Opgave 4: Gebruik van de List operator

Een listiterator wordt gebruikt om alle elementen van een lijst te overlopen. Op deze manier kunnen we elk element van de lijst bekijken zonder zelf iets aan de lijst aan te passen.

Het programma

Het programma begint met het aanmaken van de klasse persoon (Person). Deze klasse persoon bevat de velden voornaam (name), achternaam (surname) en woonplaats (place).

```
package com.company;

public class Person
{
    private String name;
    private String surname;
    private String place;

    public Person(String name, String surname, String place)
    {
        this.name = name;
        this.surname = surname;
        this.place = place;
    }
```

Elk veld krijgt ook zijn eigen getter en setter method, dit zal later nog van pas komen.

De tweede klasse PersonData maakt een ArrayList van het type Person. Deze ArrayList krijgt de naam data.

```
public class PersonData
{
    private ArrayList<Person> data;

    public  PersonData()
    {
        this.data = new ArrayList<>();
    }
```

De eerste method die wordt geschreven is voor het inlezen van het tekstdocument.

```
public ArrayList<Person> Readfile(String pathname) throws IOException
{
    for (String line: Files.readAllLines(Path.of(pathname)))
    {
        String[] dataString = line.split(" ");

        Person tempPerson = new Person(dataString[0], dataString[1],
dataString[2]);

        data.add(tempPerson);
    }
    return data;
}
```

De for lus gaat elke lijn van de file lezen. Op de volgende regel gaat het programma kijken naar spatie karakter en splits de String op en zet elk deel in een array. Daarna wordt een nieuw personage aangemaakt met als velden het eerste deel uit de string met daarin de achternaam. Het tweede deel met daarin de voornaam, en ook het derde met de geboorteplaats. Deze tijdelijke persoon die is aangemaakt wordt dan met het commando add aan de ArrayList data toegevoegd.

Het eerste deel van de opdrachten wat het afprinten van alle namen uit de ArrayList. Om dit te doen schreef ik het programma PrintList.

```
public void printList()
{
    ListIterator<Person> li =this.data.listIterator();
    int i=0;
    System.out.println("Alle Personen:");

    while(li.hasNext())
    {
        Person tempPerson=li.next();
        System.out.println(i+" "+tempPerson.getSurname()+ "
"+tempPerson.getName()+" "+tempPerson.getPlace());
        i++;
    }
}
```

In deze method komt de Listiterator voor. Ze wordt in het begin aangemaakt en bezit het type Person. De naam van de iterator is li en deze wordt gebruikt om da ArrayList data te itereren.

De while lus gaat kijken of er nog een element is in de array met de method hasNext( ). In de lus wordt een tijdelijk persoon van het type Person aangemaakt en gaat naar het volgende element met .next( ). Daarna worden alle gegevens van deze persoon afgedrukt. Om aan de

gegevens te geraken gebruiken we de setters. De i zorgt voor de nummering van de personen bij het afrukken. Als resultaat krijgen we dit:

```
Alle Personen:
0 Konings Albert Laken
1 Tura Will Ieper
2 Merckx Eddy Meise
3 Jansen Jan Antwerpen
4 Peeters Pol Antwerpen
5 Pieters Anna Brussel
6 Smet Albert Antwerpen
7 Merckx Marleen Vilvoorde
```

Het volgende deel van de opdracht was enkel het afrukken van de personen met voornaam Albert. Het programma verloopt vrij gelijkaardig als het vorig.

```
public void printName(String firstName)
{
    ListIterator<Person>li = this.data.listIterator();
    System.out.println("\nDe personen met voornaam Albert:");

    while(li.hasNext())
    {
        Person tempPerson=li.next();
        if (tempPerson.getName().equals(firstName))
            System.out.println(tempPerson.getSurname()+ " "+tempPerson.getName()+"
"+tempPerson.getPlace());

    }
}
```

Het enige verschil is dat we enkel gaan afrukken als er voldaan wordt aal de if statement. Hierin wordt vergeleken of de naam gelijk is aan firtName (bij ons is dit Albert).

Het derde deel van de opdracht was het zoeken naar achternamen die begonnen met een M en deze in tegengestelde volgorde afdrukken.

```
public void printSurnameStartWith(String firstLetter)
{
    ListIterator<Person> li = this.data.listIterator(data.size());
    System.out.println("\nPersonen met familienaam M*");
    while(li.hasPrevious())
    {
        Person tempPerson=li.previous();
        if (tempPerson.getSurname().substring(0,1).equals(firstLetter))
            System.out.println(tempPerson.getSurname()+ " "+tempPerson.getName()+"
"+tempPerson.getPlace());

    }
}
```

Bij het creëren van de ListIterator wordt de plaats meegegeven waar hij moet beginnen. Dit is achteraan de ArrayList deze keer. In plaats van hasNext( ) te gebruiken in de while statement wordt hasPervious( ) gebruikt. De tijdelijk persoon wordt ook de method .previous toegepast. Om na te gaan of de eerste letter een M is wordt er eerst de achternaam genomen van de persoon met getSurname en daarvan wordt er een substring gemaakt die gaat van karakter 0 tot 1. Enkel De eerste letter dus. Deze wordt vergeleken met de op voorhand doorgegeven letter (firstLetter). Is dit het geval dan worden de gegevens weer afgedrukt.

Het laatste deel van de opdrachten gaan we zoeken naar de geboorteplaats van de personen. Als we een match hebben wordt de plaats van de persoon in array in een andere ArrayList (placeInList) geplaats. Daarna overlopen we ArrayList met integers en drukken we de gegevens af van de bijhorende persoon met getters.

```
public void PrintAddress(String birthPlace)
{
    ArrayList<Integer> placeInList = new ArrayList<>();
    ListIterator<Person> li = this.data.listIterator();


    int i =0;
    while (li.hasNext())
    {
        Person tempPerson=li.next();
        if(tempPerson.getPlace().equals(birthPlace))
            placeInList.add(i);
        i++;
    }

    Iterator<Integer> it = placeInList.iterator();

    while(it.hasNext())
    {
        int tempInt =it.next();
        System.out.println(tempInt+" "+ data.get(tempInt).getSurname()+"
"+data.get(tempInt).getName()+" "+data.get(tempInt).getPlace());
    }

}
```

In de eerste while lus kijken we na of de geboorteplaatsen hetzelfde zijn, als dit het geval is wordt de plaats van de persoon in de ArrayList (is i) opgeslagen in placeInList. Wanneer we iedereen overlopen hebben gaan we deze personen afdrukken. We creëren een iterator van het type Integer die over de ArrayList placeInList gaat itereren. We drukken de persoon af door de juiste persoon uit data te halen. Dit doen we met get( ) met hierin de index in de ArrayList. Dit wordt gevolgd door de getters die we zelf hebben aangemaakt.

In de main van het programma roepen we alles aan en catchen we ook wat we hebben getrowed bij het inlezen van het tekstbestand.

```
public class Main {

    public static void main(String[] args)
    {
        programTest();
    }

    public static void programTest()
    {
        try {
            PersonData testData = new PersonData();
            testData.Readfile("D:\\Documenten\\UAsem3\\Java\\3-
Datastructures\\Sessie4\\ListData.txt");
            testData.printList();
            testData.printName("Albert");
            testData.printSurnameStartWith("M");
            testData.PrintAddress("Antwerpen");


        } catch (IOException e)
        {
            System.out.println(e.getMessage());
        }
    }
}
```