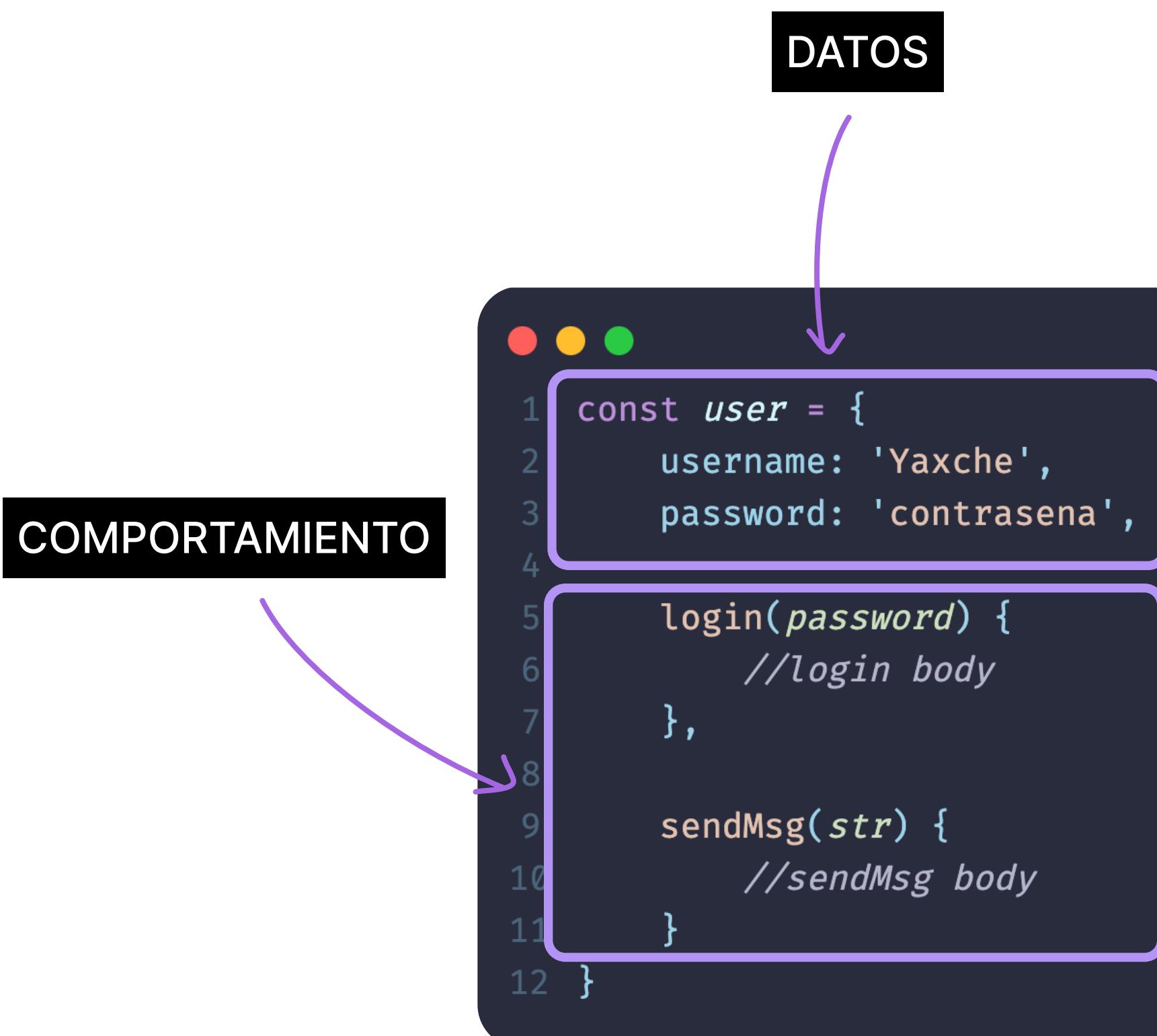


QUÉ ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?



Forma particular de pensar y escribir código para resolver problemas.

- Paradigma de programación basado en el concepto de objetos.
- Usamos objetos para modelar (describir) elementos o características reales o abstractas.
- Los objetos pueden contener datos (propiedades) y código (métodos). Al usar objetos, podemos empaquetar o contener datos y comportamientos en un solo bloque

QUÉ ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?

- En la POO, los objetos son bloques de código auto-contenidos (mini aplicaciones)
- Los objetos son los bloques de construcción de las aplicaciones, estos interactúan entre ellos.
Los objetos interactúan entre ellos a través de una interfaz pública (API): métodos que el código fuera de los objetos puede usar para comunicarse con el objeto.
- La POO fue desarrollada para organizar código, de tal manera que sea más flexible y mantenible.

