Inheritance and Polymorphism

Sotaquira Vergara Maria Angelica
est.mariaa.sotaquira@unimilitar.edu.co
Teacher: Miguel Monroy

***Summary*— This paper will introduce the concepts of inheritance and object-oriented polymorphism.**

Inheritance allows the reuse of properties and methods of an existing class to create a new class. There are two types of inheritance: single and multiple inheritance. While polymorphism allows objects from different classes to work in a uniform way through a common interface. These concepts are key to create more efficient software.

***Palabras clave*—Inheritance, polymorphism, class, objects, attributes, methods, POO.**

I. Introduction

In object-oriented programming (OOP), the concepts of inheritance and polymorphism are key to create more efficient and adaptable code structures.

Inheritance is an object oriented programming mechanism that allows to derive information from one class to another, that is, we have a parent class and from there two or more child classes are derived. From the child classes we have access to the methods and attributes of the parent class. [1]

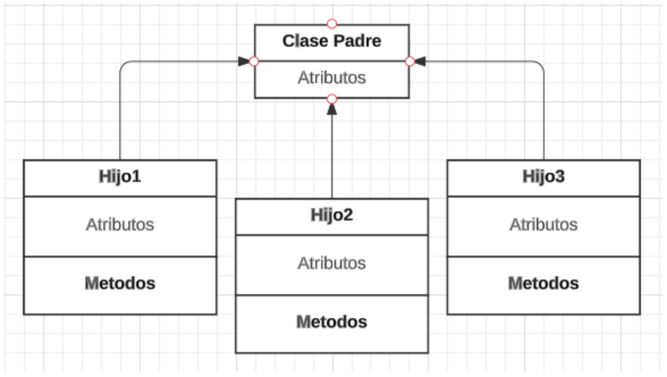


Fig 1. Reference image Inheritances

Polymorphism is the ability of an object to perform an action in different ways using the same methods that are implemented differently in various classes.

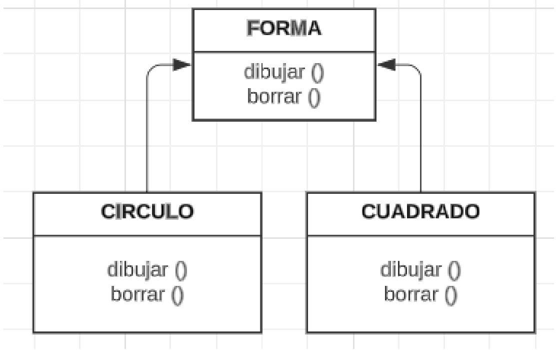


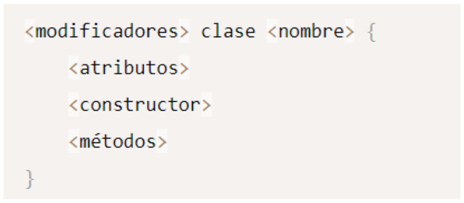
Fig 2. Polymorphism reference image.

Polymorphism allows defining different behaviors for an object depending on the way in which it is implemented and also helps to build concise programs and shorter codes than they could be. [2]

II. Theoretical Framework

A. Class

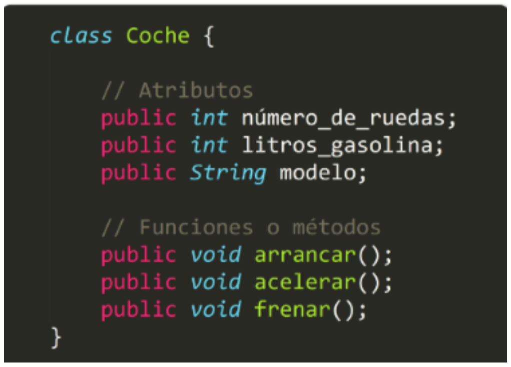
It is a template that will help us to create objects in a predefined way. They are used to represent concepts or entities, basically it is used to represent any noun. Each class has defined a series of attributes and methods and what these methods do is to operate with those attributes, each object created is called an instance of the class. [3]



```
<modificadores> clase <nombre> {
    <atributos>
    <constructor>
    <métodos>
}
```

