



UNIDAD 9: INTERFACES

CLASE I: DEFINICIÓN DE INTERFACES

CLASES ANIMALES

CARACOL

- Mover
- Comer
- Dormir

PERRO

- Mover
- Comer
- Dormir
- hacerSonido

GATO

- Mover
- Comer
- Dormir
- hacerSonido

CLASES ANIMALES: HERENCIA



CLASES ANIMALES: HERENCIA

ANIMAL

- Mover
- Comer
- Dormir
- hacerSonido

CARACOL hereda
OBLIGADO A
IMPLEMENTAR SONIDO

PERRO
hereda

GATO
hereda

CLASES ANIMALES: HERENCIA

ANIMAL

- Mover
- Comer
- Dormir
- hacerSonido

CARACOL hereda
OBLIGADO A
IMPLEMENTAR SONIDO

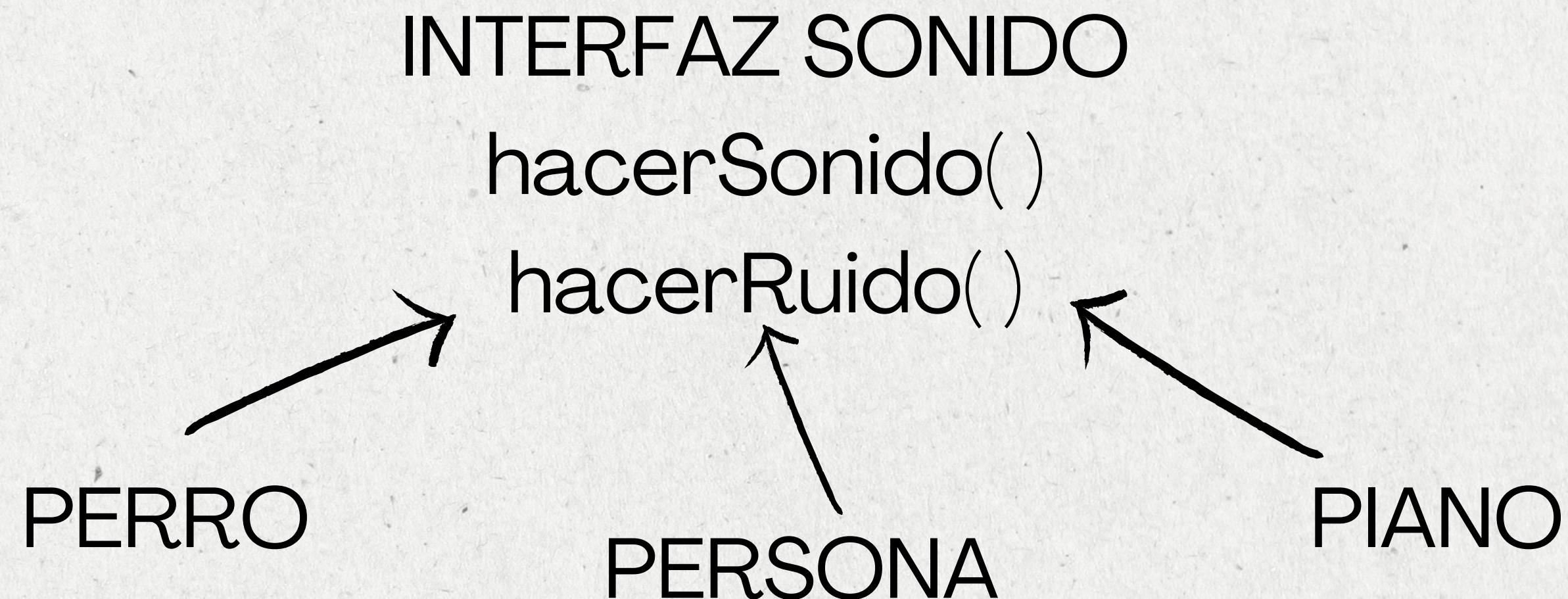
PERRO
hereda

GATO
hereda

PERSONA?
PIANO?
Hacen sonido!

