APLICAR PROGRAMACIÓN ORIENTADA A OBJETO CON C#

José Fernando Galindo Suárez

APLICAR PROGRAMACIÓN ORIENTADA A OBJETO CON C#

3

CLASES ABSTRACTAS CON C#

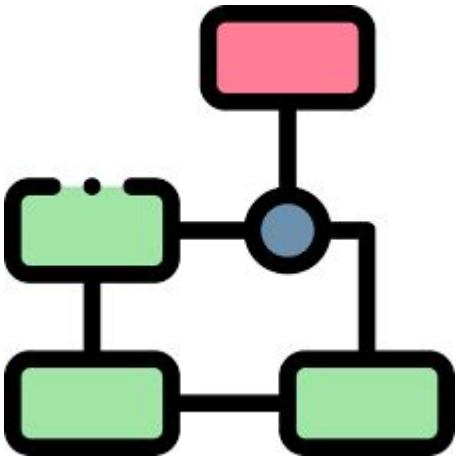
OBJETIVOS

Después de completar esta lección usted estará en la capacidad de:

- **Identificar las clases abstractas.**
- **implementar Clases abstractas.**

Ver Pagina 81.

CLASES ABSTRACTAS



Las clases abstractas son aquellas que por sí mismas no se pueden identificar con algo 'concreto' (no existen como tal en el mundo real), pero sí poseen determinadas características que son comunes en otras clases que pueden ser creadas a partir de ellas.

CLASES ABSTRACTAS

Para declarar una **clase** o método como abstractos, se utiliza la palabra reservada abstract. Una **clase abstracta** no se puede instanciar(es decir no se pueden volver en objetos), pero si se puede heredar y las **clases** hijas serán las encargadas de implementar la funcionalidad a los métodos abstractos.

CLASES ABSTRACTAS

patas

tipo

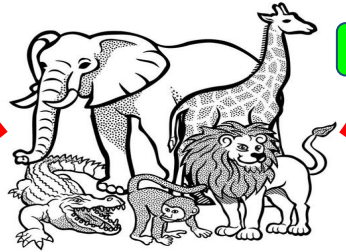
piel

getmispatas():int

M

Piel():void

A



ANIMAL<<abstract>>

:

:

Pez(String Tipo,String Piel,int Patas)

override Piel():void

getmispatas():int

PEZ

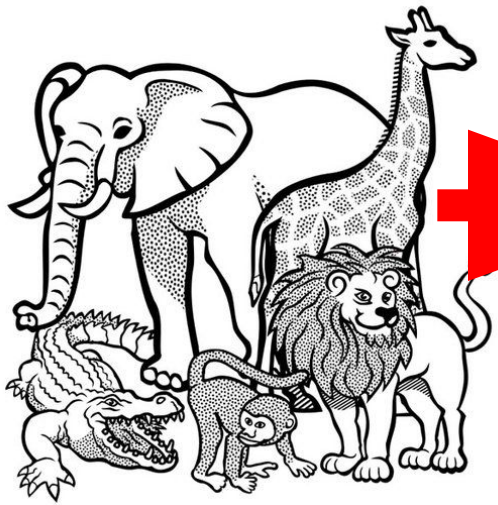
Aves(String Tipo,int Patas)

override Piel():void

getmispatas():int

AVES

CLASES ABSTRACTAS



ANIMAL

```
public abstract class Animal{
    public int patas;
    public String  tipo;
    public String piel;
    public int getmispatas(){return patas;}
    public abstract void Piel();
}
```

1

2

3

CLASES ABSTRACTAS

ANIMAL

PEZ

```
public class Pez:Animal{
    public Pez(String Tipo,String Piel,int Patas){ 1
        tipo=Tipo;piel=Piel;patas=Patas;
        Console.WriteLine("Es un {0}, tienen {1} patas y tiene {2}",tipo,patas,piel);
    }
    public override void Piel(){ 2
        Console.WriteLine("Tienen {0}\n",piel);
    }
    public int getmispatas(){return patas;} 3
}
```