Fig 3. Reference image structure of a class. [3]

Example:

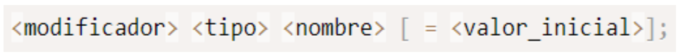


```
class Coche {

    // Atributos
    public int número_de_ruedas;
    public int litros_gasolina;
    public String modelo;

    // Funciones o métodos
    public void arrancar();
    public void acelerar();
    public void frenar();
}
```

Fig 4. Reference image. [4]

B. *Attributes*

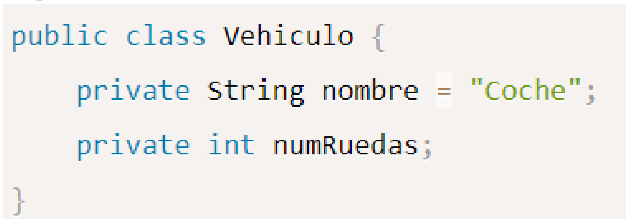
The attributes of a class are those properties that will define the characteristics of a class.



```
<modificador> <tipo> <nombre> [ = <valor_inicial>];
```

Fig 5. Reference image structure attributes. [3]

Example:

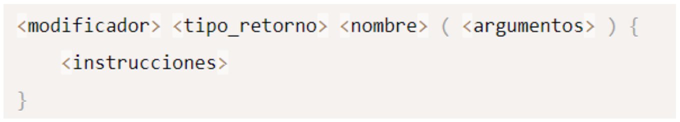


```
public class Vehiculo {
    private String nombre = "Coche";
    private int numRuedas;
}
```

Fig 6. Reference image. [3]

C. *Methods*

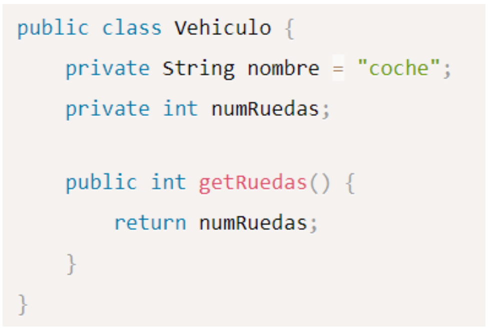
The methods of a class are those functions that will define the behavior or functionality of the objects of the class.



```
<modificador> <tipo_retorno> <nombre> ( <argumentos> ) {
    <instrucciones>
}
```

Fig 7. Reference image structure methods. [3]

Example:



```
public class Vehiculo {
    private String nombre = "coche";
    private int numRuedas;

    public int getRuedas() {
        return numRuedas;
    }
}
```

Fig 8. Reference image. [3]

D. *Superclass*

The class whose characteristics are inherited is known as a superclass (or a base class or a main class).[5]

E. *Subclase*

The class that inherits the other class is known as a subclass (or a derived class, extended class or child class). The subclass can add its own fields and methods, in addition to the fields and methods of the superclass.[5]

F. *POO*

It is a programming model or style that provides guidelines on how to work with it and is based on the concept of classes and objects. It is used to structure a software program in simple and reusable pieces of code planes (classes) to create individual instances of objects.[6]

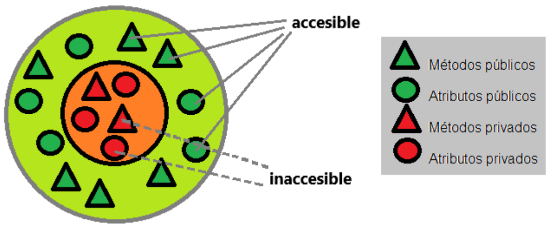


Fig 9. Reference image. [3]

III. DEVELOPMENT

INHERITANCE

Inheritance refers to the succession of property, rights or obligations left by a deceased person to his or her relatives or acquaintances.

