

Etapas de Profundización, Actividad práctica aplicada

Laura Alejandra Mendoza Prieto

Análisis y Desarrollo de Sistemas de Información, Fundación Universitaria UCompensar

Programación orientada a objetos

Profesor: JOSE HEMBER SOLORZANO PARDO

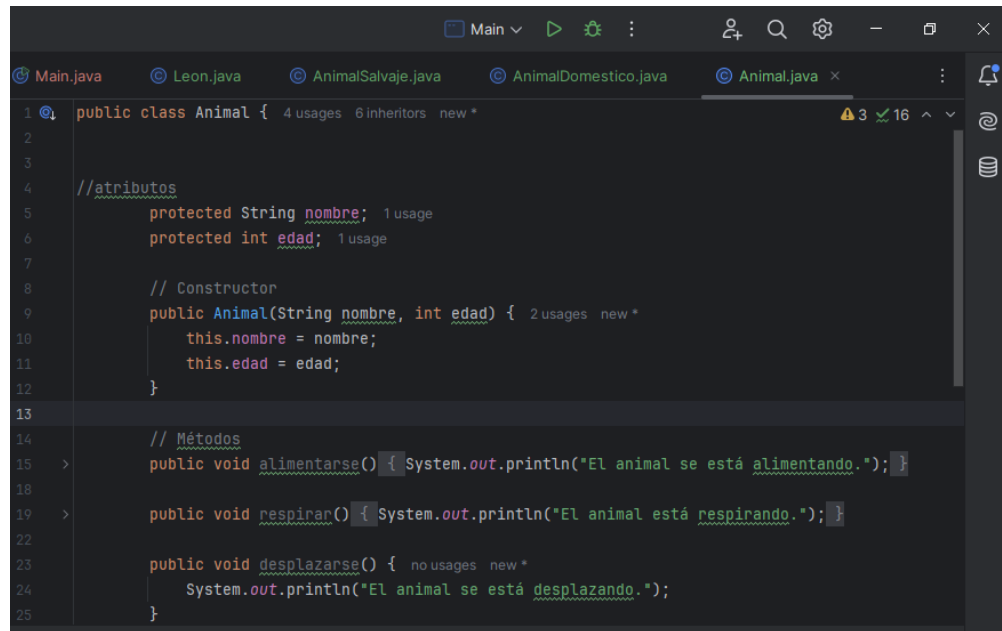
Octubre 13, 2024

Introducción

En el mundo de la programación orientada a objetos, los conceptos de herencia y polimorfismo son fundamentales para estructurar y organizar sistemas complejos de manera eficiente. La herencia permite crear una nueva clase a partir de una existente, de manera que la subclase hereda atributos y métodos de la clase primaria, promoviendo la reutilización de código, y facilitando la creación de relaciones jerárquicas. A partir de este concepto, surge el polimorfismo que otorga a los objetos la capacidad de adquirir múltiples formas, permitiendo que el mismo método actúe de manera diferente según el contexto.

