World of Robots

Master 1 Génie Physiologique et Informatique

Del 3-3: Competencies

Group 14 : Laurène DONG, Marion DOSPITAL, Lauranne LAIR, Thibault NIGGEL, Florian PAILLAS, Ophélie SOCHARD

During this session, we worked on the turtle project and the robot project (V4 and V5). Through this exercise, we learned a comprehensive set of skills covering both object-oriented design and programming. We learned how to effectively use the link between external classes, define instance members, and implement encapsulation by controlling access to class members.

Using relationships between classes is an effective way to streamline code and improve its readability. These relationships not only help connect different parts of a program, but they also open the door to using methods from one class in another, thereby simplifying the way code is organized and structured.

For example, let's take an association between the two classes seen in the turtle project, a Turtle class and a TextualLOGOTurtle class. Each method present in the Turtle class is called in the TextualLOGOTurtle class to be used.

The definition of instance members are properties specific to each object of a class. They represent the unique characteristics of each instance, such as color, position. These members are defined at the class level and determine the state of each object. They can be of various types, such as integers, strings, or other objects. A precise definition is crucial for designing well-structured and functional objects.

Encapsulation controls access to class members, grouping data and methods into an entity. Visibility levels (public and private) define who can access members. Public allows external access and private limits access inside the class. This ensures correct use, protects data and facilitates maintenance, thereby improving the robustness and reusability of code in object-oriented programming. For example, the show method in the turtle class is a private method and it is called a public method.

We have learned how to implement instance members by creating and defining variables and methods that belong to a class. In the example of the turtle, we declared numerous attributes such as private int x and y, private int direction, private boolean trace, and private string color. These declarations have allowed us to acquire the skill of declaring attributes using various types, variables, and syntaxes. We have also used methods to make the turtle move, turn it, and draw a line to see its traces. For example: public void turn(), public void draw(), and public void goForward() to determine and anticipate the different directions the turtle could take.

For the implementation of accessors and mutators, we started by creating various simple setter and getter methods for the position, which includes the coordinates x and y, as well as the name (of the turtle or robot) and the color. Then, we needed to implement more complex modifiers, especially for the turtle's movement, including the trace that summarizes the turtle's path.

The trace following the turtle posed several problems for us because it was not positioned correctly in relation to the turtle. This led us to consider various possibilities and to make the code more complex.

We had to adhere to various standards of Java such as class names, method names, and variables. We had to follow a specific indentation style, and we added comments to make it easier to read the various methods.

Various objectives were given to us. We had to design methods to be called in order to change the turtle's color, see its traces but behind it, move it according to a reference frame, and therefore pay attention to its orientation in its movements, which should not exceed 3. These tasks have allowed us to develop skills in designing methods according to specific instructions.

Finally, protecting class members is also a skill we have acquired. Indeed, in the example of turtles, we had to create a private function to display the turtle and its movements. This function was then called in all the other public methods.