Object Oriented Programming
Programming report
Assignment 2: People

Corradini Matteo Berke Atac
S3051390 S3075168

April 26, 2016

1 Problem description

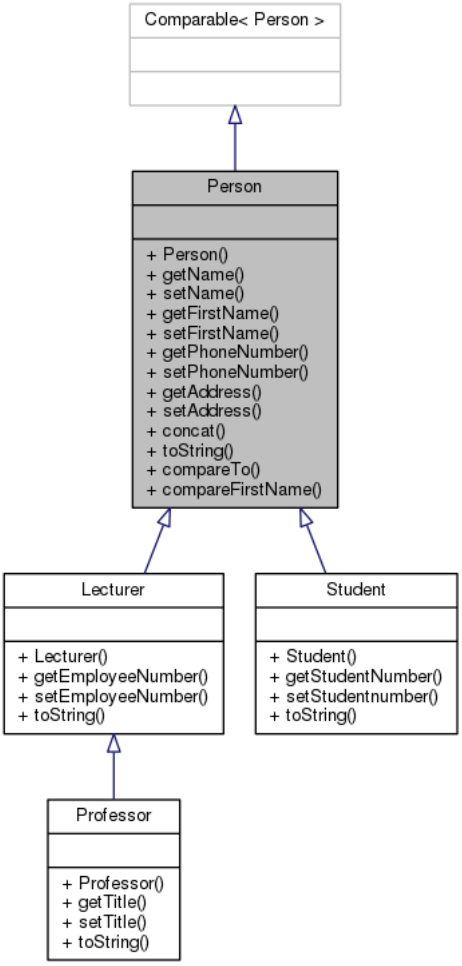
The problem was to compare strings by implementing interface methods and inherit new classes from an existing one.

In the given data structure there were different type of objects representing people, that extended the same abstract class. The information of the objects are given in String format, thus the main problem was to manage the String type objects. We had to implement two different sort methods, and also create and override String method. Furthermore we had to create a new class, and find out the logically correct class to apply the principle of inheritance.

2 Problem analysis

In the beginning we had the following Java source-files:
`Main.java`, `Person.java`, `Student.java`, `Lecturer.java`.

The classes `Student` and `Lecturer` extend the class `Person`, while the class `Main.java` contains the main program. We analyzed the code, we compiled it and we executed it. `Main.java` contains the prime data structure, an ArrayList of `Person` where the objects instantiate `Student` and `Lecturer` class. One of the problem was sort this data structure, first sorted by name and second sorted by first name, implementing two different interfaces (`Comparable` and `Comparator`). Another problem was to create the `Professor` class, which had to be inherit from one of the other class.



3 Program design

In this program we encountered 2 different principles of Object Oriented Programming:

- Inheritance
  - We choose to insert the `phoneNumber` field in the `Person` class, because every type of person has a telephone number, and this class is implements the general information of a person.
  - We choose to extend `Lecturer` class in order to create the `Professor` class. Because a professor is still a lecturer, although they have an extra field for their own title.
- Polymorphism
  - In order to print the information related to a person, we had to reimplement the abstract method because each class has different fields. We used `@Override` clause to reimplement the `String toString()` method. In this way we can use the same method, but each class gives the specific information.

In order to sort the data structure, we implement two methods (in the class `Person` from two different interfaces:

- We sorted by name using `Collections.sort(persons)`, implementing by `int compareTo(Person other)` from `Comparable` interface;

- We sorted by first name using using `Collections.sort(persons, Person.compareFirstName())`, `int compare(Person person1, Person person2)` from `Comparator` interface;

4 Evaluation of the program

Th output of our program:

