POLIMORFISMO EN JAVA

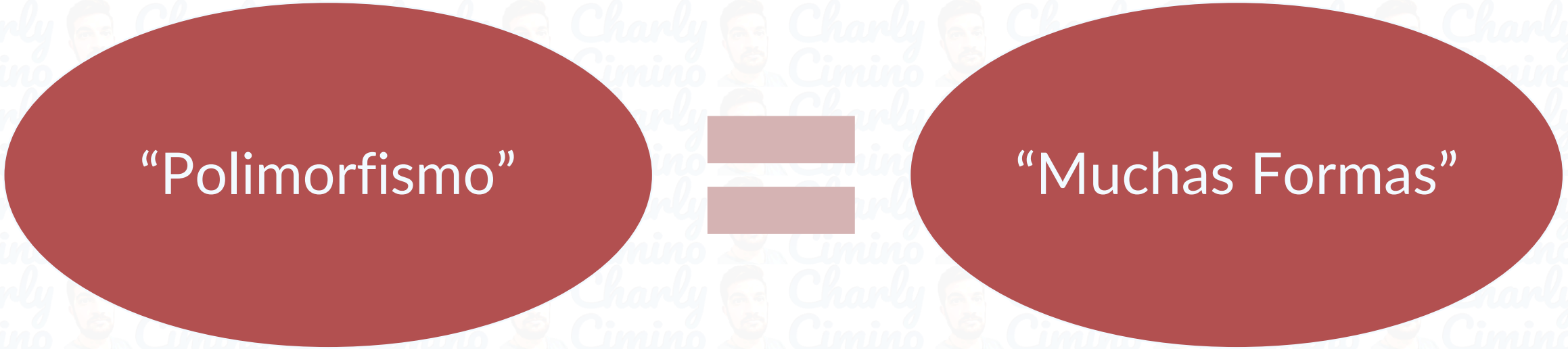
Charly Cimino

Polimorfismo en Java

Charly Cimino

Versión 2023.06.26

Definición

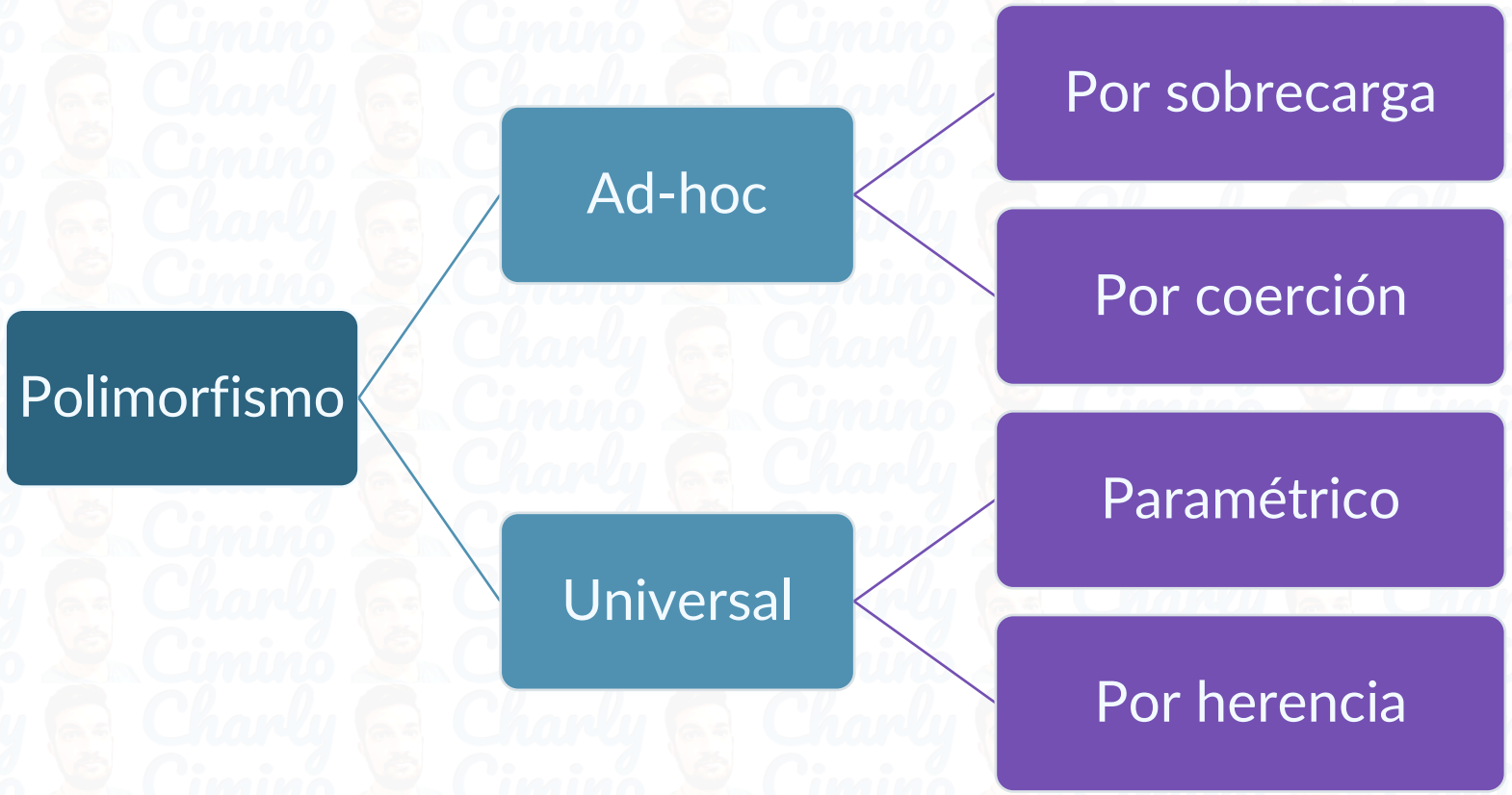


El pilar fundamental de la POO.

Nos permite programar de forma genérica y con clases desacopladas.
El software se vuelve escalable, mantenible y reutilizable.

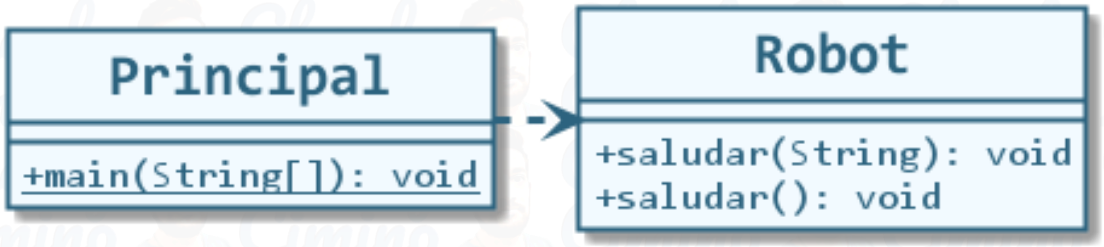
Clasificación

Cardelli y Wegner* clasifican al polimorfismo en diferentes categorías



* *Cardelli, L. & Wegner, P. "On Understanding Types, Data Abstraction, and Polymorphism". En: Computing Surveys (Diciembre, 1985). Vol. 17, n. 4, p. 471*

Ejemplo de polimorfismo por sobrecarga



Principal.java

```java
public class Principal {
    public static void main(String[] args) {
