

QUICK BANKING

CLIENT STATISTICS PROJECT

Interactive Systems And business Consulting

~Team ChaRlize~

Team Members:

### Bianca Tazlauanu

### Catalin Zevedei

27 July 2017

Executive Report

# Description of the project:

The project aims to provide a steady and easy-to-use solution for the banking environment. With this application, banks can now classify their clients by gender, integrate them in an age group and make different statistics with the newly obtained data.

# The challenges of the project:

* Determining the gender of the clients: Making a trustworthy solution to catalogue each name with its corresponding gender.
* Ambiguity of the names: Providing a solution for the client names that can be cataloged as both a male and female name.
* Determining the age of the clients: Assigning a random age for each client depending on their gender.
* Implementing the requested functionalities: Writing appropriate methods for each given requirement for the application.
* Delivering the corresponding tests: Writing tests for each method to validate their functionality and find vulnerabilities in the program.
* Making a graphical user interface: Providing an easy way of interacting with the application for our clients.

# The solutions and benefits of the project:

* To catalog each name with its specific gender, our program offers two solutions, one based on a local name list and one using a much bigger data base which is offered by an online application. Each of them has perks and disadvantages, but they complement each other to offer a complete solution no matter the problem. The program can firstly use the local list of names to determine the gender. To make the search more efficient, different files were used so the search is made by the first letter of the names. If a name starts with letter ‘C’, then the respective name will be searched in its corresponding file which contains only those names starting with that letter. The local files contain a total of 3200 boy names and exceptions. If the results don’t respond to every request the user has, then another solution comes in hand. The program can also search the given list of names on a web site using a get request. At the moment, the database of the website contains 216286 distinct names across 79 countries and 89 languages.
* For those cases when the name can go as both a male and a female name, it is marked in conformity, in this way the user can manually decide what to do with them.
* Proper methods were written for each functionality that was requested. The application can successfully count the number of male and female clients that the bank has or how many of them exist in a specified age group. It can also compute the age average for the males and females and extend this feature so it can search after a given pattern in the names and use it in determining the age.
* Alongside the methods used in the program, adequate tests were written to prove their functionality and flawlessness.
* Using windows form we created a simple and intuitive GUI so every option could be accessed by the cleints.

Abstract

The approach used for this application was to first divide the whole project into smaller tasks and resolve each one of them accordingly to provide the best solution. Because of this reason, the program consists mainly of two classes. One is used to define the characteristics of each client and the other one implements a number of functions that will be applied on the list of clients to extract different features of the list. In the project are also included 26 files ordered alphabetically which contain names starting with a specific letter and a special text file containing exceptions of names (names that can be cataloged both as a male and a female name).

Software Solution

The main project has four parts, two classes used in the main program to generate the list of clients and the statistics which result from the list, and a folder with names and exceptions which is used to generate the gender of the clients. Last but not least, a graphical user interface that puts all the options and functions of the application together.

# The “ClientBase” Class:

Contains the characteristics of a client: first name, gender, age.

With this class these attributes will be saved more easily for each client and then a list will be made with this data.

# The “Client” Class:

Contains the most important methods of the program:

