



El futuro digital  
es de todos

MinTIC

# Clase Abstracta

+Abst

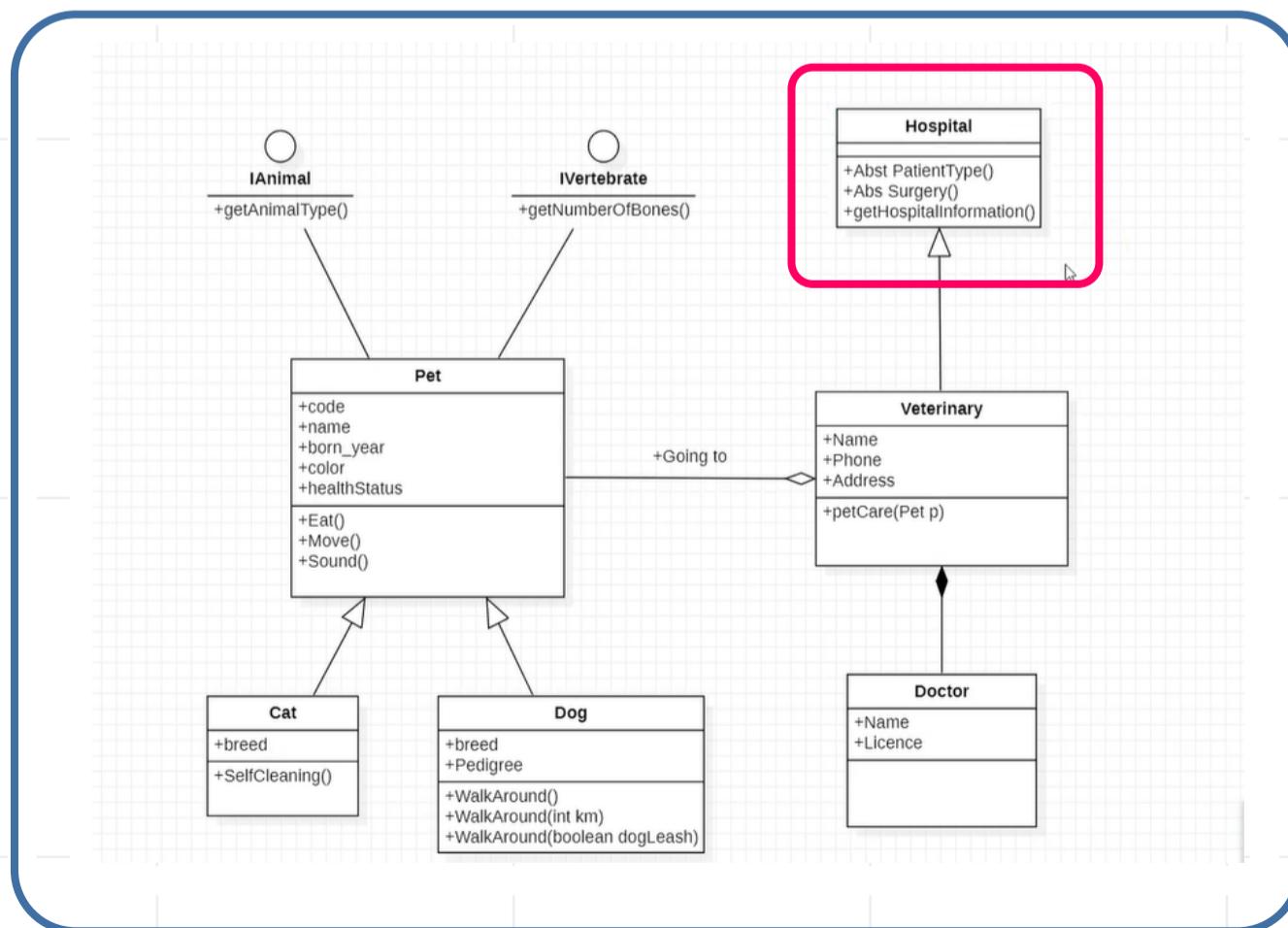


Universidad de Caldas

# Hola:

Ya conocemos los dos (2) tipos de clases, la primera clase donde se debe definir todo y la segunda donde solo se define la firma de los métodos. Es este caso el punto intermedio será la **Clase Abstracta** y que no representan algo específico y se puede usar para crear otras clases. No pueden ser instanciadas, por lo que no se pueden crear nuevos objetos con ellas y se ubican en un intermedio entre clase e interface, esto debido a que la clase abstracta puede definir solo la firma de un método o implementarlo completamente. Las clases abstractas pueden ser heredadas y las clases hijas deberán implementar los métodos que no tengan cuerpo.