        Robot r = new Robot();
        r.saludar("Juan"); // "Hola Juan"
        r.saludar(); // "Hola extraño"
    }
}
```

Robot.java

```java
public class Robot {
    public void saludar(String nombre) {
        System.out.println("Hola " + nombre);
    }
    public void saludar() {
        System.out.println("Hola extraño");
    }
}
```

Ejemplo de polimorfismo por coerción

Principal.java

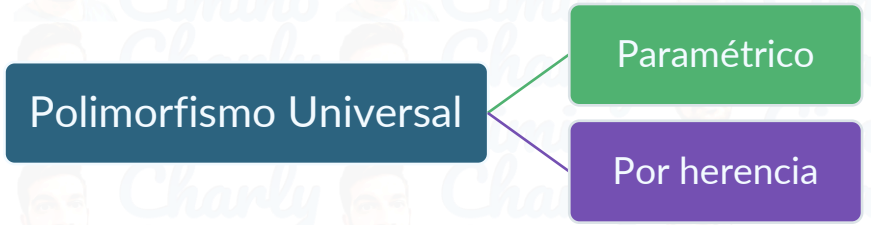
```java
public class Principal {
    public static void main(String[] args) {
        Triplicador t = new Triplicador();
        double num1 = 2.5;
        int num2 = 5;
        System.out.println(t.triplicar(num1)); // 7.5
        System.out.println(t.triplicar(num2)); // 15.0
    }
}
```

