INHERITANCE AND POLYMORPHISM

Castaño Silva Laura Daniela

{est.laura.castano@unimilitar.edu.co}

Teacher: Miguel Monroy

Resumen- **Para el proceso del software en lo que ha trascurrido el tiempo se han logrado definir muchos paradigmas en la programación como es el caso de: la herencia. Esta es esencial para la Programación Orientada a Objetivos (POO), debido a que es una extensión, donde se pueden añadir nuevas operaciones o subclases que presentan operaciones para redefinir o, también se puede implementar desde la clase padre sin hacerle algún cambio. Sin embargo, no se pueden realizar herencias múltiples, pero, sí se pueden codificar varias interfaces.**

Palabras claves-. **Algoritmos, superclase, métodos heredados, get y set, constructores, consola, simplificar**

Summary- For the software process in the course of time, many paradigms have been defined in programming, as is the case of inheritance. This is essential for Goal Oriented Programming (OOP), because it is an extension, where new operations or subclasses that present operations to redefine can be added or, it can also be implemented from the parent class without making any changes to it. However, multiple inheritances cannot be performed, but several interfaces can be coded.

Keywords-. **Algorithms, superclass, inherited methods, get and set, constructors, console, simplify.**

I. Objective

To observe how an operation or attribute cannot be suppressed in the inheritance mechanism.

II. THEORETICAL FRAMEWORK

Attributes

Attributes are the properties that objects within a class can assume. They are descriptions of the data. Objects in a class have the same attributes, but their values may differ. Attributes are descriptions of data that are common to the objects of a class. [1].

Methods

A method in Java is a set of instructions defined within a class, which performs a certain task and can be called by a name. When a method is called, the execution of the program passes to the method and when it finishes, the execution continues from the point where the call was made. [2].

Constructor

A constructor is an element of a class whose identifier matches that of the corresponding class and whose purpose is to force and control how an instance of a given class is initialized, since the Java language does not allow the member variables of a new instance to remain uninitialized. In addition, unlike methods, constructors are only used when you want to create a new instance. [3].

Superclass

The superclass would be the class that has all attributes and methods in common. [4].

Subclass

The subclasses will be those that inherit from the first one, with this we establish a system of class hierarchy, where the higher the specification, the lower the levels and there is no limit, we can have a class that inherits from another class that in turn has inherited from another class. [4].

III. PRACTICE DEVELOPMENT

Inheritance is known as a hierarchy. However, it has properties to extend certain functionality that allows to define a new class that inherits characteristics of the parent class or, some existing one. Allowing to save time and lines of code, since the information is extended, for example, it will have attributes and methods that had been defined in a previous class. Therefore, this new one will receive the name of "subclass". While, polymorphism is used to optimize functions based on particular environments or types as is the case of interacting with several objects to give them different virtues and/or behaviors. Polymorphism can be used in an abstract class. Another case, would be when an interface is used or created where the principle of this is obtained. At the same time, there is something called "Encapsulation", where an object must be isolated to create a lock to its behaviors and prevent modifications by means of a control in the access of this information.

Therefore, it is said that Java is a very essential Object Oriented programming language when implementing a code that can be reused at another time. *"This type of language is based on 4 principles that allow describing the relationships between classes: inheritance, encapsulation, abstraction and polymorphism."* [5]. *Where there will be at least one class in each program. As well as, it includes abstractions or syntax as an POO support. Also, it presents reserved words when*

defining interfaces and classes, called "private" in the implementation of a code.

On the other hand, when interacting with a code, a class can be implemented where a code or resource is reused by means of a class, the information of this one, can be extended in another one, inheriting virtues of an old one, also called "parent class". For example, in a code the word "extends" is used at the time of inheritance. Also, *"it is only allowed to inherit from a single class, there is no multiple inheritance. However, it is allowed to implement multiple interfaces".* [6] Interfaces are known as a set of methods existing in a class. As for the attributes of inheritance, it is said that there is an overlapping or disjunct, used to determine whether an object has the ability to be instantiated by more than two subclasses. It can even be determined if all instances coming from the parent class are implemented by child classes. Obtaining as information that: *"there are objects of the parent class that do not belong to any subcategory of those reflected by the child classes (incomplete)".”* [7].

