CPE 102

Computer Programming

HW3S23 Report

Arda Ocak - 64220037

PopularName Class

This class stores the ranking in the population, the name and total number of babies with that name which is all of String data type.

```
public class PopularName {
    private String ranking;
    private String name;
    private String number;

    public PopularName(String ranking, String name, String number) {
        this.ranking = ranking;
        this.name = name;
        this.number = number;

    }
```

Figure 1.0

Getters And Setters

These are getter and setter functions of instance variables.

```
public String getRanking() { return ranking; }

public void setRanking(int ranking) { this.ranking = String.valueOf(ranking); }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getNumber() {
    return number;
}

public void setNumber(String number) { this.number = number; }
```

Main Function

In main function there is some parts that I want to explain.

```
ArrayList<String> personArrayList = new ArrayList<>();
ArrayList<PopularName> maleList = new ArrayList<>();
ArrayList<PopularName> femaleList = new ArrayList<>();
```

Figure 2

In this part, I created three array lists. PersonArrayList for the part that program will read the data from csv file and split it then add to this array list. I declared this in String data type because when Scanner read the data from csv file, it'll read the file as string.

openFile() Function

```
public static void openFile(ArrayList<String> personArrayList) {
    try {
        // Ask the user to enter the file name.
        Scanner input = new Scanner(System.in);
        System.out.println("--------------------------------");
        System.out.println("Please enter a File name: ");
        System.out.println("--------------------------------");
        String fileName = input.next();
        // Read the file and store the data in the ArrayList.
        Scanner inputPrinter = new Scanner(new File(fileName));
        System.out.println("File founded.");
        while (inputPrinter.hasNext()) {
            // Read the data line by line.
            String person = inputPrinter.nextLine();
            // Split the data by comma and store it in the ArrayList.
            StringTokenizer tokenizer = new StringTokenizer(person, delim: ",");
            // Add the data to the ArrayList.
            while (tokenizer.hasMoreTokens()) {
                personArrayList.add(tokenizer.nextToken());
            }
        }
        // Close the file.
        inputPrinter.close();
    // Catch the exception if the file is not found.
    } catch (Exception e) {
        System.out.println(e.getMessage());
        System.out.println("Please enter a valid file name.");
        System.exit( status: 0); // Exit the program.
    }
```

Figure 2.1

As you can see from the Figure 2.1, program will read data from the csv file, in order to do this, I used Scanner class to read line by line the csv file. Program will ask for the fileName to user. We must use try-catch blocks because of if there is no file exists called such as filename there will be an error called FileNotFoundException. To avoid this, we use try-catch blocks.

StringTokenizer simply does is its split the lines depending on the delimiter which is taking as parameter. I chose this parameter as ", " (comma). Because of our data is written as **rank, male-name, male-number, female-name, female-number**. To split this to words I used comma as delimiter.

```
for (int i = 0; i < personArrayList.size(); i++) {
    if (i % 5 == 1) {
        maleList.add(new PopularName(
                personArrayList.get(i - 1),
                personArrayList.get(i),
                personArrayList.get(i + 1)
        ));
    }

    if (i % 5 == 3) {
        femaleList.add(new PopularName(
                personArrayList.get(i - 3),
                personArrayList.get(i),
                personArrayList.get((i + 1)
        )));
    }
}
```

Figure 2.2

In the code snippet above, program creating new PopularName objects and adds to our gender lists. In the below, I showed you how this code snippet works.