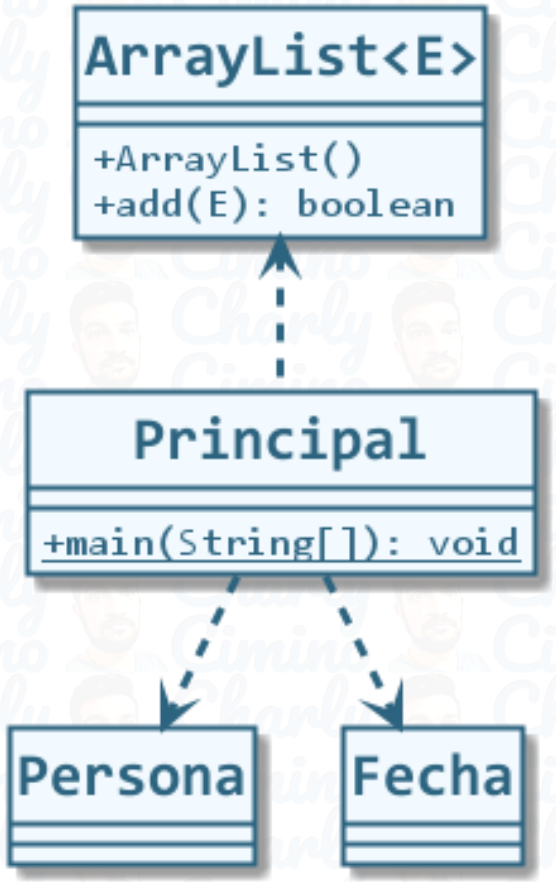
Triplicador.java

```java
public class Triplicador {
    public double triplicar(double num) {
        return 3 * num;
    }
}
```

Ejemplo de polimorfismo paramétrico

Este tema será ampliado cuando veamos 'Generics'.





Principal.java

```java
public class Principal {
    public static void main(String[] args) {
        ArrayList<Fecha> fechas = new ArrayList<Fecha>();
        fechas.add(new Fecha());
        ArrayList<Persona> personas = new ArrayList<Persona>();
        personas.add(new Persona());
    }
}
```

ArrayList.java

```java
public class ArrayList<E> {
    public boolean add(E e) {
        // Guarda el elemento...
    }
}
```

Polimorfismo por herencia

"Polimorfismo por herencia"

=

"Polimorfismo"

Cuando decimos "polimorfismo" a secas, por lo general nos referimos particularmente al polimorfismo por herencia.

