way.

Also, we included all the information needed to document the project correctly, adding method and class implementations.

1. **Introduction**

The tasks that have been performed in this version of the project are:

* Person class (with all the information required)
* Initial menu (not text based)
* Adding a person to the network
* Upload people’s data from a file and adding them to the network
* Printing a list with the people from the social network
* Relating people and printing it to a file

1. **First version of the project**

This version of the project acts like a draft for our final version, so many of the methods and implementations may be changed in the future in order to make the project more efficient and simple.

These are the points included:

* 1. **Classes**

**2.1.1. Person class:**

The main class that we’ve created for this project is the **Person** class. This class contains all the information needed for creating a person in the social network (identifier, name, surname, birthdate, birthplace, home, where did they study, where did they work at, movies they like and their group code) and all get and set methods. The implementation is the following:

**//attributes of the class**

public class Person implements Comparable<Object>{

private String identifier;

private String name;

private String fSurname;

private String sSurname;

private String gender;

private int birthDay;

private int birthMonth;

private int birthYear;

private String birthplace;

private String home;

private DoubleOrderedList<String> studiedAt = new DoubleOrderedList<String>();

private DoubleOrderedList<String> workedAt = new DoubleOrderedList<String>();

private DoubleOrderedList<String> movies = new DoubleOrderedList<String>();

private String groupCode = "G611630";

**//all the getters and setters needed for every attribute**

public Person(String id) {}

public String getIdentifier() {}

public void setIdentifier(String identifier) {}

public String getName() {}

public void setName(String name) {}

public String getSurnames() {}

public void setSurnames(String[] surnames) {}

public String getBirthdate() {}

public void setBirthdate(int day, int month, int year) {}

public String getBirthplace() {}

public void setBirthplace(String birthplace) {}

public String getHome() {}

public void setHome(String home) {}

public String getStudiedAt() {

public void setStudiedAt(String[] Studied) {}

public String getWorkedAt() {}

public void setWorkedAt(String[] worked) {}

public String getMovies() {}

public void setMovies(String[] Films) {}

public void setGroupCode(String code){}

public String getGroupCode() {}

public String getGender() {}

public void setGender(String gender) {}

**Other methods in the Person class:**

public boolean equals(Object obj) {…}

**//method to see if two people are the same**

public String toString() {…}

**//method to save the information into a String**

public int compareTo(Object anobject) {…}

**//method to compare people**

public String print() {…}

**//method to print all the information**

* + 1. **Class PersonInfo**

This class implements the methods needed to show people’s information in a window. In the main window there is a tab called “People”. When clicking in that tab, a list with all the people in the network will show, and we will be able to click in any person. By doing that, the person’s information will show in a window, which is implemented in this class.

public class PersonInfo {

**//class attributes; for person attribute there is a JTextField object**

private JFrame frame;

private JTextField textFieldName;

private JTextField textFieldSurnames;

private JTextField textFieldBirthdate;

private JTextField textFieldBirthplace;

private JTextField textFieldHome;

private JTextField textFieldStudy;

private JTextField textFieldGender;

private JTextField textFieldWork;

private JTextField textFieldGroup;

private JTextField textFieldId;

**//class methods:**

public static void main(Person P) {…}

**//launches the application**

public PersonInfo(Person P) {

**//creates the application**

private void initialize(Person P) {

**//initializes the values of the window**

* + 1. **Class PersonList**

This class implements all the necessary methods so that the people in the network are actually linked, including the methods to add, remove and update the list of people. It uses the DoubleOrderedList<T> class.

**//class attributes**

public int count;

private DoubleOrderedList<Person> list = new DoubleOrderedList<Person>();

**//creates a doubleorderedlist of Person**

private String path2File = "example path";

**//path where the file with data is located**

**//class methods**

public void initialize() throws IOException{}

**//reads the data from the file and adds it to the list of people**

private void addPerson(Person P){}

**//adds a person to the list**

public void updateList(File inputFile) throws FileNotFoundException{}

**//updates the people in the list, adding the new people from the input file**

private Person createPerson(String input) {

**//creates a person from a String**

public DefaultListModel<String> populate(){}

**//creates a DefaultListModel<T> object to pass to the graphic UI list**

public Person find (int Index) {}

**//given an index, finds the person of the list at that index**

public boolean contains(Person P){}

**//given a person, returns true if the person is in the list**

* + 1. **Class FriendList**

This class implements all the necessary methods so that the people in the network can have friendship relations, including the methods to add, remove and update the list of friends. It uses the DoubleOrderedList<T> class.

**//class attributes**

**private** DoubleOrderedList<String> list = **new** DoubleOrderedList<String>();

**private** PersonList Plist = **new** PersonList();

**//creates a doubleorderedlist of Person**

private String path2Friends = "example path";

**//path where the file with data is located**

**//class methods**

public void initFriends() throws IOException{}

**// reads the data from the file and adds it to the list of friends**

private void updateFriends(File input){}

**//updates the people in the friend list with the relationships from the input file**

* 1. **Data Structures used**

The data structure used in this project is the DoubleOrderedList, which implements the ListADT and uses the DoubleNode, DoubleList and the DoubleOrderedList interfaces.