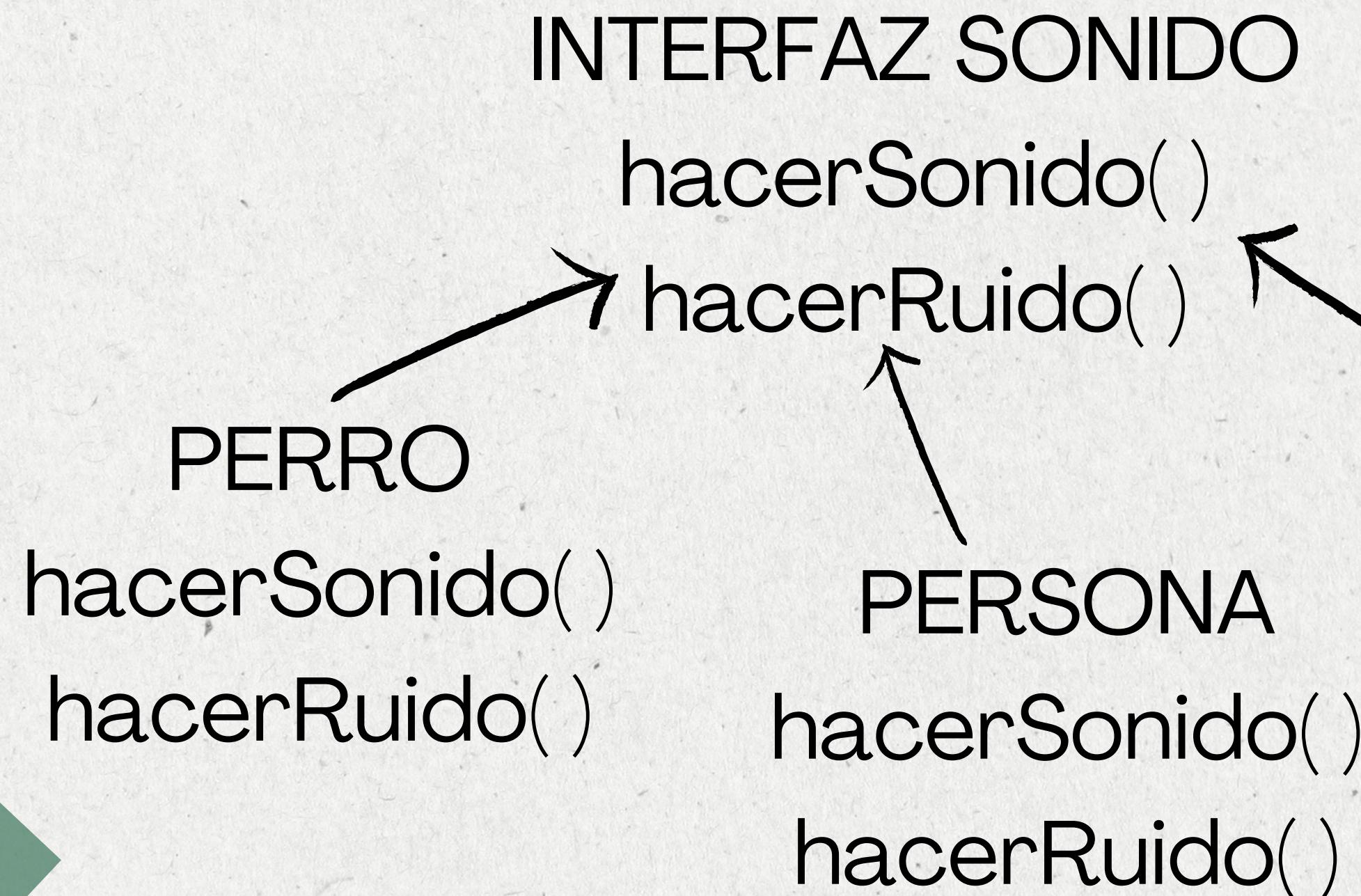
INTERFACES

Expresan funcionalidades comunes de clases que no tienen ninguna relación entre sí