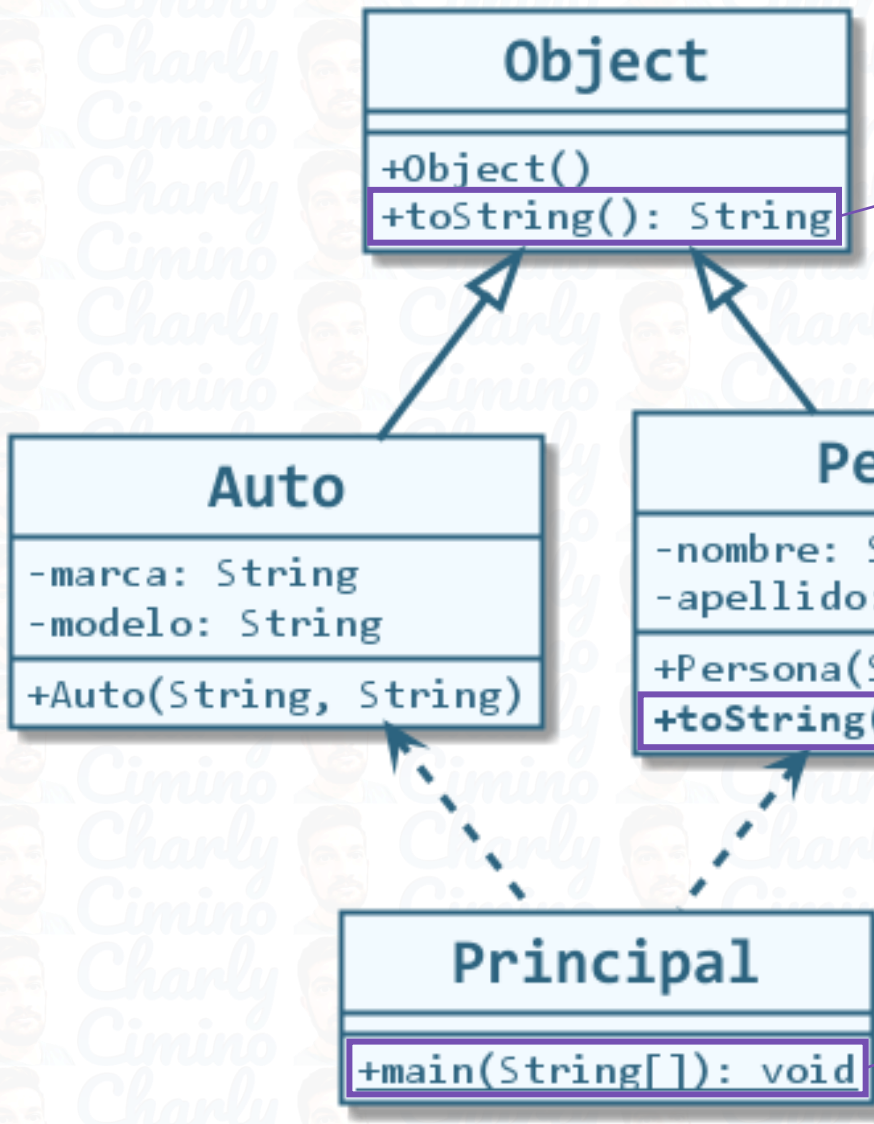
Polimorfismo por herencia

Mecanismo que permite enviar mensajes sintácticamente iguales a objetos de tipos distintos, cada uno con su propio comportamiento.

“Encender”

Ambos entienden el mensaje, pero lo responden de maneras diferentes.

Polimorfismo en acción



Object.java

```java
public String toString() {
    return getClass().getName() +
    '@' +
    Integer.toHexString(hashCode());
}
```

Persona.java

```java
@Override
public String toString() {
    return nombre + " " + apellido;
}
```

Principal.java

```java
public static void main(String[] args) {
    Auto a = new Auto("Ford", "Ka");
    System.out.println(a.toString());
    // Mostraría "Auto@a1b2c3d4"
    Persona p = new Persona("Ana", "Ríos");
    System.out.println(p.toString());
    // Mostraría "Ana Ríos"
}
```

Ligadura dinámica

No se sabe exactamente qué método se va a ejecutar.
Se liga en tiempo de ejecución el llamado al método con su respectiva implementación.

Principal.java

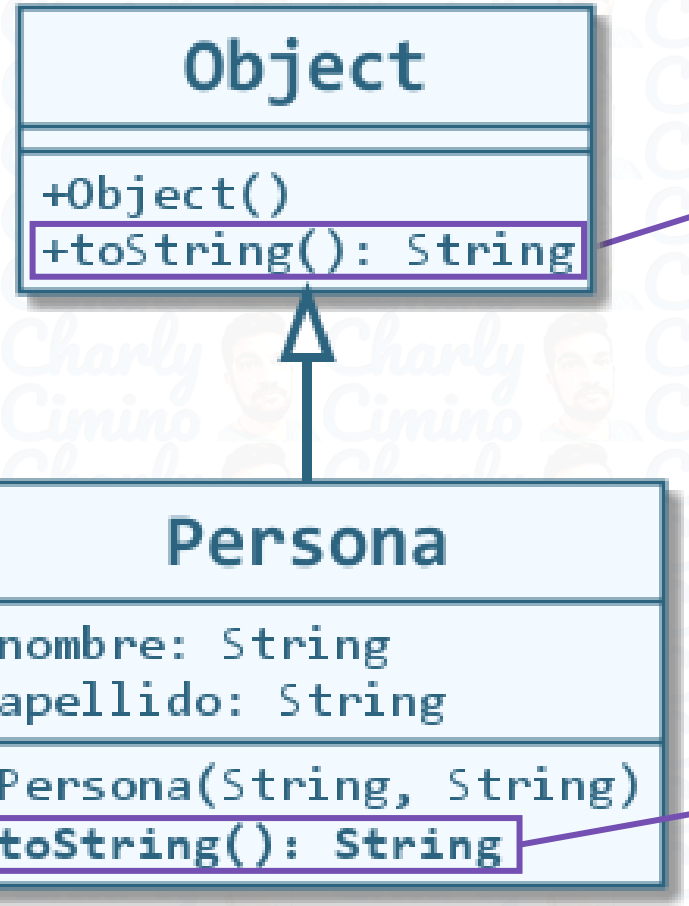
```java
public static void main(String[] args) {
    mostrar(new Persona("Ana", "Ríos"));
    // Mostraría "Ana Ríos"
}

public static void mostrar(Object x) {
    System.out.println( x.toString() );
}
```