This is what we mean in programming when we talk about inheritance " It is the mechanism by which a class inherits the characteristics (attributes and methods) of another class..”[5]

Inheritance relates classes hierarchically; a parent class over child classes. The successors of a class (children) inherit all the attributes and methods that their ancestors (parents) have specified as inheritable, although they can also create their own.

In java only simple inheritance is allowed (it indicates that new classes can be defined only from an initial class [8]) In other words, each class can only be derived from another one, the first one is called superclass and the derived class is called subclass.

In java there are different types of inheritances, which will be mentioned since throughout this course we are going to work with the java language.

1. ***Single inheritance:*** Subclasses can inherit the characteristics of a superclass.

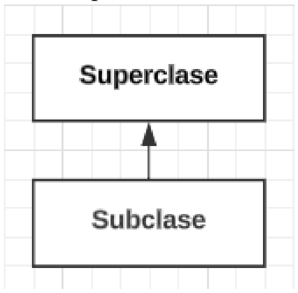


Fig 10. Single inheritance diagram.

2. ***Multilevel Inheritance:*** From a subclass will inherit a superclass and likewise that subclass will be superclass for another subclass.

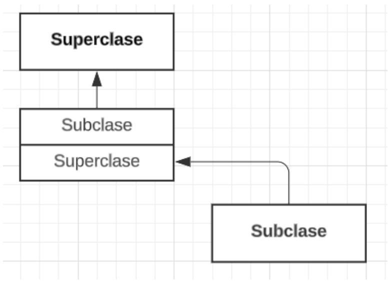


Fig 11. Multilevel inheritance diagram.

3. ***Hierarchical Inheritance:*** A class will be a superclass for other subclasses.

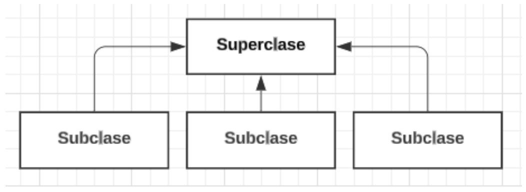
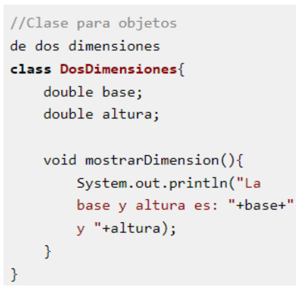


Fig 12. Hierarchical inheritance diagram.

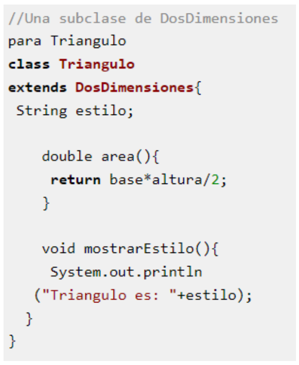
In order to explain in a more didactic way what inheritances are, the following code will be used as a reference to complement the explanation already given.

1. A superclass (parent class) called Dosdimensiones was created with base and height attributes.



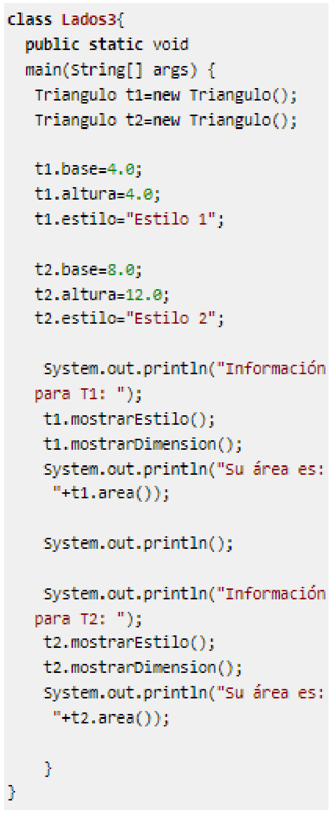
```
//Clase para objetos
de dos dimensiones
class DosDimensiones{
    double base;
    double altura;

    void mostrarDimension(){
        System.out.println("La
        base y altura es: "+base+"
        y "+altura);
    }
}
```

