Algorithms and Computational Thinking
Autumn 2017

Tuesday, 28th November 2017

Understanding Classes, Objects and Methods

Objective : In these exercises, you will get a basic understanding of the concepts of Classes, Objects and Methods. At the end of the two following exercises, you should be able to understand all the advantages of object oriented programming.

Exercise 1

The goal of the first exercise is to create classes containing attributes and methods related to a specific context. Then, you will create objects by instantiating these classes and use their methods to do the exercise.

In this exercise you will have to create an application for a library. To reach this goal, you will implement two classes : Library and Book.

A book has the following attributes : title, author, category, year. And a library contains a list of books with the methods corresponding to the following actions :

— Storing a book,
— Removing a book from the library using its title,
— Printing all the books existing in a the library.

After having created the library and several books (objects of classes Library and Book), you should be able to add or remove books in the library and print all of them in a nice way.

Exercise 2

The first goal of the second exercise is to consolidate your understanding about the creation of a *k-dimensional* tree and the implementation of a nearest neighbor search (if it was not achieved after the exercise of week 8 - 14th,

15th and 16th of November 2017). The second goal of the second exercise is to understand the creation of classes related to a specific context, which is the *k-dimensional* tree.

As previously described in the exercise of week 8, a *k-dimensional* tree is a data structure helping to organize multiple points in a *k-dimensional* space. The explanation of the algorithm to build a *k-d* tree can be found on this following link : `https://en.wikipedia.org/wiki/K-d_tree` (source : wikipedia).

You must do exactly the same exercise that is described in in the exercise of week 8 by using the concept of class for each node of the *k-d* tree. If you did not succeed or finish the exercise of week 8, the solution without using object oriented programming can be found on the website.

As described in the statement of the exercise of week 8, you must implement the creation of a *k-d* tree as well as the nearest neighbor search using *recursion*. This exercise must be done in all programming languages, such as Python, Scala and Swift, in order to see how you can create a class in these three different programming langages.

You must create a class, called *Node*, containing a tuple (x, y) that describes the position of the node as well as the references to the left child node and the right child node, both are Node type. In this specific example, you will use a *class* in Python, a *case class* in Scala and a *struct* in Swift. In Scala and Swift, you can also create a *class* but, in the context of this exercise, it is better to follow our advice because the content of the nodes of the *k-d* tree will not change. You must always consider the following set of points to build the tree : (2,3), (5,4), (9,6), (4,7), (8,1), (7,2).