Recordar concepto de "Upcasting"

A pesar de que llegue como **Object**, analiza el tipo de dato en tiempo de ejecución y deduce que es de tipo **Persona**.

Como en **Persona** está el método sobrescrito, se ejecuta tal implementación.



Object.java

```java
public String toString() {
    return getClass().getName() +
    '@' +
    Integer.toHexString(hashCode());
}
```

Persona.java

```java
@Override
public String toString() {
    return nombre + " " + apellido;
}
```

Ligadura dinámica

No se sabe exactamente qué método se va a ejecutar.
Se liga en tiempo de ejecución el llamado al método con su respectiva implementación.

Principal.java

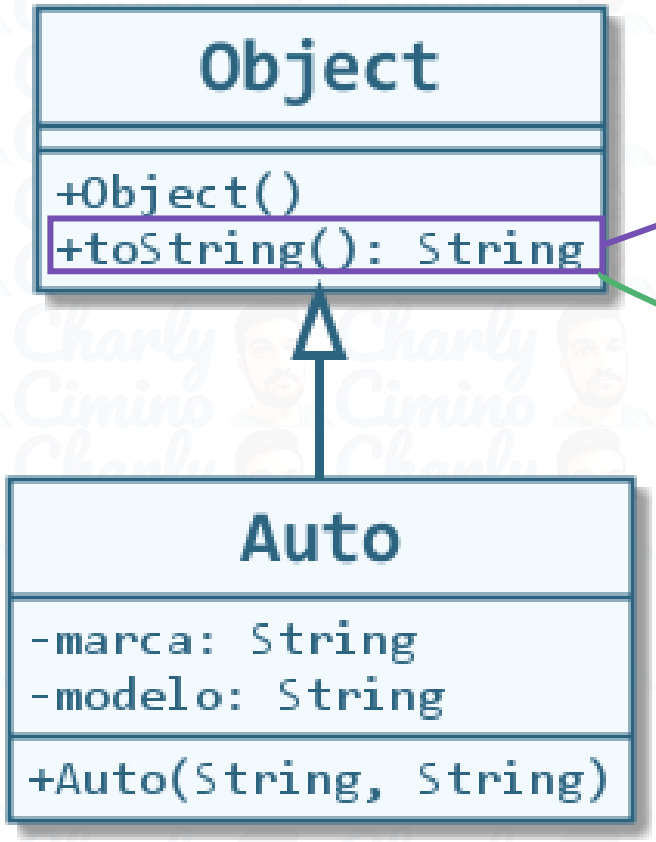
```
public static void main(String[] args) {
    mostrar(new Auto("Ford", "Ka"));
    // Mostraría "Auto@a1b2c3d4"
}

public static void mostrar(Object x) {
    System.out.println( x.toString() );
}
```

Recordar concepto de "Upcasting"

A pesar de que llegue como **Object**, analiza el tipo de dato en tiempo de ejecución y deduce que es de tipo **Auto**.

