CIS016-1 - Principles of Programming 2015/2016
Exercise Week 21

This exercise sheet is related to Week 20 and previous lectures, so please consult the lecture notes when attempting the exercises. In particular, please also download and investigate the example code and think about event-based programming. Everything you need can be found on BREO under "**Guided Learning → Week 21**".

A classical application of *recursion* is to traverse a tree. Here we try to implement a simple ancestry tree with the goal to print the ancestors of a person. To do so we will first create a suitable class to model a person (**Exercise 1**) and his/her parents and then use this in a recursive method to print all ancestors (**Exercise 2**).

Exercise 1: People and their Moms and Dads

1. Create a class **Person** that holds a first name and a last name of a person. First and last name should be given in the constructor. Furthermore, implement access methods **getFirstName()** and **getLastName()** that return the first and last name, respectively.

2. Extend your **Person** class so that it can hold the mother and father of a person. Mother and father are instances of **Person** as well. Create methods to get and set the father and the mother of a person, ie.. **joe.setMother(anne)** would specify that Anne is the mother of Joe (if joe and anne are instances of **Person**).

Exercise 2: Ancestors

1. Use the Person class from **Exercise 1** to develop an orchestrating class that generates a small ancestry database comprising a person, his/her parents and grandparents.

   Hint: you may take our own ancestry as example or make something up. Create a **Person** instance for each parent and grandparent. Use **setMother()** and **setFather()** to set up the ancestry relationship; ie. your grandmother is your mother's mother.

2. Extend your **Person** class with a static and recursive method **printAncestors (Person p)** that takes a **Person** instance **p** as parameter and prints all known ancestors of this person (parents, grandparents etc).

   Example: **printAncestors(joe)** should print Joe's name, the name of his parents and the names of his grandparents, etc. As mentioned **printAncestors(...)** must be recursive, so it needs to invoke itself just as you've seen in the lecture. Make sure that you do not create an infinite loop and that you handle the condition that **p** is null (think about what this means – can we use this to avoid an infinite loop?).

CIS016-1 - Principles of Programming 2015/2016
Exercise Week 21

3. Extend your orchestrating class so that it prints the ancestors of a selected person using **printAncestors(...)**. Please make sure that you print at least parents and grandparents. You can use the example you created in (a).