CLASES ABSTRACTAS

ANIMAL

AVES

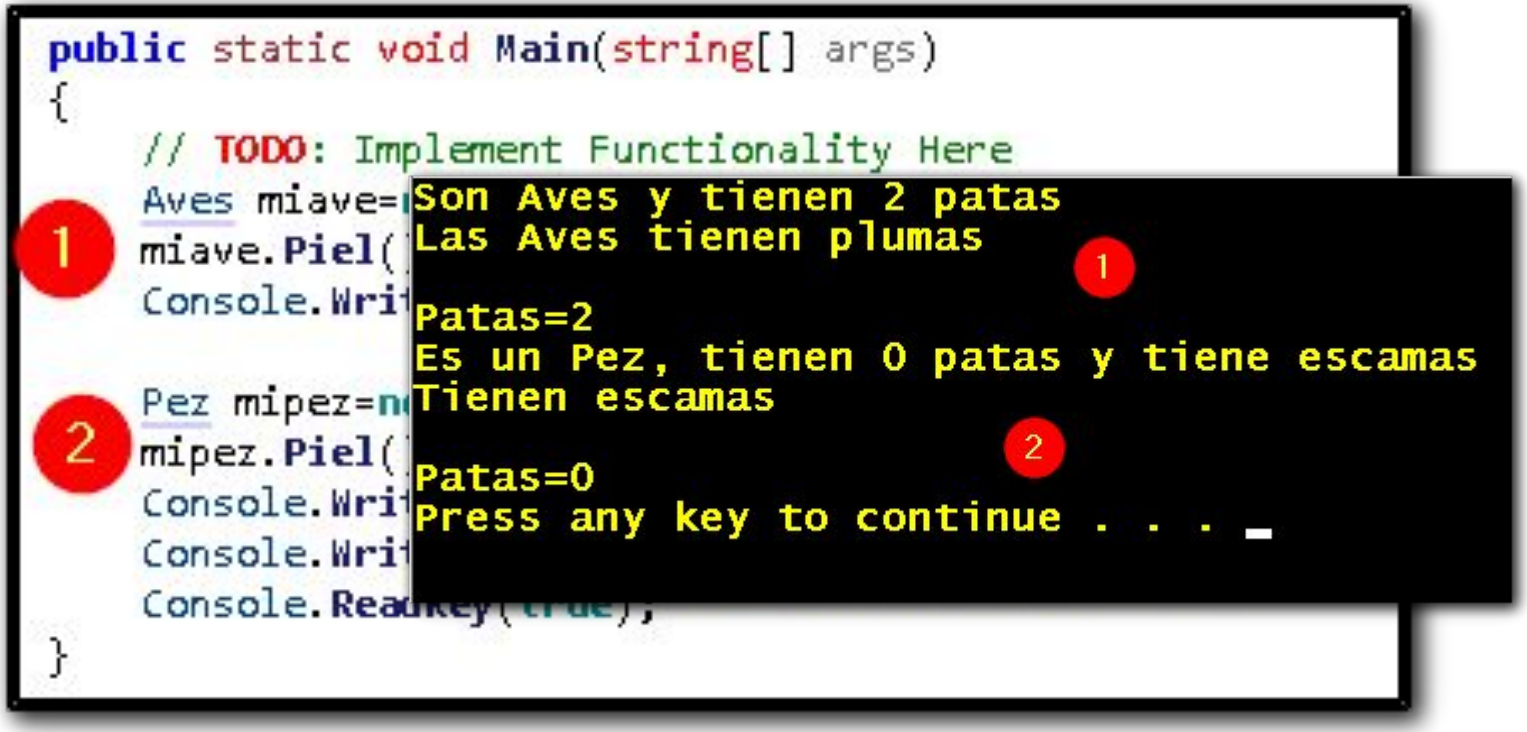
```
public class Aves:Animal{
    public Aves(String Tipo,int Patas){ 1
        patas=Patas;
        tipo=Tipo;
        Console.WriteLine("Son {0} y tienen {1} patas ",tipo,patas);
    }
2   public override void Piel(){
        Console.WriteLine("Las {0} tienen plumas\n",tipo);
    }
    public int getmispatas(){return patas;} 3
}
```

CLASES ABSTRACTAS

```
public static void Main(string[] args)
{
    // TODO: Implement Functionality Here
    Aves miave=new Aves("Aves",2);
1   miave.Piel();
    Console.WriteLine("Patas={0}",miave.getmispatas());

    Pez mipez=new Pez("Pez","escamas",0);
2   mipez.Piel();
    Console.WriteLine("Patas={0}",mipez.getmispatas());
    Console.Write("Press any key to continue . . . ");
    Console.ReadKey(true);
}
```