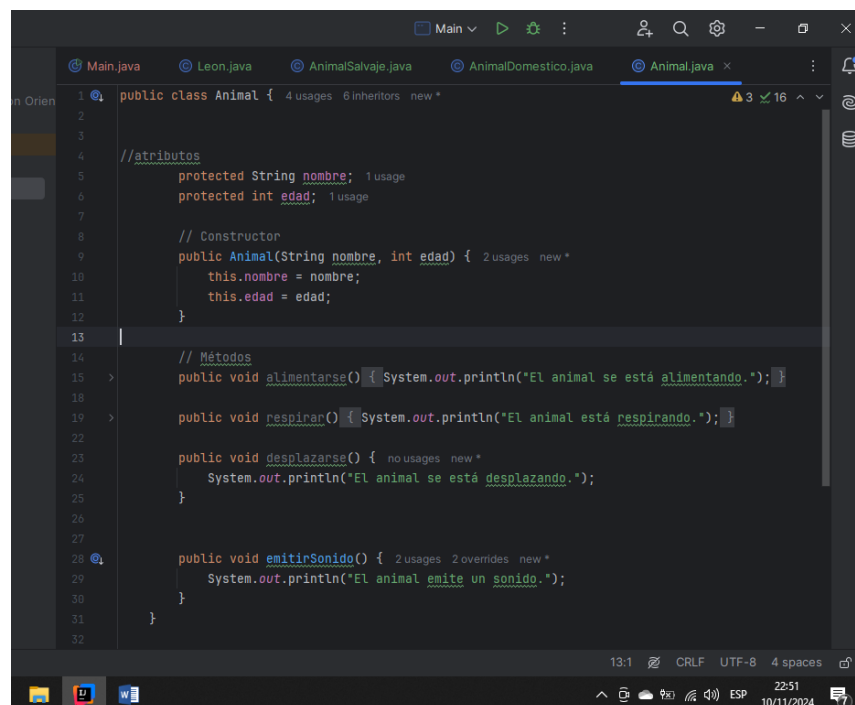
En este trabajo se aplicara conceptos de desarrollo de un modelo de animales, clasificándolos en animales domésticos y salvajes. Cada grupo posee comportamientos y características particulares, pero también comparten rasgos básicos comunes que ponemos definir en una clase base. Ademes se integra el uso de interfaces para añadir comportamientos adicionales, demostrando así el uso práctico de la herencia y el polimorfismo en la resolución de problemas del mundo real.

1. Listar un conjunto de características y comportamientos que tengan en común todos los animales.



```
1 public class Animal { 4 usages 6 inheritors new *
2
3
4 //atributos
5     protected String nombre; 1 usage
6     protected int edad; 1 usage
7
8     // Constructor
9     public Animal(String nombre, int edad) { 2 usages new *
10         this.nombre = nombre;
11         this.edad = edad;
12     }
13
14 // Métodos
15     public void alimentarse() { System.out.println("El animal se está alimentando."); }
16
17
18     public void respirar() { System.out.println("El animal está respirando."); }
19
20
21     public void desplazarse() { no usages new *
22         System.out.println("El animal se está desplazando.");
23     }
24
25 }
```

2. Crear la clase Superclase de dicho objeto del mundo real con sus propiedades y métodos.

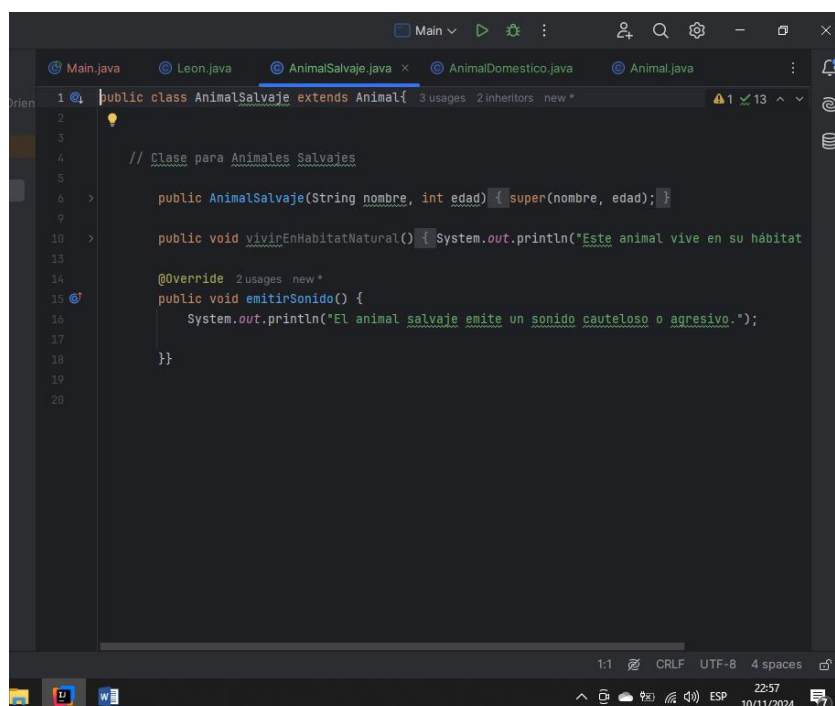


```
1 public class Animal { 4 usages 6 inheritors new *
2
3
4 //atributos
5     protected String nombre; 1 usage
6     protected int edad; 1 usage
7
8     // Constructor
9     public Animal(String nombre, int edad) { 2 usages new *
10         this.nombre = nombre;
11         this.edad = edad;
12     }
13
14 // Métodos
15     public void alimentarse() { System.out.println("El animal se está alimentando."); }
16
17
18     public void respirar() { System.out.println("El animal está respirando."); }
19
20
21     public void desplazarse() { no usages new *
22         System.out.println("El animal se está desplazando.");
23     }
24
25
26
27     public void emitirSonido() { 2 usages 2 overrides new *
28         System.out.println("El animal emite un sonido.");
29     }
30
31
32 }
```