CLASES E INSTANCIAS



CLASE

✗ Solo es una representación, NO es la sintaxis correcta de JavaScript

✗ JavaScript NO acepta clases como tal

```
1 User = {  
2   username  
3   password  
4   email  
5  
6   login(password) {  
7     //login body  
8   },  
9  
10  sendMsg(str) {  
11    //sendMsg body  
12  }  
13 }
```

INSTANCIA



```
1 {  
2   username: 'Yaxche',  
3   password: 'contrasena',  
4   email: 'yaxche@mail.com',  
5  
6   login(password) {  
7     //login body  
8   },  
9  
10  sendMsg(str) {  
11    //sendMsg body  
12  }  
13 }
```

Nuevo objeto creado. Es la casa (real) creada con el plano que teníamos

INSTANCIA



```
1 {  
2   username: 'Diego',  
3   password: 'contrasena-2',  
4   email: 'diego@mail.com',  
5  
6   login(password) {  
7     //login body  
8   },  
9  
10  sendMsg(str) {  
11    //sendMsg body  
12  }  
13 }
```

new User('Yaxche')

new User('Diego')

new User('Laura')

INSTANCIA



```
1 {  
2   username: 'Laura',  
3   password: 'contrasena-3',  
4   email: 'laura@mail.com',  
5  
6   login(password) {  
7     //login body  
8   },  
9  
10  sendMsg(str) {  
11    //sendMsg body  
12  }  
13 }
```

LOS 4 PRINCIPALES FUNDAMENTOS DE LA POO

abstracción

encapsulación

herencia

polimorfismo

→ 🤔 Y cómo creamos estas clases para que representen objetos reales?



LOS 4 PRINCIPALES FUNDAMENTOS DE LA POO

abstracción

encapsulación

herencia

polimorfismo

```
1 Phone {  
2     charge  
3     volume  
4     voltage  
5     temperature  
6  
7     homeBtn() {}  
8     volumeBtn() {}  
9     screen() {}  
10    verifyVolt() {}  
11    verifyTemp() {}  
12    vibrate() {}  
13    speakers() {}  
14    frontCameraOn() {}  
15    frontCameraOff() {}  
16    rearCameraOn() {}  
17    rearCameraOff() {}  
18 }
```

REAL



```
1 Phone {  
2     charge  
3     volume  
4  
5     homeBtn() {}  
6     volumeBtn() {}  
7     screen() {}  
8 }
```

ABSTRAÍDO



→ Abstracción es ignorar u ocultar los detalles sin importancia, dándonos la oportunidad de tener un panorama más amplio en vez de ver muchos detalles que no son necesarios en la implementación.

LOS 4 PRINCIPALES FUNDAMENTOS DE LA POO

👉 Esto no es código JavaScript!

abstracción

encapsulación

herencia

polimorfismo

✗ NO los podemos tocar fuera de la clase

✓ Podemos acceder a estos datos DENTRO de la clase

✗ NO los podemos tocar fuera de la clase

```
1 User {  
2   username  
3   private password  
4   private email  
5  
6   login(word) {  
7     this.password === word  
8   }  
9  
10  comment(text) {  
11    this.checkSPAM(text)  
12  }  
13  
14  private checkSPAM() {}  
15 }
```

PARA QUÉ?

→ Previene que código externo manipule accidentalmente nuestra lógica

→ Nos permite jugar con la lógica de los objetos sin impactar al código externo

→ La encapsulación mantiene las propiedades y los métodos privadas dentro la clase para que no sean accesibles por código fuera de las clases.

