World of Robots

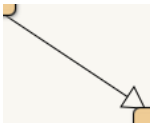
Master 1 Génie Physiologique et Informatique

Schedules

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

In Java, inheritance is a way to compartmentalize and reuse a portion of previously written code in a class (superclass or parent) and create new attributes and methods in a subclass (or child) that shares properties with the superclass. Inheritance allows us, on one hand, to indicate that the subclass inherits from the superclass and, on the other hand, is used to create a hierarchy of classes where a derived class inherits attributes and methods from a base class. In the Turtle project, there is a superclass "Turtle," and three subclasses of turtles with specific characteristics: "FastTurtle," "SmartTurtle," and "ColorTurtle." There are two ways to name an inheritance relationship: specialization when moving from a subclass to the superclass, and generalization when moving from the superclass to the subclass. Abstraction is when we generalize, and refinement is when we specialize.

In the Java language, inheritance is identified by the "extends" keyword (e.g., "public class FastTurtle extends Turtle").



In Unified Modeling Language (UML), inheritance is represented using a solid, continuous arrow pointing from the subclass to the superclass.

There are three ways to achieve inheritance:

1. **Extension**: This involves adding new members (methods or attributes). It extends the capabilities of the superclass, making the subclass more specific. In the example of the Turtle project, it exclusively involves extension because the turtles have the same characteristics as in the superclass but each has additional associated attributes and methods.
2. **Override**: This means modifying a member. You can rewrite or reuse methods from the superclass.
3. **Overload**: This is about duplicating a member by designing a new function with the same name as the superclass but with different parameters.

Inheritance has several benefits, including extension programming to provide new possibilities to a class, specialization for creating more specialized objects, overriding for changing behavior, and code reuse.

Polymorphism is the complex aspect of inheritance. It allows you to create a variable, function, or object that can take on multiple forms. The advantage is that explicit programming is not needed, extension is straightforward, development is faster, and organization is simpler and improved. Polymorphism is well demonstrated in the example of the School project. In this case, the "Professor" and "Student" classes are subclasses of the superclass "Person" because there is an inheritance relationship, but there is also polymorphism. Both are people and are represented by the same attributes and methods, but they serve different roles in the project, and this is what creates polymorphism. In other words, polymorphism is the ability of objects from different classes to respond in a similar manner to the same methods.