Guía teórico-práctica de clases abstractas y Estáticos

¿Qué son las clases abstractas en TypeScript?

Las clases abstractas son clases que no se pueden instanciar directamente y sirven como base para otras clases. Definen una interfaz común que deben implementar las subclases.

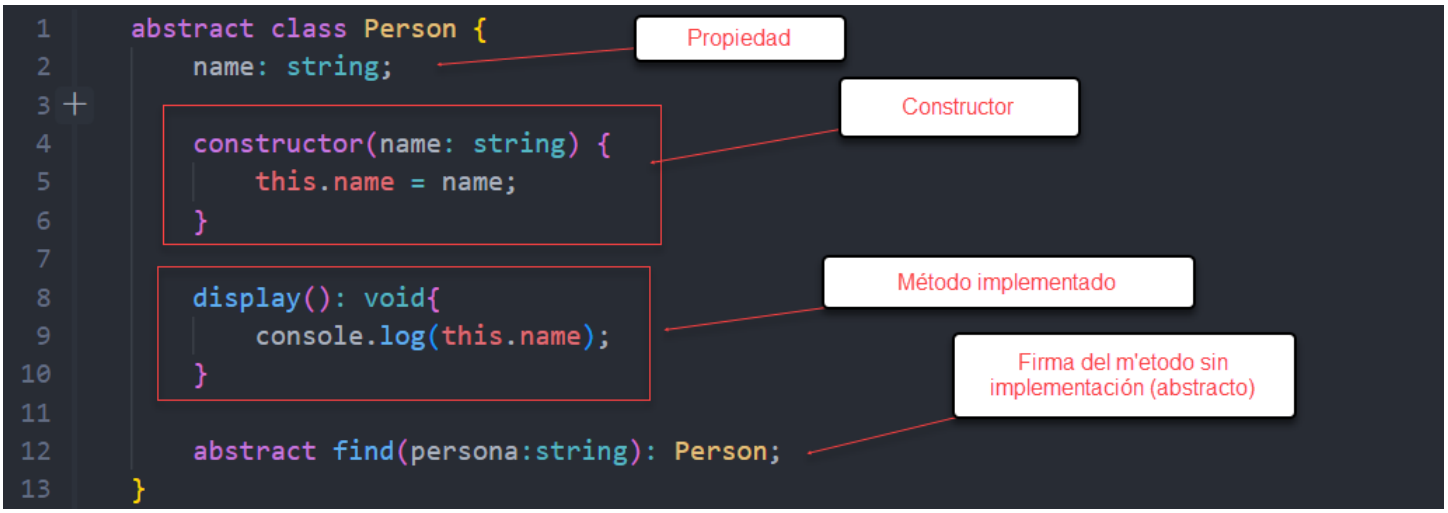
Características de las clases abstractas:

No se pueden crear instancias directamente de una clase abstracta.

Pueden contener métodos abstractos que son declarados pero no implementados.

Los métodos abstractos deben ser implementados por las subclases concretas.

Pueden contener métodos concretos con implementación.




```typescript
abstract class Person {
    name: string;

    constructor(name: string) {
        this.name = name;
    }

    display(): void{
        console.log(this.name);
    }

    abstract find(persona:string): Person;
}
```

```typescript
class Employee extends Person {
    empCode: number;

    constructor(name: string, code: number) {
        super(name); // must call super()
        this.empCode = code;
    }

    find(name:string): Person {
        return new Employee(name, 1);
    }
}

let emp: Person = new Employee("James", 100);
emp.display(); //James

let emp2: Person = emp.find('Steve');

emp2.display(); //Steve
```

```ts
abstract class Animal {
    abstract makeSound(): void;

    move(): void {
      console.log("Moving along!");
    }
  }

  class Dog extends Animal {
    makeSound() {
      console.log("Woof woof!");
    }
  }

  const d = new Dog();
  d.makeSound(); // Woof woof!
  d.move(); // Moving along!
```

Clase abstracta

Método abstracto

Método con implemnetación

Herencia

Implementación del método

instancia

```ts
abstract class Shape {
    abstract getArea(): number;
  }

  class Circle extends Shape {
    constructor(private radius: number) {
      super();
    }

    getArea() {
      return Math.PI * this.radius ** 2;
    }
  }

const x = new Circle(3);
console.log(x.getArea());
```

```ts
abstract class Persona {
    abstract nombre: string;
    abstract edad: number;

    abstract saludar(): void;
}

class Estudiante extends Persona {
    constructor(public nombre: string, public edad: number) {
        super();
    }

    saludar(): void {
        console.log(`Hola, soy ${this.nombre} y tengo ${this.edad} años.`);
    }
}

const estudiante = new Estudiante('Juan', 23);
estudiante.saludar();
```

propiedades abstractas

```ts
class Employee {
  public empName: string;
  protected empCode: number;

  constructor(name: string, code: number) {
    this.empName = name;
    this.empCode = code;
  }
}
```

Modificador de acceso public, private y protected

```ts
class SalesEmployee extends Employee {
  private department: string;

  constructor(name: string, code: number, department: string) {
    super(name, code);
    this.department = department;
  }
  mostrar() {
    console.log(this.empName);
    console.log(this.empCode);
    console.log(this.department);
  }
}

let emp = new SalesEmployee("John Smith", 123, "Sales");
emp.empName = "Jane Doe";
emp.mostrar();
```

¿Qué es static en TypeScript?

La palabra clave static en TypeScript se utiliza para declarar miembros estáticos de una clase. Los miembros estáticos (propiedades y métodos) son accesibles en la clase directamente sin necesidad de instanciar un objeto.

```typescript
class MiClase {
  static propiedadEstatica = "propiedad estatica";
  static mostrar() {
    return "Hola desde la clase";
  }
}
console.log(MiClase.propiedadEstatica);
console.log(MiClase.mostrar());
```

```typescript
class Constantes {
    static readonly PI = 3.14;
}

console.log(Constantes.PI) // 3.14
```

estático y solo lectura

```typescript
class Punto {
  constructor(private _x: number, private _y: number) {}
  get x() {
    return this._x;
  }
  set x(value: number) {
    this._x = value;
  }
  get y() {
    return this._y;
  }
  set y(value: number) {
    this._y = value;
  }
  static origen() {
    return new Punto(2, 7);
  }
}

let p = Punto.origen();
console.log(p.x);
```

Método que llama al constructor de la clase.