TALENTO DIGITAL
INTELIGENCIA HUMANA

Talento Digital para Chile:

MÓDULO 1
PROGRAMACIÓN BÁSICA EN JAVA

UN PROYECTO DE:

DESARROLLADO POR:

MÓDULO 1 - PROGRAMACIÓN BÁSICA EN JAVA

1.4 HERENCIA Y POLIMORFISMO

Semana 3 - Día 17

Objetivo de la jornada

- Entender y aplicar los conceptos de Polimorfismo y aplicarlos en los ejercicios planteados.
- Representar Polimorfismo en un diagrama de clases según los requerimientos señalados.

Polimorfismo

El Polimorfismo es uno de los 4 pilares de la programación orientada a objetos (POO) junto con la Abstracción, Encapsulación y Herencia. Para entender qué es el polimorfismo es muy importante que tengáis bastante claro el concepto de la Herencia, por tanto recomendamos que veáis la entrada en la que hablamos de la Herencia: Herencia en Java, con ejemplos.

Para empezar con esta entrada, se ha de decir que el término "Polimorfismo" es una palabra de origen griego que significa "muchas formas". Este término se utiliza en la POO para "referirse a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos". Como esta definición quizás sea algo difícil de entender, vamos a explicarla con el ejemplo que pusimos en la entrada de la herencia en la que queríamos simular el comportamiento que tendrían los diferentes integrantes de la selección española de fútbol; tanto los Futbolistas como el cuerpo técnico (Entrenadores, Masajistas, etc...). Para este ejemplo nos vamos a basar en el siguiente diagrama de clases:

Implementando Polimorfismo mediante herencia

Para entender el Polimorfismo vamos a realizar un pequeño ejemplo que consistirá en una clase padre y dos clases hijas. Estas heredan del mismo padre, cada clase sobrescribirá un método del padre para ver cómo se comporta el polimorfismo.

Clase padre

Esta clase tiene un método llamado imprimir(), el cual solo va a imprimir un mensaje en la consola de salida.

```
public class Animal {
    private String especie;
    public Animal(String especie) {
        this.especie = especie;
    }

    public void imprimir() {
        System.out.println("Soy un animal de la especie: " + this.getEspecie());
    }

    public String getEspecie() {
        return especie;
    }

    public void setEspecie(String especie) {
        this.especie = especie;
    }
}
```

Las clases hijas

Cada una tiene sus métodos y atributos, las dos heredan la funcionalidad del padre, también las dos hijas están sobrescribiendo el método imprimir() del padre.

```
public class Perro extends Animal {
    private String nombre;
    public Perro(String especie, String nombre){
        super(especie);
        this.nombre = nombre;
    }

    public void imprimir(){
        super.imprimir();
        System.out.println("Soy un perro que ladra");
    }


        public String getNombre() {
            return nombre;
        }

        public void setNombre(String nombre) {
            this.nombre = nombre;
        }
 }
```

La segunda clase derivada sería la siguiente:

```
public class Gato extends Animal{
    private String nombre;
    public Gato(String especie, String nombre){
        super(especie);
        this.nombre = nombre;
    }

    public void imprimir(){
        super.imprimir();
        System.out.println("Soy un gato que maulla");
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

Ejecución del ejemplo de Polimorfismo en Java

Para probar el polimorfismo vamos a realizar un programa que implementa a las clases que acabamos de crear en las imágenes anteriores y vamos a imprimir con el método imprimir() para ver su comportamiento.

```
public class PolimorfismoMain {
    public static void main(String []args){
        Animal fido, snarf;
        fido = new Perro("Perro", "Fido");
        snarf = new Gato("Gato", "Snarf");

        fido.printMensaje();


            snarf.printMensaje();
        }
    }
```

Como se puede apreciar en la imagen anterior, se están declarando dos variables de la clase Animal. Posteriormente esas variables son instanciadas con el operador “new” y pasan a ser instancias de las clases “Perro” y “Gato” respectivamente.

Después cada objeto imprime su propio mensaje.

Salida: en el resultado podemos observar que se imprime un mensaje que esta declarado en el padre y otro mensaje cuando se sobrescribió el método printMensaje() en cada uno de los hijos.

Con esto podemos comprobar la herencia ya que los hijos están presentando comportamiento propio y también comportamiento de su clase padre.

Soy un animal de la especie: Perro
Soy un perro que ladra
Soy un animal de la especie: Gato
Soy un gato que maulla