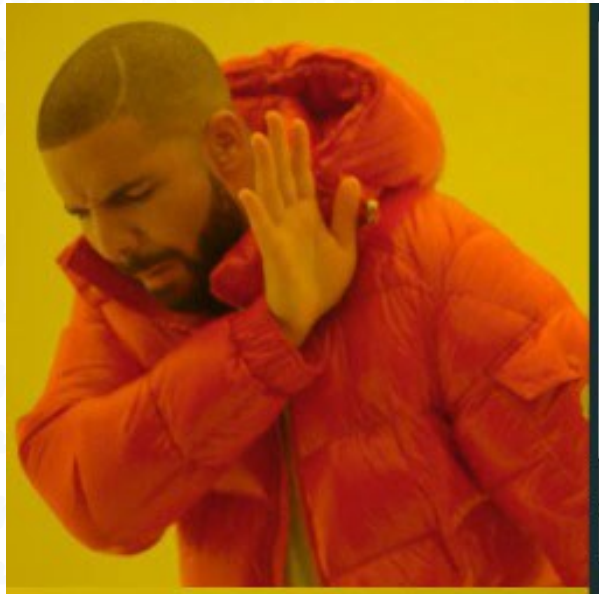
Como en **Auto** no está el método sobrescrito, se busca tal implementación en las respectivas superclase hasta encontrar una implementación.

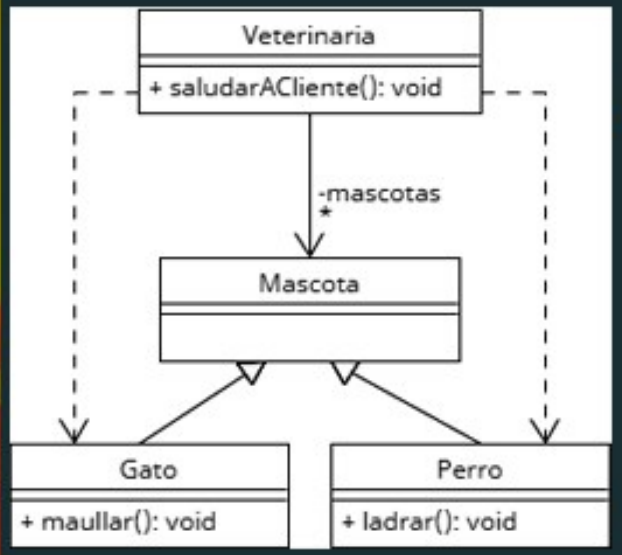


Object.java

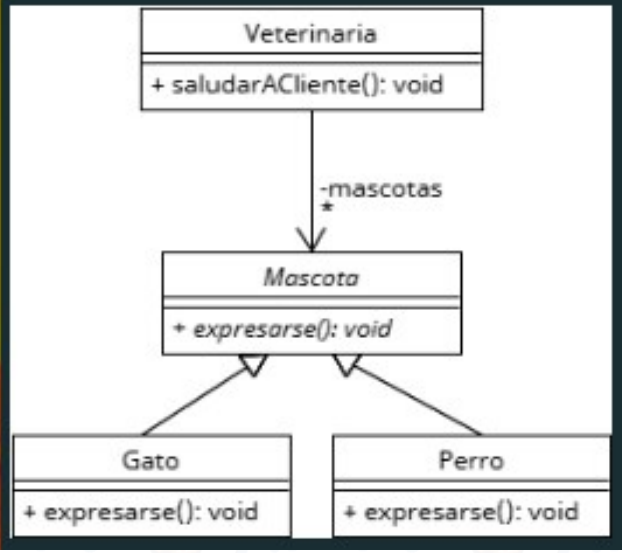
```
public String toString() {
    return getClass().getName() +
    '@' +
    Integer.toHexString(hashCode());
}
```

¿Por qué usar polimorfismo?






```java
// Método que NO usa polimorfismo
public void saludarACliente() {
    for (Mascota mascota : mascotas) {
        if (mascota instanceof Perro) {
            ((Perro) mascota).ladrar();
        } else if (mascota instanceof Gato) {
            ((Gato) mascota).maullar();
        }
    }
}
```




```java
// Método que usa polimorfismo
public void saludarACliente() {
    for (Mascota mascota : mascotas) {
        mascota.expresarse();
    }
}
```

FIN DE LA PRESENTACIÓN

Encontrá más como estas en mi sitio web.