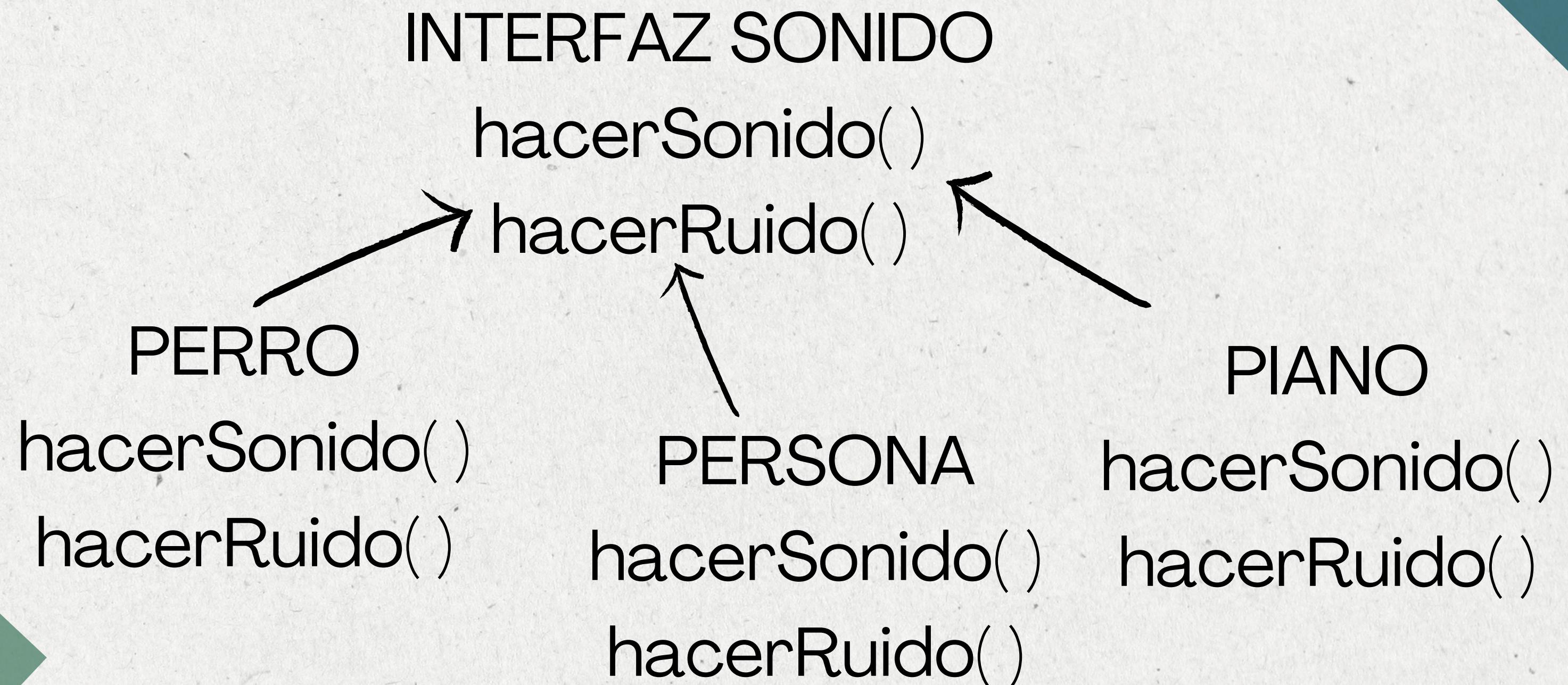
INTERFACES

Todas las clases que implementen una interfaz tienen que tener todos sus métodos!



INTERFACES

Todas las clases que implementen una interfaz tienen que tener todos sus métodos!