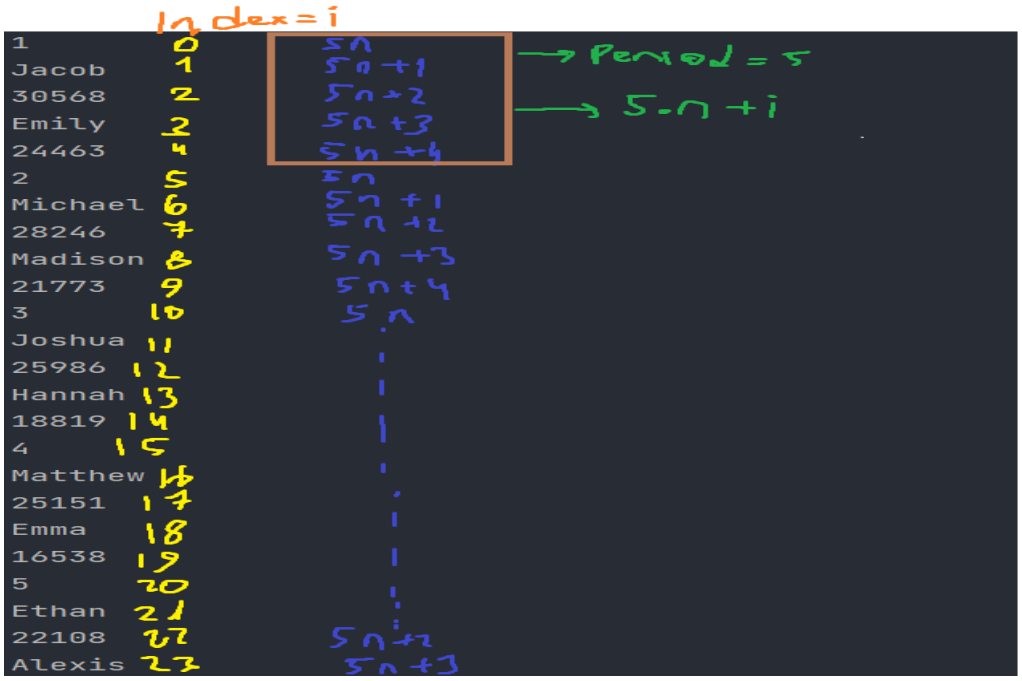

Figure 2.3

Our data is like on the left of this screenshot after we split it. As you can see, each 5 elements it turns back to the ranking (the period is 5). This means that 5n+i, (n can be any integer from 0 up to 200 and i is from 0 up to 4) if i=0 it will give ranking, if i=1 it will give male-name, if i=2 it will give male-number, if i=3 it will give female-name and if i=4 it will give female-number. In other words, after the ranking number, first two elements are respectively male-name and male-number, last two elements are respectively female-name and female-number. (If i%5 (I mod 5) == 1 then this i is index of male-name, if i%5 == 3 then i is index of female-name).

```
        while(choice.equals("y")) {
            System.out.println("Enter a gender: ");
            String gender = input.next();
            System.out.println("Enter a name: ");
            String name = input.next();

            if (gender.equalsIgnoreCase( anotherString: "male")) {
                getInfo(maleList, name);
            } else if (gender.equalsIgnoreCase( anotherString: "female")) {
                getInfo(femaleList, name);
            }
            System.out.println("Do you want to search for a name and see its statistics (y/n)?");
            choice = input.next();
        }
    }
```

Figure 2.4

As you can see at the code snippet above, we are asking to user a gender and a name to search.

getInfo() Function

```
public static void getInfo(ArrayList<PopularName> list, String name) {
    PopularName babyObject = null;
    for (PopularName baby : list) {
        if (baby.getName().equalsIgnoreCase(name)) {
            babyObject = baby;
        }
    }
    if(babyObject == null) {
        System.out.println("This name doesn't exists in this file.");
    }else {
        System.out.printf(
                "---------------------------------\n" +
                        name + ": \n" +
                        "Index in sorted list: " + list.indexOf(babyObject) + "\n" +
                        "Rank in popularity: " + babyObject.getRanking() + "\n" +
                        "Number of babies: " + babyObject.getNumber() + "\n" +
                        "Percentage of babies: " + "%.2f%% \n---------------------------------\n", getPercentage(list,name)
        );
    }
}
```

Figure 2.5

This function takes to parameters namely, list which is a ArrayList of PopularName objects and name which is a String.

Firstly, I must create a null object because if I directly search the given name, if this name doesn't exist in the file, I can't print "this name doesn't exist". It will print "this name doesn't exist" for each object if object name and given name is not same.

getPercentage() Function

```
public static float getPercentage(ArrayList<PopularName> list, String name) {
    float total = 0;
    float babyNumber = 0;
    for(PopularName baby : list) {
        total += Float.parseFloat(baby.getNumber());
        if(name.equalsIgnoreCase(baby.getName())) {
            babyNumber  = Float.parseFloat(baby.getNumber());
        }
    }
    return (babyNumber)*100/total;
}
```

Figure 2.6

This function takes two parameters namely, list which is a ArrayList of PopularName objects and a name which is string.

All it does is simply, calculate the total number of babies in this gender and calculate the percentage of given baby name.

sortAlphabetically() Function

```
public static void sortAlphabetically(ArrayList<PopularName> list) {
    //Bubblesort algorithm.
    for (int i = 0; i < list.size(); i++) {
        for (int j = 0; j < list.size()-1; j++) {
            if (list.get(j).getName().compareTo(list.get(j+1).getName()) > 0) {
                PopularName temp = list.get(j);
                list.set(j, list.get(j+1));
                list.set(j+1, temp);
            }
        }
    }
}
```

Figure 2.7

As you can see from the Figure 2.7, this method is sorting an array list alphabetically. I used bubble sort algorithm here. I want to explain how compareTo() to works here. The compareTo() method compares two strings lexicographically and the comparison based on the Unicode of each character in the strings. If first element's name comes before the second's elements name (lexicographically) then it will return positive number, if they are equal, it will return 0, if second comes before first then it will return negative number.