Video de clase abstracta implementadas por clases (Clase mascota con método implementado de comer y hacer sonido será abstracto).



The screenshot shows the NetBeans IDE interface with the project "PetManager" open. The code editor displays the `AbstHospital.java` file, which contains the following code:

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package Classes;
7
8 /**
9 *
10 * @author USUARIO
11 */
12 public abstract class AbstHospital {
13     public abstract String getPatientType();
14     public abstract String Surgery();
15 }
16
17 }
```

The Navigator pane on the left shows the members of the `AbstHospital` class, including `Surgery()` and `getPatientType()`. The Output pane at the bottom is empty.

The screenshot shows the NetBeans IDE interface with the project "PetManager" open. The code editor displays the `clsVeterinary.java` file, which contains the following code:

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package Classes;
7
8 /**
9 *
10 * @author USUARIO
11 */
12 public class clsVeterinary extends AbstHospital {
13     private String name;
14     private String phone;
15     private String address;
16     private clsDoctor doctor;
17
18     public clsVeterinary(String name, String phone, String address, clsDoctor doctor) {
19         this.name = name;
20         this.phone = phone;
21     }
22 }
```

The Navigator pane on the left shows the members of the `clsVeterinary` class, including its constructor parameters. The Output pane at the bottom is empty.

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.
- Projects Tab:** Shows the PetManager project structure under JavaIntroduction.
- Source Editor:** Displays the content of AbstractClass.java. The code is as follows:

```
8 import Classes.clsDOCTOR;
9 import Classes.clsVeterinary;
10
11 /**
12 *
13 * @author USUARIO
14 */
15 public class AbstractClass {
16     public static void main(String[] args) {
17
18         // Instancias de veterinaria y de doctor
19         clsDoctor doctor = new clsDoctor("María Fernández", "L12345");
20         clsVeterinary veterinary = new clsVeterinary("Veterinaria UdC", "036 878 15 00", "Calle 65 N 26-10", doctor);
21
22         String data = veterinary.getHospitalInformation();
23         String type= veterinary.getPatientType();
24         String surgery = veterinary.Surgery();
25     }
26 }

```

- Output Tab:** Shows the output of the run command: "Output - PetManager (run)"