INTERFACES

Una clase puede implementar múltiples interfaces

INTERFAZ SONIDO

hacerSonido()

hacerRuido()

INTERFAZ MOVIMIENTO

andar()

correr()

PERSONA

hacerSonido()

hacerRuido()

andar()

correr()

CÓMO CREAR INTERFACES

```
public interface Sonido {  
    public void hacerSonido();  
    public void hacerRuido();  
}
```

CÓMO CREAR INTERFACES

```
public class Perro implements Sonido{
    Perro(){}
    public void hacerSonido() {
        System.out.println("Guau!");
    }
    public void hacerRuido() {
        for(int i = 0; i < 10;i++) {
            System.out.print("GUAU!");
        }
    }
}
```

CÓMO CREAR INTERFACES

```
public class main {  
    public static void main(String[] args) {  
        Perro p = new Perro();  
        p.hacerSonido();  
        p.hacerRuido();  
    }  
}
```

Guau!

GUAU! GUAU! GUAU! GUAU! GUAU! GUAU! GUAU! GUAU! GUAU!

CÓMO CREAR INTERFACES

```
public interface Movimiento {  
    public int andar();  
    public int correr();  
}
```

CÓMO CREAR INTERFACES

```
public class Persona implements Sonido,Movimiento{  
    int posicion = 0;  
    Persona(){}
    public int andar() {posicion++; return posicion;}
    public int correr(){posicion+=3; return posicion;}
    public void hacerSonido() {
        System.out.println("Hola!");
    }
    public void hacerRuido() {
        System.out.println("*Se queja en humano*");
    }
}
```

CÓMO CREAR INTERFACES

```
public class main {  
    public static void main(String[] args) {  
        Persona p = new Persona();  
        p.hacerSonido();  
        p.hacerRuido();  
        for(int i = 0; i < 3;i++)  
            System.out.print(p.andar()+" ");  
        for(int i = 0; i < 3;i++)  
            System.out.print(p.correr()+" ");  
    }  
}
```

Hola!

Se queja en humano

1 2 3 6 9 12

INTERFACES CON ATRIBUTOS

Las interfaces **pueden** tener atributos.

- Son estáticos y finales (solo pueden ser modificados desde la interfaz)
- Pueden ser leídas con la clase interfaz o con un objeto de una clase que la implemente.

INTERFACES CON ATRIBUTOS

```
public interface Movimiento {  
    int[] ejesDisponibles = {1,0,0};  
    public int andar();  
    public int correr();  
}
```

```
System.out.println(Arrays.toString(p.ejesDisponibles));  
System.out.println(  
    Arrays.toString(  
        Movimiento.ejesDisponibles));
```

[1, 0, 0]

INTERFACES CON MÉTODOS IMPLEMENTADOS

Las interfaces **pueden** tener métodos implementados dentro.

- Se denominan métodos de extensión.
- Pueden ser públicos, privados, estáticos y no estáticos.

INTERFACES CON MÉTODOS IMPLEMENTADOS

```
public interface Sonido {  
    public void hacerSonido();  
    public void hacerRuido();  
    default void durmiendo() {  
        System.out.println("Zzz");  
    }  
}
```

Zzz
Zzz

```
Persona p = new Persona();  
Perro pr = new Perro();  
pr.durmiendo();  
p.durmiendo();
```

INTERFACES CON MÉTODOS IMPLEMENTADOS

```
public class Piano implements Sonido{  
    Piano(){  
        public void hacerSonido() {  
            System.out.println("Do!");  
        }  
        public void hacerRuido() {  
            for(int i = 0; i < 10;i++) {  
                System.out.print("DO!MI!RE!");  
            }  
        }  
        @Override public void durmiendo(){  
            System.out.println("Estoy apagado");  
        }  
    }
```

```
Piano piano = new Piano();  
piano.durmiendo();
```

Estoy apagado

INTERFACES CON MÉTODOS IMPLEMENTADOS

```
public interface Movimiento {  
    int[] ejesDisponibles = {1,0,0};  
    public int andar();  
    public int correr();  
    public static double desplazar(  
        double posicion, double desplazamiento) {  
        return posicion+desplazamiento;  
    }  
}
```

```
System.out.println(Movimiento.desplazar(13, -4));
```