3. Listar un conjunto de características y comportamientos que tengan en común los animales domésticos y los animales salvajes.

Características comunes de los Animales Domésticos	<ol style="list-style-type: none"> 1. Viven con humanos 2. Reciben cuidados humanos
Comportamientos comunes de los animales domésticos	<ul style="list-style-type: none"> - Socializar con humanos - Poderse entrenar
Características comunes de los animales Salvajes	<ol style="list-style-type: none"> 1. Viven en la naturaleza 2. Busca su propio alimento
Comportamientos comunes de los animales salvajes	<ul style="list-style-type: none"> - Defienden su territorio - Buscan su propio hogar o refugio

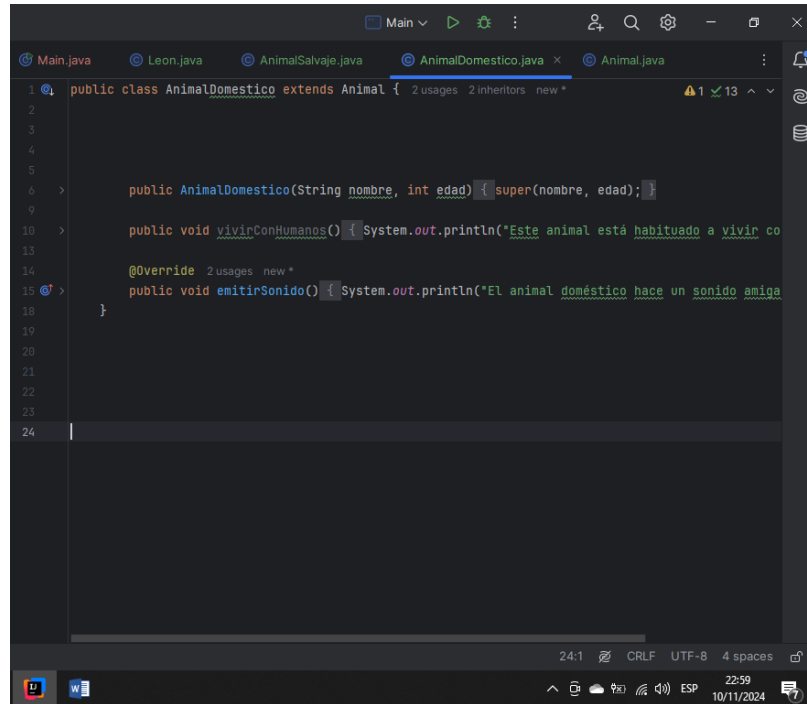
4. Crear las clases Subclase de dichos objetos del mundo real con sus propiedades y métodos.



```

1 public class AnimalSalvaje extends Animal {
2
3
4     // Clase para Animales Salvajes
5
6     public AnimalSalvaje(String nombre, int edad) { super(nombre, edad); }
7
8
9
10    public void vivirEnHabitatNatural() { System.out.println("Este animal vive en su hábitat"); }
11
12
13
14    @Override
15    public void emitirSonido() {
16        System.out.println("El animal salvaje emite un sonido cauteloso o agresivo.");
17    }
18
19
20
21
22
23
24
25
26
27
28
29