* “*Write*” method:
  + Prints on the screen a given list of clients with their attributes: name, gender and age.
* “*GetGenderStatic*” method:
  + Reads the given list of names from a file and determines for each one of them its gender. Firstly, it looks for the name in the list of exceptions and if it is found it is added in the “ClientList” with gender “X” and age zer0. In the end the user can select if he wants to see only the uncatalogued clients and decide for each of them the gender and age. If the name is not found in the list of exceptions it is searched in the file which contains only male names with the same first letter. If it is found there it is added in the list of clients with the gender “M”, otherwise it is added with the gender “F”.
* “*GetGender*” method:
  + Reads the given list of names from a file and determines for each one of them its gender. This solution uses a get request to the web site genderize.io which contains a very large database. Depending on the received response, the client in then added in the list.
* “*IsAllLetters*” method:
  + Checks if a string contains only letters to prevent further errors of misspelling.
* “*GenerateRandomAge*” method:
  + For each client generates a random age depending on their gender: a number between 38 and 52 for males and between 32 and 42 for females.
* “*GenerateRandomClients*” method:
  + It generates a list of clients choosing randomly between 420 and 750 clients from the original list of the bank.
* “*MaleCount*” method:
  + Counts how many men are in the list of clients.
* “*FemaleCount*” method:
  + Counts how many women are in the list of clients.
* “*MaleCount*” with an integer parameter method:
  + Counts how many men are in the list with a specified age as a characteristic.
* “*FemaleCount*” with an integer parameter method:
  + Counts how many women are in the list with a specified age as a characteristic.
* “*MaleAverage*” method:
  + Counts the age average of all the men in the list of clients. If the number of men is zero, then an error will occur that will be displayed on the screen.
* “*FemaleAverage*” method:
  + Counts the age average of all the men in the list of clients. If the number of women is zero, then an error will occur that will be displayed on the screen.
* “*MaleAverage*” with a string parameter method:
  + Counts the age average of all the men in the client list who have the first name containing the string given as parameter. If the number of men is zero, then an error will occur that will be displayed on the screen.
* “*FemaleAverage*” with a string parameter method:
  + Counts the age average of all the female in the client list who have the first name containing the string given as parameter. If the number of women is zero, then an error will occur that will be displayed on the screen.
* “*FemaleAverageYoungerThan*” with an integer parameter method:
  + Counts the age average of all the women in the client list who are younger than a given parameter. If the number of women is zero, then an error will occur that will be displayed on the screen.
* “*FemaleAverageRange*” with two integers parameters method:
  + Counts the age average of all the women in the client list who have the age between two given parameters. If the number of women is zero, then an error will occur that will be displayed on the screen.

# The “NameList” folder:

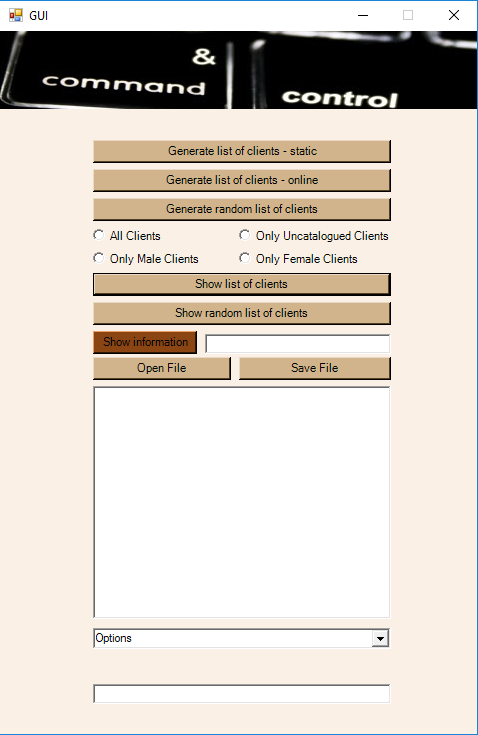
Contains 26 text documents, each one of them having a list of boy names starting with a specific letter. In this way, when searching for a given name, the search will be made easier and faster.

Also, it contains a text file entitled “unisex\_name\_list” which contains name exceptions, names that can be declared as both a male and female name.

# The GUI:

The graphical user interface is easy to use and intuitive. It also doesn’t allow the user to make any mistakes and shows errors if there are any.

The graphical interface of the application



Conclusion

The project provides a simple solution for multiple problems and it manages to cover all the requested requirements:

* Adding all the clients in a list.
* Determining the gender for each client.
* Separating clients who could be catalogued by their gender from whose who could not.
* Setting a random age for each group of clients: men and women.
* Generating a random sample of clients from the original list.
* Applying different methods on the random list to generate different statistics like:
  + The number of women in the client list
  + The number of men in the client list
  + The age average of men and women in the client list
  + The number of women with a specified age
  + The number of men with a specified age
  + The number of women with a specified pattern in their first name
  + The number of men with a specified pattern in their first name
  + The number of women who are younger than a specified age
  + The number of women who have an age between two specified ages