INTERFACES CON MÉTODOS IMPLEMENTADOS

Pueden tener métodos privados como “métodos auxiliares” que otros métodos podrán utilizar.

```
public static double desplazar(  
    double posicion, double desplazamiento) {  
    return posicion+calcularEnMetros(desplazamiento);  
}  
  
private static double calcularEnMetros(double tam_km) {  
    return tam_km*1000;  
}
```

EJERCICIOS

Ejercicio 1:

Crea la interfaz **Figura** que tendrá dos métodos:

- **calcularArea()** devolverá un decimal.
- **calcularPerimetro()** devolverá un decimal.
- **areaCirculo()** es un método estático al que se le pasa un radio y devuelve un decimal. Calcula el área con la fórmula: número pi por el radio al cuadrado.
Para recuperar el número pi debéis utilizar el método siguiente
- **redondearPI()** es un método privado que recuperará el número pi de la librería math y lo redondeará a 2 decimales.

EJERCICIOS

Crea la clase **Círculo** que implemente **Figura** y sus dos métodos calcularArea y Perímetro($2*\pi*\text{radio}$). Además, tiene un radio que deberá tomar valor en el constructor.

- Crea la clase **Rectángulo** que implemente Figura y sus dos métodos calcularArea($\text{base}*\text{altura}$) y Perímetro ($2\text{base}+2\text{altura}$). Además, tendrá una base y altura que deberán tomar valor en el constructor.
- Crea un main que contenga un Circulo, un Rectángulo y muestre sus perímetros y áreas. Además comprueba que el cálculo del círculo sea correcto llamando al método estático de la interfaz con los mismos valores.

EJERCICIOS

Ejercicio 2:

Crea la interfaz **Animal** que tendrá tres métodos:

- **dormir()** devolverá un decimal.
- **avanzar()** recibirá y devolverá un decimal.
- **alimentarse()** recibirá y devolverá un decimal.

Crea la interfaz **Volador** que tendrá un método:

- **volar()** que recibirá y devolverá un decimal.

Crea la interfaz **Nadador** que tendrá un método:

- **nadar()** que recibirá y devolverá un decimal.

EJERCICIOS

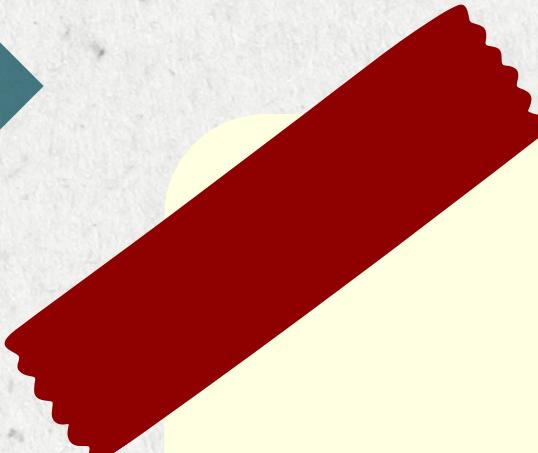
Crea la clase **Pato** que implementará las tres interfaces:

- Tendrá 1 atributo que será **energía**, un decimal.
- **dormir()** cada vez que se invoque el método disminuirá en 1 su energía.
- **avanzar()** disminuirá la energía en la cantidad pasada por parámetro.
- **volar()** disminuirá la energía dos veces la cantidad pasada por parámetro.
- **nadar()** disminuirá la energía la mitad de la cantidad pasada por parámetro.

Si cualquiera de los métodos anteriores detecta que la energía es 0 no se ejecutará.

- **alimentarse()** aumentará la energía en la cantidad pasada por parámetro

EJERCICIOS



Crea un main que contenga un objeto de la clase Pato y que aleatoriamente llame a los 4 métodos que contiene con cantidades decimales aleatorias entre 1 y 10. La energía inicial del pato será de 20 y se deberá mostrar en todo momento el avance de la energía del pato mostrándola por pantalla.