



El futuro digital  
es de todos

MinTIC



# Interface en Java



Universidad de Caldas

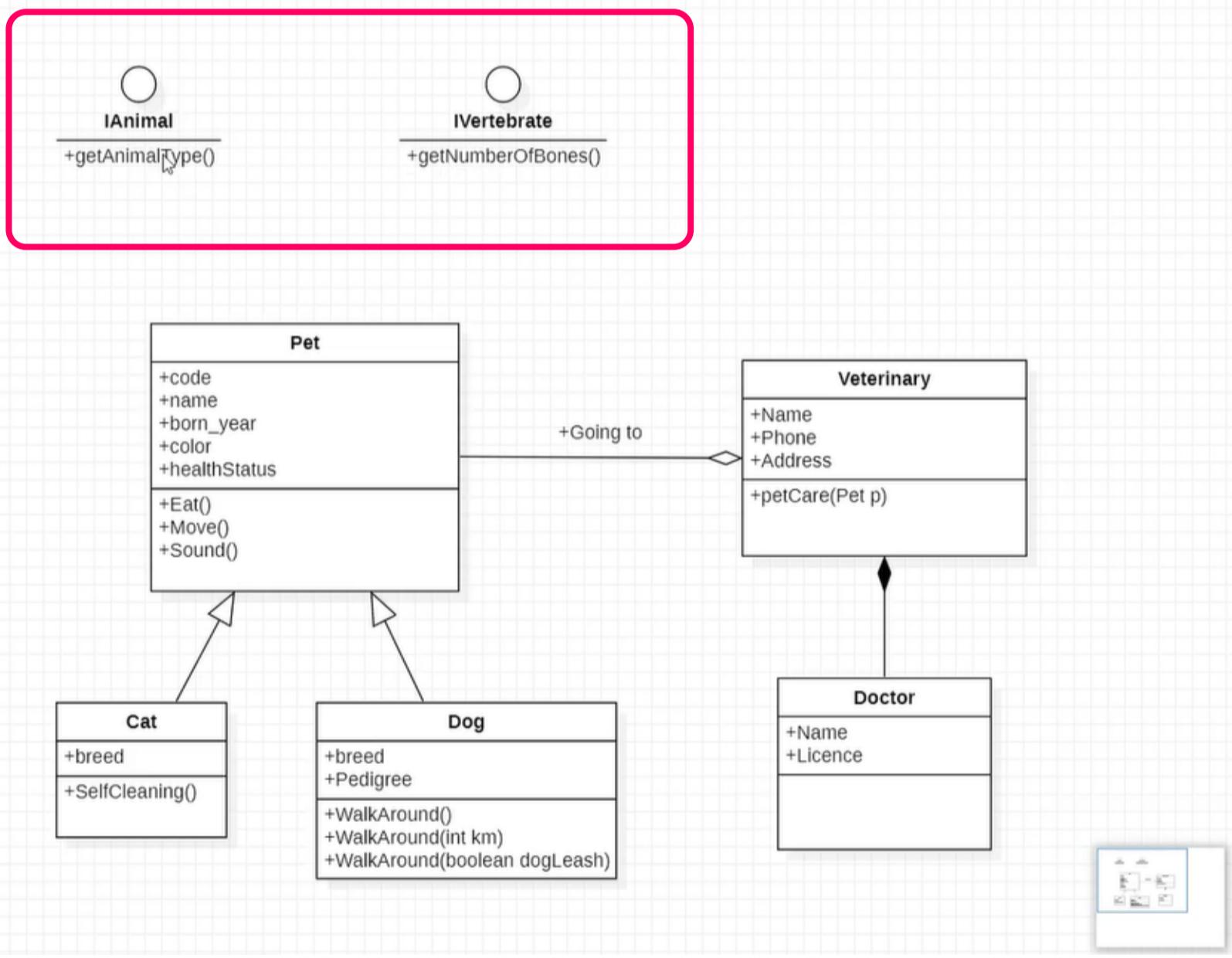
# Hola:

Con todo el conocimiento adquirido hasta el momento podemos profundizar aún más en la programación orientada a objetos. En el desarrollo de un software bastante robusto se utilizan algunas estructuras diferentes a las clases que hemos visto hasta el momento. Aunque la **clase** como tal sigue siendo el núcleo de todo, en este caso queremos explicarles la diferencia entre clase, clase abstracta e interfase. Los tres (3) conceptos son clave cuando necesitamos aplicar buenas prácticas de programación como son los principios SOLID, principios que les recomendamos que revisen en la documentación adicional de la semana para que comprendamos la importancia de estos conceptos.