Listing 1: `Output`

```
Name : Smedinga, First name: Rein, Phone Number: 0123456789, Employee Number: 2345
Name : Styles, First name: Oliver, Phone Number: 1111111111, Student Number: 1231231
Name : Horan, First name: Harry, Phone Number: 2222222222, Student Number: 4564564
Name : Doe, First name: John, Phone Number: 333333333, Employee Number: 6789
Name : Payne, First name: Jack, Phone Number: 444444444, Student Number: 7897897
Name : White, First name: Sow, Phone Number: 555555555, Employee Number: 0123
Name : Malik, First name: Charlie, Phone Number: 6666666666, Student Number: 1011121
Title: Dr., Name : Corradini, First name: Matteo, Phone Number: 1011121, Employee Number: 777777777
Title: Dr., Name : Atac, First name: Berke, Phone Number: 1011121, Employee Number: 888888888
Title: Dr., Name : Palmieri, First name: Viola, Phone Number: 1011121, Employee Number: 9999999999

after sorting:

Title: Dr., Name : Atac, First name: Berke, Phone Number: 1011121, Employee Number: 888888888
Title: Dr., Name : Corradini, First name: Matteo, Phone Number: 1011121, Employee Number: 777777777
Name : Doe, First name: John, Phone Number: 333333333, Employee Number: 6789
Name : Horan, First name: Harry, Phone Number: 2222222222, Student Number: 4564564
Name : Malik, First name: Charlie, Phone Number: 6666666666, Student Number: 1011121
Title: Dr., Name : Palmieri, First name: Viola, Phone Number: 1011121, Employee Number: 9999999999
Name : Payne, First name: Jack, Phone Number: 444444444, Student Number: 7897897
Name : Smedinga, First name: Rein, Phone Number: 0123456789, Employee Number: 2345
Name : Styles, First name: Oliver, Phone Number: 1111111111, Student Number: 1231231
Name : White, First name: Sow, Phone Number: 555555555, Employee Number: 0123

after sorting by first name:

Title: Dr., Name : Atac, First name: Berke, Phone Number: 1011121, Employee Number: 888888888
Name : Malik, First name: Charlie, Phone Number: 6666666666, Student Number: 1011121
Name : Horan, First name: Harry, Phone Number: 2222222222, Student Number: 4564564
```

```
Name : Payne, First name: Jack, Phone Number: 444444444, Student Number: 7897897
Name : Doe, First name: John, Phone Number: 333333333, Employee Number: 6789
Title: Dr., Name : Corradini, First name: Matteo, Phone Number: 1011121, Employee Number: 777777777
Name : Styles, First name: Oliver, Phone Number: 1111111111, Student Number: 1231231
Name : Smedinga, First name: Rein, Phone Number: 0123456789, Employee Number: 2345
Name : White, First name: Sow, Phone Number: 555555555, Employee Number: 0123
Title: Dr., Name : Palmieri, First name: Viola, Phone Number: 1011121, Employee Number: 9999999999
```

5 Conclusions

Our program solves the assignment requests. It helped us to understand:

- How to inherit from abstract class, reusing the fields, the methods and the constructor or implementing new ones;
- How to manage the String object and the use of `varargs`;
- How to implement methods from interfaces: in this case how to sort an `ArrayList` using `Comparable` and `Comparator`;

6 Appendix: program text

Listing 2: `Main`

```
import java.util.Collections;
import java.util.Comparator;
import java.util.ArrayList;

public class Main {

    public static void main(String[] argv) {

        ArrayList<Person> persons = new ArrayList<Person>();
        persons.add(new Lecturer("Smedinga", "Rein", "Broadway␣32", "2345", "0123456789"));
        persons.add(new Student("Styles", "Oliver", "George␣St␣5", "1231231", "1111111111"));
        persons.add(new Student("Horan", "Harry", "Regent␣St␣11", "4564564", "2222222222"));
        persons.add(new Lecturer("Doe", "John", "Main␣St␣153", "6789", "333333333"));
        persons.add(new Student("Payne", "Jack", "Seven␣Bridges␣Way␣3", "7897897", "444444444"));
        persons.add(new Lecturer("White", "Sow", "Fairy␣Ln␣1", "0123", "555555555"));
        persons.add(new Student("Malik", "Charlie", "York␣Rd␣27", "1011121", "6666666666"));

        /* Add new Professor*/
        persons.add(new Professor("Corradini", "Matteo", "York␣Rd␣27", "1011121", "777777777", "Dr."));
```

```
20          persons.add(new Professor("Atac", "Berke", "York␣Rd␣27", "1011121", "
                888888888", "Dr."));
21          persons.add(new Professor("Palmieri", "Viola", "York␣Rd␣27", "1011121"
                , "9999999999", "Dr."));
22
23          for (Person person : persons) {
24              System.out.println(person);
25          }
26
27          /* Sort by name field*/
28          println("\nafter␣sorting␣by␣name:\n");
29          Collections.sort(persons);
30          for (Person person : persons) {
31              System.out.println(person);
32          }
33
34          /* Sort by first name field*/
35          Collections.sort(persons, Person.compareFirstName());
36          System.out.println("\nafter␣sorting␣by␣first␣name:\n");
37          for (Person person : persons) {
38              System.out.println(person);
39          }
40
41      }
42  }
```