CLASES ABSTRACTAS



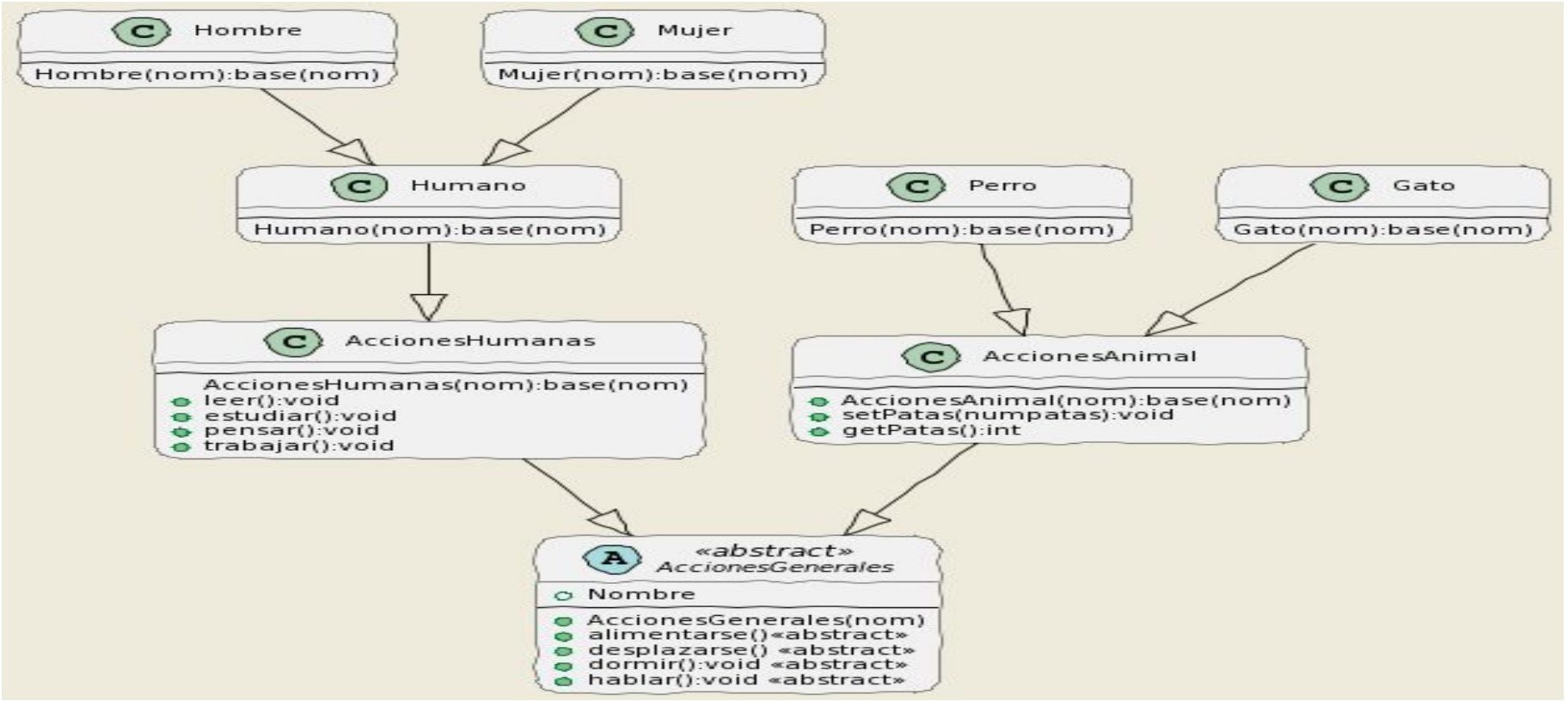
CLASES ABSTRACTAS COMO TRABAJO AUTÓNOMO

Ajuste el programa creando una clase llamada “Perro” con las siguientes características:

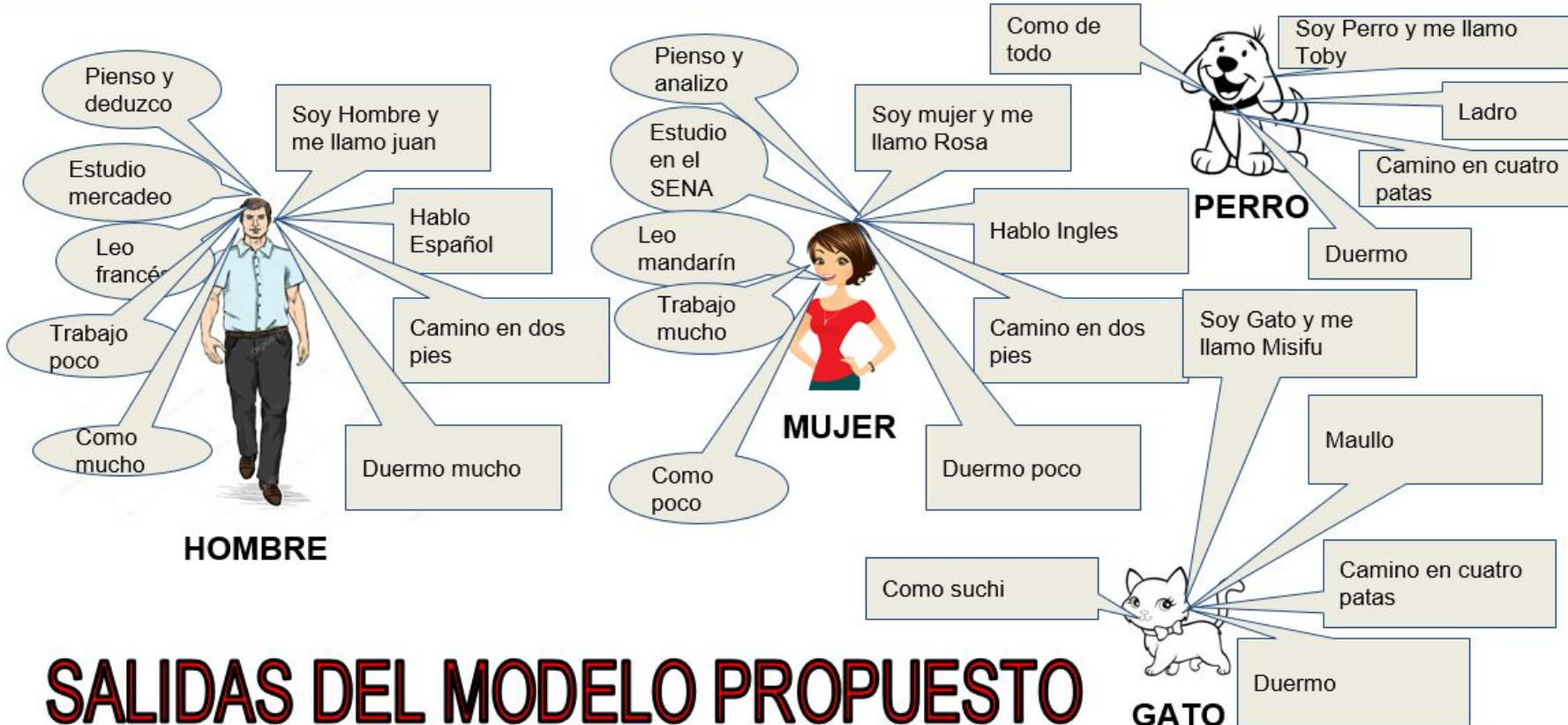
- Cuatro patas
- Tiene pelos
- Tiene orejas?

Debe mostrar mediante el método Piel: “*El perro tiene orejas, 4 patas y tiene pelos*”, utilizando atributos heredados desde la clase abstracta.

CLASES ABSTRACTAS COMO TRABAJO AUTÓNOMO EJERCICIO EN CLASE



CLASES ABSTRACTAS COMO TRABAJO AUTÓNOMO



SALIDAS DEL MODELO PROPUESTO

DESCARGAR ARCHIVOS

COMPILADOR SHARPDEV

DESCARGAR PDF

G R A C I A S

Línea de atención al ciudadano: 018000 910270
Línea de atención al empresario: 018000 910682

@SENAcomunica

www.sena.edu.co

CRÉDITOS

Realizado por el instructor José Fernando Galindo Suárez
jgalindos@sena.edu.co 2022