Una vez conocido el concepto de clase, definimos atributos y métodos. Cómo sabemos los métodos de una clase deben estar implementados, es decir, deben contar con un cuerpo que define su funcionamiento. Por el contrario, las interfaces plantean sólo la firma de los métodos sin llegar a su implementación, es decir, se informa sobre el nombre del método y sus parámetros, pero no sobre su funcionamiento.

¿Entonces dónde está la implementación? Pues desde la misma clase pura que ya conocemos podemos implementar una interfase y allí será donde se agregue el cuerpo de los métodos. Esta es una estrategia que se plantea para solucionar el problema de la herencia, donde una clase solo puede heredar de otra única clase. En este caso, una clase puede implementar N interfaces, estando obligada a implementar los métodos de todas las interfaces. Veamos un ejemplo práctico.

Video de interfaces implementadas por clases. Aquí se puede indicar aún más detallado el concepto de **polimorfismo puro** mediante la interfase.



The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** PetManager - NetBeans IDE 8.2
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard NetBeans toolbar with icons for file operations.
- Project Explorer:** Shows a Java project named "PetManager" with packages "Source Packages" and "Interfaces". Under "Interfaces", there are files "IAnimal.java" and "IVertbrate.java".
- Code Editor:** The "IAnimal.java" file is open. The code is as follows:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Interfaces;

/**
 *
 * @author USUARIO
 */
public interface IAnimal {
```
- Context Menu:** A context menu is open over the code editor, with the "New..." option highlighted.
- Output Panel:** An empty panel showing the output of the build process.

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** PetManager - NetBeans IDE 8.2
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard NetBeans toolbar with icons for file operations.
- Project Explorer:** Shows a Java project named "PetManager" with packages "Source Packages" and "Interfaces". Under "Interfaces", there are files "IAnimal.java" and "IVertbrate.java".
- Code Editor:** The "IVertbrate.java" file is open. The code is as follows:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Interfaces;

/**
 *
 * @author USUARIO
 */
public interface IVertbrate {
```
- Output Panel:** An empty panel showing the output of the build process.

The screenshot shows the NetBeans IDE interface with the project 'PetManager' open. The code editor displays the `clsPet.java` file, which implements the `IAnimal` and `IVertebrate` interfaces. The `getAnimalType()` method is implemented by throwing an `UnsupportedOperationException`. The `getNumberOfBones()` method is also implemented by throwing an `UnsupportedOperationException`. The Navigator panel on the left shows the members of the `clsPet` class.

```

121
122     * @param veterinary the veterinary to set
123     */
124     public void setVeterinary(clsVeterinary veterinary) {
125         this.veterinary = veterinary;
126     }
127
128     @Override
129     public String getAnimalType() {
130         throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
131     }
132
133     @Override I
134     public int getNumberOfBones() {
135         throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
136     }
137 }

```

The screenshot shows the NetBeans IDE interface with the project 'PetManager' open. The code editor displays the `Interface.java` file, which contains a main method that creates instances of `clsDog` and `clsCat` and prints their animal types. The output window shows the results of the execution.

```

8     import Classes.clsCat;
9
10    import Classes.clsDog;
11
12    /**
13     *
14     * @author USUARIO
15     */
16
17    public class Interface {
18        public static void main(String[] args) {
19            // Instancias de las clases hijas
20            clsDog dog = new clsDog("criollo", false, "001", "Firulais", 2013, "Negro", "Sano");
21            clsCat cat = new clsCat("Angora", "002", "Minino", 2013, "Blanco y negro", "Enfermo");
22
23            System.out.println(dog.getAnimalType());
24            System.out.println(" " + cat.getAnimalType());
25
26            System.out.println(dog.getNumberOfBones());
27            System.out.println(" " + cat.getNumberOfBones());
28        }
29    }

```

Output - PetManager (run):

```

fun:
doméstico
Gato
0
230
BUILD SUCCESSFUL (total time: 0 seconds)

```

Desde nuestra experiencia el concepto de interfase es uno de los más utilizados en el desarrollo de software a gran escala, cumpliendo con estándares como los principios *SOLID*. Los invitamos a profundizar en este tema con el apoyo del material adicional de la semana.



**Mision  
TIC2022**

The logo features the text "Mision TIC2022" in a bold, sans-serif font. The word "Mision" is in blue, "TIC" is in red, and "2022" is in blue. A red curved line starts from the top of the letter "i" in "Mision" and ends at the bottom of the letter "c" in "TIC". The background of the logo is a white circle with a gray halftone dot pattern, set against a dark red circular frame.

Universidad de Caldas