```

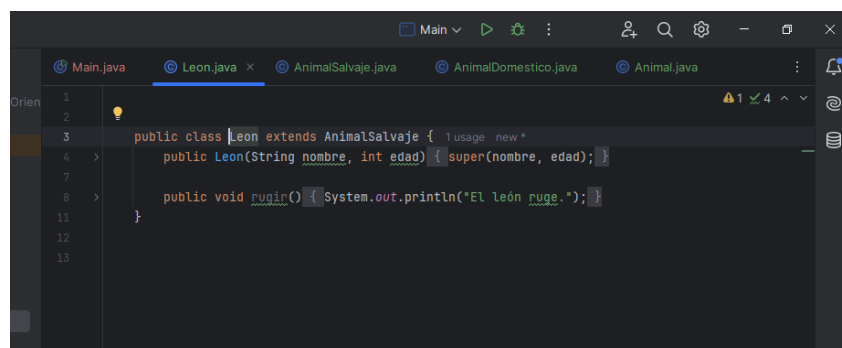


```

1 public class AnimalDomestico extends Animal {
2
3
4
5
6     public AnimalDomestico(String nombre, int edad) { super(nombre, edad); }
7
8
9
10    public void vivirConHumanos() { System.out.println("Este animal está habituado a vivir co
11
12
13    @Override
14    public void emitirSonido() { System.out.println("El animal doméstico hace un sonido amig
15 }
16
17
18
19
20
21
22
23
24

```

5. Crear 3 Subclases de cada una de las clases de animal doméstico y salvaje con sus propiedades y métodos.



```

1
2
3 public class Leon extends AnimalSalvaje {
4     public Leon(String nombre, int edad) { super(nombre, edad); }
5
6
7
8     public void rugir() { System.out.println("El león ruge."); }
9
10
11
12
13

```

```

Main.java | Leon.java | Gato.java x | AnimalSalvaje.java | AnimalDomestico.java |
1 public class Gato extends AnimalDomestico { no usages new *
2
3     public Gato(String nombre, int edad) { super(nombre, edad); }
4
5
6     public void maullar() { System.out.println("El gato maúlla."); }
7
8 }
9
10
11
12

```

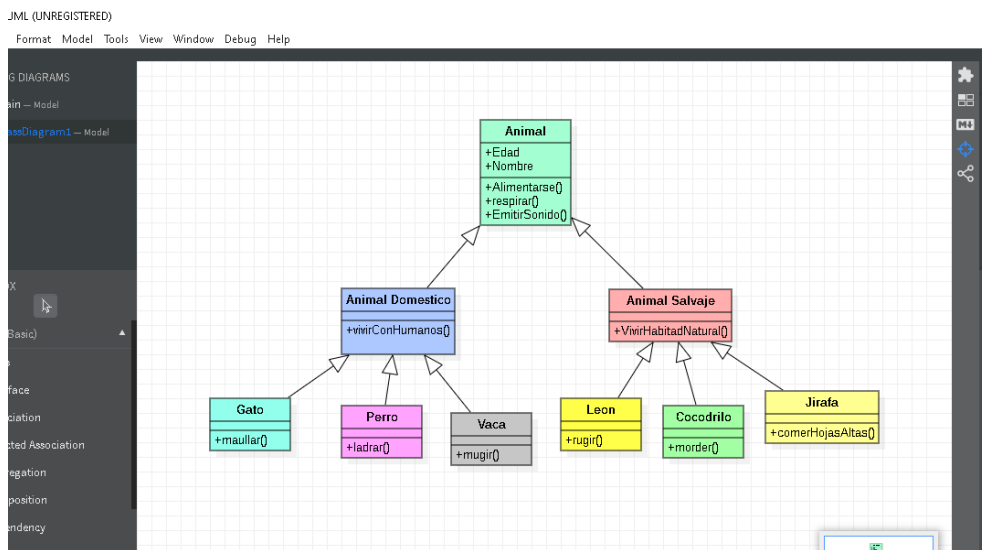
6. Añadir uno de los tipos de polimorfismo en un método de la Superclase Animal y sus Subclases (a elección del estudiante).

```

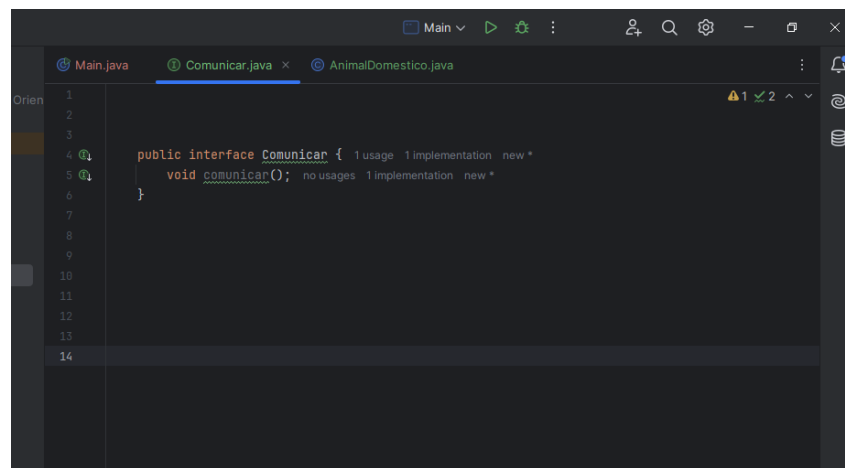
@.
public void emitirSonido() { 2 usages 2 overrides new *
    System.out.println("El animal emite un sonido.");
}
}