Listing 3: `Person`

```
1   import java.util.*;
2
3   public abstract class Person implements Comparable<Person> {
4
5       private String name;
6       private String firstName;
7       private String address;
8       private String phoneNumber;
9
10      public Person(String name, String firstName, String address, String
            phoneNumber) {
11          this.name = name;
12          this.firstName = firstName;
13          this.address = address;
14          this.phoneNumber = phoneNumber;
15      }
16
17      public String getName() {
18          return name;
19      }
20
21      public void setName(String name) {
22          this.name = name;
23      }
24
25      public String getFirstName() {
26          return firstName;
27      }
28
29      public void setFirstName(String firstName) {
30          this.firstName = firstName;
```

```
31      }
32
33
34      public String getPhoneNumber() {
35          return phoneNumber;
36      }
37
38      public void setPhoneNumber(String phoneNumber) {
39          this.phoneNumber = phoneNumber;
40      }
41
42      public String getAddress() {
43          return address;
44      }
45
46      public void setAddress(String address) {
47          this.address = address;
48      }
49
50      /* Return a String as concatenations of the input ones */
51      public String concat(String... stringToConcat){
52          String buffer = new String("");
53          for (String string : stringToConcat)
54              buffer += string;
55          return buffer;
56      }
57
58      @Override
59      public String toString(){
60          return concat("Name␣:␣", this.getName(), ",", "␣First␣name:␣", this.
                getFirstName(),",",
61                  "␣Phone␣Number:␣", this.getPhoneNumber());
62      }
63
64      /* Implement sort by name */
65      public int compareTo(Person other){
66          return this.getName().compareTo(other.getName());
67      }
68
69      /* Implement sort by first name */
70      public static Comparator<Person> compareFirstName(){
71          return new Comparator<Person>(){
72              public int compare(Person person1, Person person2){
73                  return person1.getFirstName().compareTo(person2.getFirstName()
                        );
74              }
75          };
76      }
77  }
```

Listing 4: `Student`

```
1
2   public class Student extends Person {
3
4       private String studentNumber;
5
6       public Student(String name, String firstName, String address,
7               String studentNumber, String phoneNumber) {
```

```
8           super(name, firstName, address, phoneNumber);
9           this.studentNumber = studentNumber;
10      }
11
12      public String getStudentNumber() {
13          return studentNumber;
14      }
15
16      public void setStudentnumber(String studentNumber) {
17          this.studentNumber = studentNumber;
18      }
19
20      @Override
21      public String toString(){
22          return concat(super.toString(),",", "␣Student␣Number:␣", this.
                getStudentNumber());
23      }
24
25  }
```

Listing 5: `Lecturer`

```
1
2   public class Lecturer extends Person {
3
4       private String employeeNumber;
5
6       public Lecturer(String name, String firstName, String address,
7               String employeeNumber, String phoneNumber) {
8           super(name, firstName, address, phoneNumber);
9           this.employeeNumber = employeeNumber;
10      }
11
12      public String getEmployeeNumber() {
13          return employeeNumber;
14      }
15
16      public void setEmployeeNumber(String number) {
17          employeeNumber = number;
18      }
19
20      @Override
21      public String toString(){
22          return concat(super.toString(),",", "␣Employee␣Number:␣", this.
                getEmployeeNumber());
23      }
24
25  }
```

Listing 6: `Professor`

```
1
2   public class Professor extends Lecturer {
3
4       private String title;
5
6       public Professor(String name, String firstName, String address, String
            employeeNumber, String phoneNumber, String title) {
7           super(name, firstName, address, phoneNumber, employeeNumber);
```

```
8           this.title = title;
9       }
10
11      public String getTitle() {
12          return title;
13      }
14
15      public void setTitle(String title) {
16          this.title = title;
17      }
18
19      @Override
20      public String toString(){
21          return concat("Title:␣",this.getTitle(),",","␣",super.toString());
22      }
23
24  }
```