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

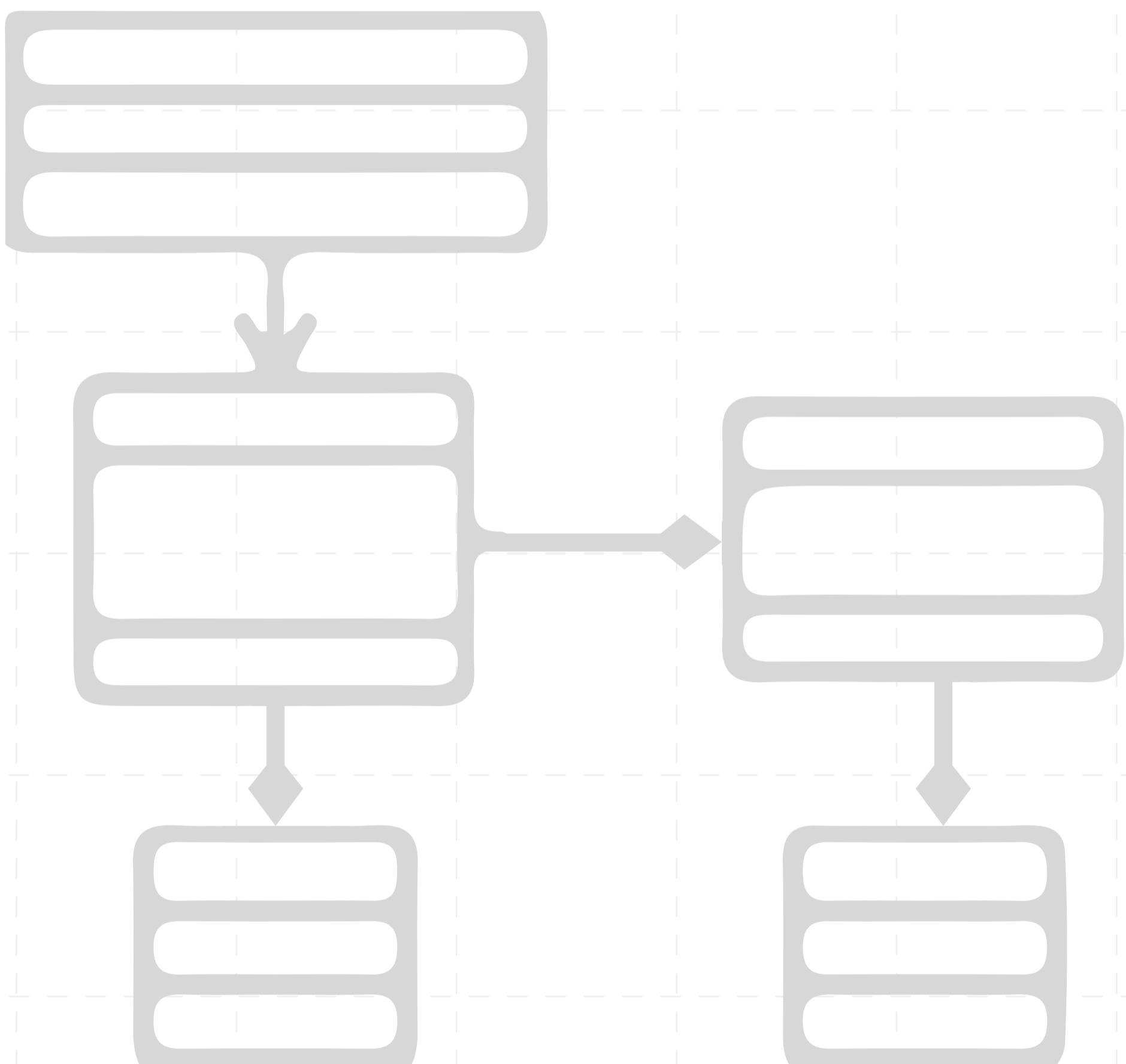
- Title Bar:** PetManager - NetBeans IDE 8.2
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard NetBeans toolbar with icons for file operations.
- Projects Tab:** Shows the PetManager project structure under JavaIntroduction.
- Source Editor:** Displays the content of AbstractClass.java. The code is as follows:

```
14 */
15 public class AbstractClass {
16     public static void main(String[] args) {
17
18         // Instancias de veterinaria y de doctor
19         clsDoctor doctor = new clsDoctor("María Fernández", "L12345");
20         clsVeterinary veterinary = new clsVeterinary("Veterinaria UdC", "036 878 15 00", "Calle 65 N 26-10", doctor);
21
22         veterinary.setData(veterinary.getName() + " - " + veterinary.getAddress());
23
24         String data = veterinary.getHospitalInformation();
25         String type= veterinary.getPatientType();
26         String surgery = veterinary.Surgery();
27
28         System.out.println("Data: " + data);
29         System.out.println("Type: " + type);
30         System.out.println("surgery: " + surgery);
31     }
32 }

```

- Output Tab:** Shows the output of the run command, including the printed data and the build status: "run:  
Data: La información es: Veterinaria UdC - Calle 65 N 26-10  
Type: Animal  
surgery: Surgery animal data  
BUILD SUCCESSFUL (total time: 0 seconds)

Con esta temática finalizamos la introducción a toda esta conceptualización, tanto teórica como práctica en el paradigma de programación orientada a objetos. Los invitamos a que pongan en práctica este nuevo conocimiento mediante el reto de la semana.





**Mision  
TIC2022**

The logo features the text "Mision TIC2022" in a bold, sans-serif font. The letters are primarily blue, except for the letter "i" which is red. A red curved line starts from the top of the first "i" and ends at the bottom of the second "i". The background of the logo is a white circle with a gray halftone dot pattern. The entire logo is set against a dark red circular background.

Universidad de Caldas