```

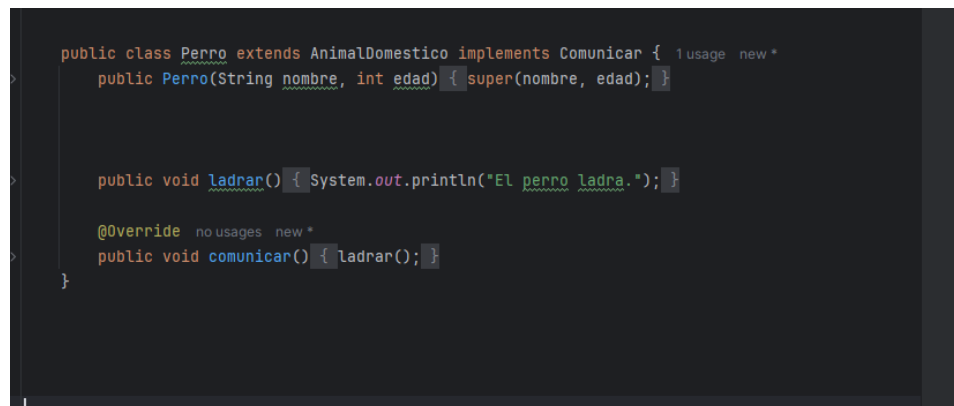
7. Crear el diagrama de clases de dicha estructura jerárquica.



8. Plantear una solución que incluya el concepto de interfaz (ya que la herencia de Java no permite que sea múltiple) e implementar en código dicha solución.



```
1  
2  
3  
4 public interface Comunicar {  
5     void comunicar();  
6 }  
7  
8  
9  
10  
11  
12  
13  
14
```



```
public class Perro extends AnimalDomestico implements Comunicar {  
    public Perro(String nombre, int edad) { super(nombre, edad); }  
  
    public void ladrar() { System.out.println("El perro ladra."); }  
  
    @Override  
    public void comunicar() { ladrar(); }  
}
```

Conclusión:

A través de esta actividad se ha explorado y aplicado los principios de herencia y polimorfismo en la programación orientada a objetos, estos conceptos facilitan la organización y estructura de sistemas complejos mediante relaciones jerárquicas y reutilización de código

Así mismo la implementación de interfaces permitió agregar comportamientos adicionales a las clases, mostrando la flexibilidad que este enfoque aporta a la programación de Java y como se puede superar la limitación de herencia simple.