Going deeper into the fact that polymorphism allows two actions to be implemented in the same way when compiling the code, for example, in Java, it is used to manipulate a child class as if it were a parent class, by means of a call to the methods defined in the parent class. In the same way it is determined that: *"it is the technique that allows us that when invoking a certain method of an object, different results can be obtained according to the object's class".* [8]. In other words, the object can have different values during the compilation and execution of the program, known as "parameter overloading". A clear example can be: We can create two distinct classes Cat and Dog, which inherit from the superclass Animal. The Animal class has the abstract makesound() method that is implemented differently in each of the subclasses (cats and dogs sound differently). Then, a third object can send the message to make sound to a group of Cat and Dog objects by means of a reference variable of class Animal, thus making a polymorphic use of those objects with respect to the message move [9].

Finally, object-oriented programming allows an optimization or proper management of the code generated through inheritance, attributes, methods, so that it can be reusable at another time. At the same time, we can show a less complex way to obtain mechanisms that can be understood in a faster and simpler way for both programmers and users.

References

[1] G. Juan, «tecno-simple,» 03 04 2023. [En línea]. Available: https://tecno-simple.com/que-es-un-atributo-en-poo-definicion-y-ejemplos/. [Último acceso: 19 08 2023].

[2] B. Jairo, «Métodos en Java,» 13 12 2015. [En línea]. Available: https://www.cartagena99.com/recursos/alumnos/apuntes/UF%205.1%20-%20Metodos%20en%20Java.pdf. [Último acceso: 19 08 2023].

[3] Arkaitz Garro, «Arkaitz Garro,» [En línea]. Available: https://www.arkaitzgarro.com/java/capitulo-13.html. [Último acceso: 19 08 2023].

[4] olvetic, «olvetic,» 13 07 2014. [En línea]. Available: https://www.solvetic.com/tutoriales/article/942-java-superclases-y-subclases/. [Último acceso: 19 08 2023].

[5] M. Durán, «hubstop,» 19 12 2022. [En línea]. Available: https://blog.hubspot.es/website/polimorfismo-java. [Último acceso: 18 08 2023].

[6] «Blog Bitix,» 31 03 2021. [En línea]. Available: https://picodotdev.github.io/blog-bitix/2021/03/los-conceptos-de-encapsulacion-herencia-polimorfismo-y-composicion-de-la-programacion-orientada-a-objetos/#herencia-e-interfaces. [Último acceso: 18 08 2023].

[7] C. Cachero y P. Ponce de León, «POO-3-Herencia-10-11,» 18 01 2011. [En línea]. Available: https://rua.ua.es/dspace/bitstream/10045/15995/1/POO-3-Herencia-10-11.pdf. [Último acceso: 18 08 2023].

[8] J. Domingo, «Pledin 3.0,» 17 07 2023. [En línea]. Available: https://plataforma.josedomingo.org/pledin/cursos/python3/curso/u53/. [Último acceso: 18 08 2023].

[9] ifgeekthen, «ifgeekthen,» 03 02 2020. [En línea]. Available: https://ifgeekthen.nttdata.com/es/polimorfismo-en-java-programaci%C3%B3n-orientada-objetos. [Último acceso: 19 08 2023].

[10] P. Cianes, «Notacion big O,» 20 10 2020. [En línea]. Available: https://pablocianes.com/notacion-big-o/. [Último acceso: 10 08 2023].

[11] Aprender BIG DATA, «Big-O Para Principiantes,» 03 10 2023. [En línea]. Available: https://aprenderbigdata.com/big-o/. [Último acceso: 10 10 2023].

[12] G. Cortez, E. Diaz, F. Gómez y I. Sandoval, «Ordenamiento Big-O,» 06 11 2019. [En línea]. Available: https://www.studocu.com/latam/document/universidad-tecnologica-de-panama/estructura-de-datos-ii/ordenamiento-big-o-este-es-un-pequeno-trabajo-de-como-funciona-la-notacion-big-o-con-todo-y-sus/5532099. [Último acceso: 11 10 2023].

[13] NetMentor, «Notacion Big-O,» 17 03 2021. [En línea]. Available: https://www.netmentor.es/entrada/notacion-big-o. [Último acceso: 12 10 2023].

[14] J. M. Alarcón, «Rendimiento de algoritmos y notación Big O,» 17 06 2016. [En línea]. Available: https://www.campusmvp.es/recursos/post/Rendimiento-de-algoritmos-y-notacion-Big-O.aspx. [Último acceso: 11 10 2023].