LOS 4 PRINCIPALES FUNDAMENTOS DE LA POO

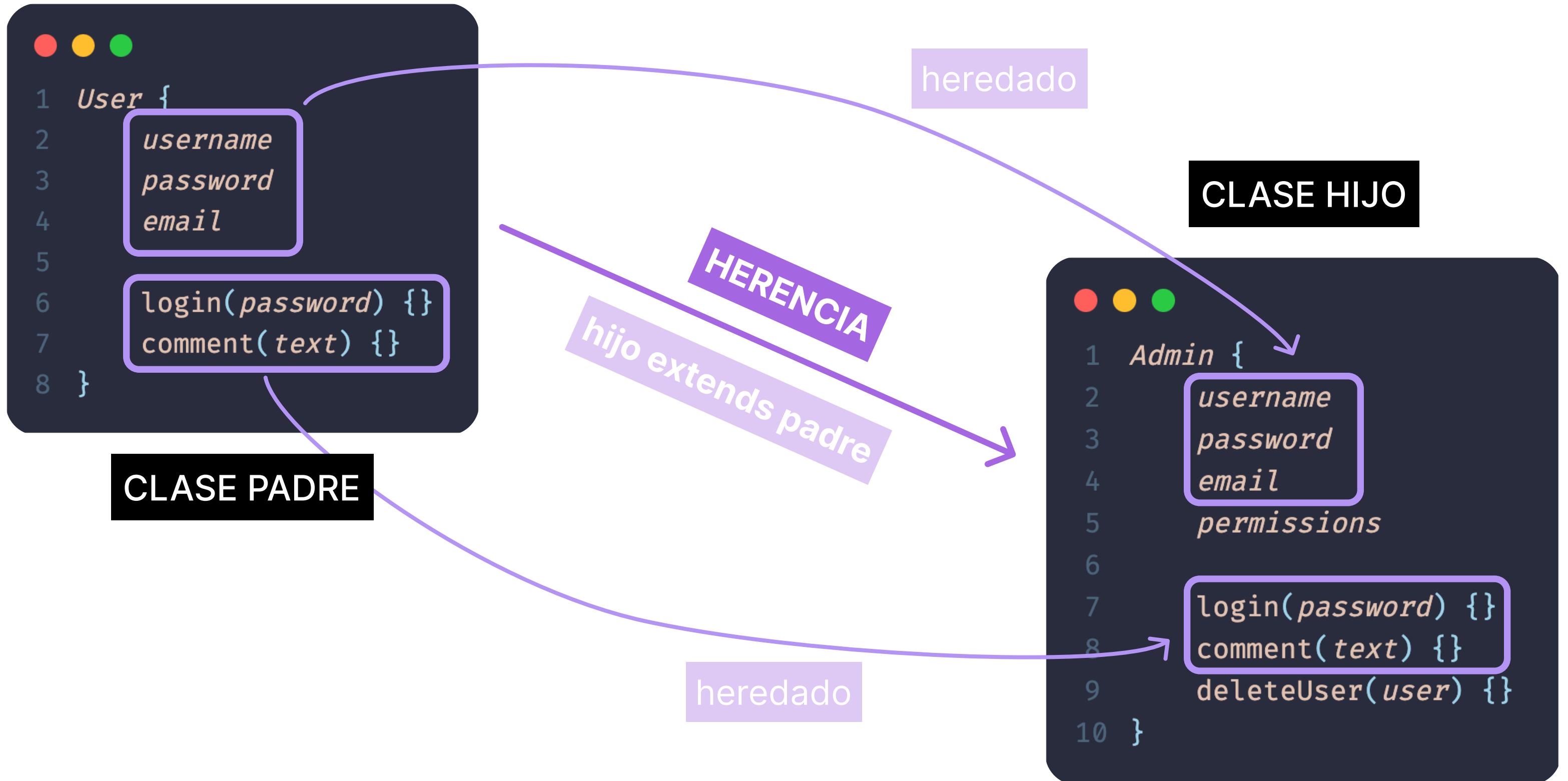
👉 Esto no es código JavaScript!

abstracción

encapsulación

herencia

polimorfismo



→ La herencia nos permite tener clases con atributos (propiedades) y comportamientos (métodos) repetidos en una clase hijo. Así podemos reusar lógica manteniendo nuestro código DRY

LOS 4 PRINCIPALES FUNDAMENTOS DE LA POO

👉 Esto no es código JavaScript!

abstracción

encapsulación

herencia

polimorfismo

```
1 User {  
2   username  
3   password  
4   email  
5  
6   login(password) {}  
7   comment(text) {}  
8 }
```

CLASE PADRE

CLASE HIJO

propiedades y métodos exclusivos de la clase hijo

```
1 Admin {  
2   username  
3   password  
4   email  
5   permissions  
6  
7   login(password) {}  
8   comment(text) {}  
9   deleteUser(user) {}  
10 }
```

→ La herencia nos permite tener clases con atributos (propiedades) y comportamientos (métodos) repetidos en una clase hijo. Así podemos reusar lógica manteniendo nuestro código DRY

LOS 4 PRINCIPALES FUNDAMENTOS DE LA POO

👉 Esto no es código JavaScript!

abstracción

encapsulación

herencia

polimorfismo

HERENCIA

CLASE PADRE

HERENCIA

```
1 Admin {  
2   username  
3   password  
4   email  
5   permissions  
6  
7   login(password, key) {}  
8   deleteUser(user)  
9 }
```

CLASE HIJO

```
1 User {  
2   username  
3   password  
4   email  
5  
6   login(password) {}  
7   sendMsg(text)  
8 }
```

```
1 Author {  
2   username  
3   password  
4   email  
5   posts  
6  
7   login(password) {}  
8   writePost(user)  
9 }
```

CLASE HIJO

→ El polimorfismo nos deja sobreescribir métodos de la clase padre