2. We have as a subclass (child class) Triángulo



```
//Una subclase de DosDimensiones
para Triangulo
class Triangulo
extends DosDimensiones{
 String estilo;

    double area(){
     return base*altura/2;
    }

    void mostrarEstilo(){
     System.out.println
   ("Triangulo es: "+estilo);
  }
}
```

3. Another subclass called Lados3 was created which can inherit from the subclass Triángulo.



```
class Lados3{
  public static void
  main(String[] args) {
   Triangulo t1=new Triangulo();
   Triangulo t2=new Triangulo();

   t1.base=4.0;
   t1.altura=4.0;
   t1.estilo="Estilo 1";

   t2.base=8.0;
   t2.altura=12.0;
   t2.estilo="Estilo 2";

    System.out.println("Información
   para T1: ");
    t1.mostrarEstilo();
    t1.mostrarDimension();
    System.out.println("Su área es:
     "+t1.area());

    System.out.println();

    System.out.println("Información
   para T2: ");
    t2.mostrarEstilo();
    t2.mostrarDimension();
    System.out.println("Su área es:
     "+t2.area());

    }
}
```

Fig 13. Reference image example. [5]

From the above program, it can be said that by creating the Triangle object the methods of the superclass are inherited into this object.

This can confirm that by using the objects of a subclass you can also access the members of a superclass.

<u>POLYMORPHISM</u>

Polymorphism is closely linked to inheritance, as it is still the ability of objects of a class to contain values of different types during program execution.

A method can be called in different objects as long as it has the same name and parameters already defined.

Something to keep in mind is that polymorphism must be resolved at runtime, that is why we talk about dynamic polymorphism, which unlike static polymorphism, this is able to be resolved while compiling and is implemented by overloading the methods.

To explain in more detail what polymorphism is, we will use the following reference code:

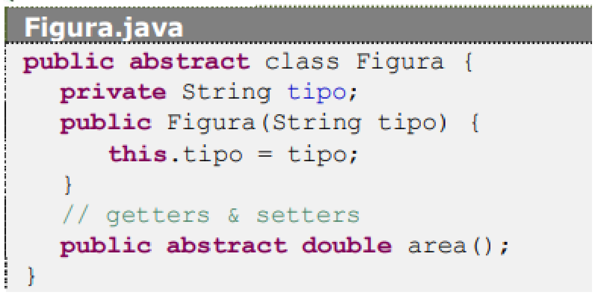
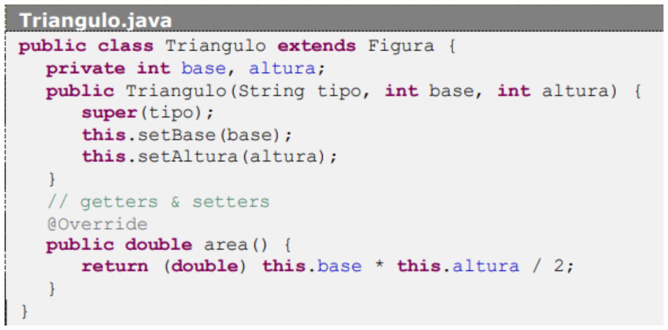


Figura.java

```
public abstract class Figura {
   private String tipo;
   public Figura(String tipo) {
      this.tipo = tipo;
   }
   // getters & setters
   public abstract double area();
}
```

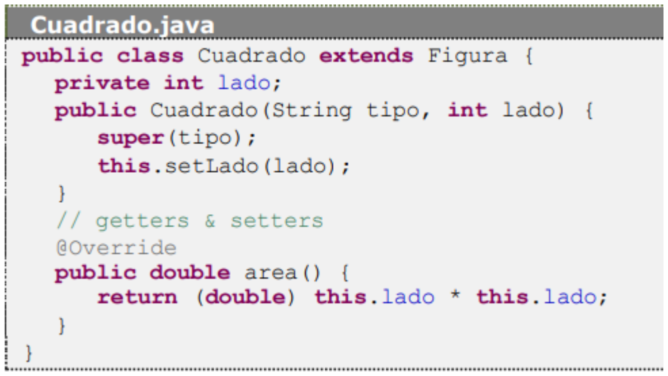
Fig 14. Superclass Figura



Triangulo.java

```
public class Triangulo extends Figura {
   private int base, altura;
   public Triangulo(String tipo, int base, int altura) {
      super(tipo);
      this.setBase(base);
      this.setAltura(altura);
   }
   // getters & setters
   @Override
   public double area() {
      return (double) this.base * this.altura / 2;
   }
}
```

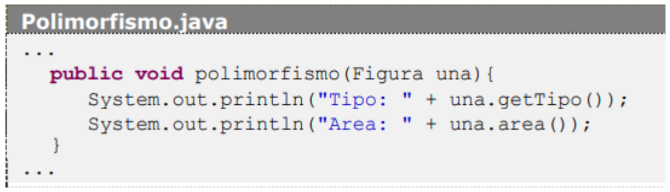
Fig 15. Subclass Triángulo



Cuadrado.java

```
public class Cuadrado extends Figura {
   private int lado;
   public Cuadrado(String tipo, int lado) {
      super(tipo);
      this.setLado(lado);
   }
   // getters & setters
   @Override
   public double area() {
      return (double) this.lado * this.lado;
   }
}
```

Fig 16. Subclass Cuadrado



Polimorfismo.java

```
...
  public void polimorfismo(Figura una){
     System.out.println("Tipo: " + una.getTipo());
     System.out.println("Area: " + una.area());
  }
...
```

Fig 17. Polymorphism.

From the previous code we can observe that from the Superclass **Figure** the **area**() method is inherited by the subclasses **Triangle** and **Square**, seeing here the polymorphism as shown in Fig 17.

IV. Conclusions

- Inheritance and polymorphism is an indispensable concept in POO since it allows designing and building more efficient programs.
- Inheritances allow us to discover more possibilities and combinations that will help us to work in more complex software with more efficiency, finally presenting an orderly and treatable code for the user and programmer.
- Inheritance allows us to establish relationships between classes in order to facilitate the organization and understanding of the code. It also allows subclasses to share attributes and methods of superclasses in order to write shorter and more precise code.
- Polymorphism allows flexibility in the class structure, and allows objects of different classes to respond to the same interface, which facilitates code creation.
- It can be stated that inheritance and polymorphism go hand in hand since both are fundamental concepts in POO that provide several benefits for software design and development.

References

[1] *Herencia en programación orientada a objetos*. (s.f.). https://www.netmentor.es/entrada/herencia-poo.

[2] *¿Qué es el Polimorfismo - POO? - Fredy Geek*. (s.f.). Fredy Geek. https://fredygeek.com/2020/09/05/que-es-el-polimorfismo-poo/

[3] *Â¿QuÃ© es una clase en programaciÃ³n? | Kiko Palomares*. (s.f.). Kiko Palomares. https://kikopalomares.com/clases/que-es-una-clase-en-programacion#:~:text=Las%20clases%20se%20utilizan%20para,sirve%20para%20representar%20cualquier%20sustantivo.

[4] (s.f.). https://uploads-ssl.webflow.com/635a453bdfb49358830950bd/63623ebc66f43b34ca973f66_1.PNG

[5] *¿Qué es la Herencia en programación orientada a objetos?* (s.f.). IfgeekthenNTTdata. https://ifgeekthen.nttdata.com/es/herencia-en-programacion-orientada-objetos

[6] *Qué es la Programación Orientada a Objetos*. (s.f.). Intelequia. https://intelequia.com/blog/post/qué-es-la-programación-orientada-a-objetos

[7] (s.f.). Alura Latam | Cursos online de tecnología. https://www.aluracursos.com/blog/assets/poo-que-es-la-programacion-orientada-a-objetos/poo-que-es-la-programacion-orientada-a-objetos-img1.png

[8] *III - Herencia*. (s.f.). Páginas Personales. http://profesores.fi-b.unam.mx/carlos/java/java_basico3_4.html#:~:text=Hay%20dos%20tipos%20de%20herencia,dos%20o%20más%20clases%20iniciales.