Proyecto Préstamo Express con TypeScript

Antonio ÁvilaDelgado

2º ASIR A

íNDICE

.

INTRODUCCIÓN

En este documento vamos a explicar, paso a paso, el montaje de un proyecto typescript en el cual utilizaremos **clases**, en concreto una superclase, de la cual se extenderán otra clase más, vía **herencia**. Y además usaremos 2 **colecciones** para establecer una relación **1:N**. Incluiremos un casos de **polimorfismo** y de **sobreescritura** de método (overwriting). Finalmente incluiremos un menú **CRUD**con distintas opciones.

PASOS

Gracias al archivo **entradaTeclado.ts**podemos introducir información.

src > view > TS entradaTeclado.ts > leerTeclado

```typescript
import readline from 'readline'
let readlineI: readline.Interface

let leeLinea = (prompt: string) =>  {
    readlineI = readline.createInterface({
        input: process.stdin,
        output: process.stdout,
    })
    return new Promise<string>( (resuelta: any, rechazada: any) => {
        readlineI.question(`${prompt}: `, (cadenaEntrada: string) => {
            resuelta (cadenaEntrada)
        }
        )
    })
}
export let leerTeclado = async (prompt: string) => {
    let valor: string
    valor = await leeLinea(prompt)
    readlineI.close()
    return valor
}
```

Después el archivo **menu.ts**(código que importa datos vía teclado y los aplica al menú CRUD)

src > auxiliar > TS menu.ts > ...

```ts
import { leerTeclado } from './lecturaTeclado'

export const menu_AAD = async () => {
    let n: number
    console.log('\n')
    console.log('0.- Establecer conexión a BD (ATLAS en nube vs LOCAL)')
    console.log('1.- CREAR un nuevo ORDENADOR')
    console.log('2.- CREAR un nuevo CLIENTE')
    console.log('3.- GUARDAR lo creado en BD')
    console.log('4.- RECUPERAR (a memoria) documentos de la BD')
    console.log('5.- ASIGNACIÓN DE UN ORDENADOR A CLIENTE(S)')
    console.log('6.- Calcular los COSTES FINALES del la flota (inventario) de ordenadores')
    console.log('7.- LISTA DE TODOS LOS CLIENTES')
    console.log('8.- BORRAR de la BD')
    console.log('9.- CALCULAR SUMA DE TODAS LAS GANANCIAS ASOCIADAS A CADA ORDENADOR ')
    console.log('10.- Calcular la GANANCIA TOTAL de la flota de ordenadores (desde fecha dada)')
    console.log('100.- SALIR')
    n = parseInt( await leerTeclado('--OPCIÓN--') )
    return n
}
```

En la carpeta **classes**, ponemos los siguientes archivos:

- **portatil.ts** (que será la superclase)
- **sobremesa.ts** (que extenderá de la superclase vía herencia)
- **cliente.ts** (que será la otra superclase)
- **clientejuridico.ts** (que extenderá de la superclase vía herencia)

Veamos los aspectos más destacables.

El archivo **automovil.ts**(que será la superclase), contiene las siguientes líneas de las que cabe destacar, como ejemplos reseñables:

> classes > TS portatil.ts > Portatil

```ts
import { Cliente } from './cliente'
export class Portatil {
  private _numSerie: string;
  protected _precioFabrica: number; // para acceder en la subclase
  private _velocidadCPU: number;
  private _cantidadRAM: number;
  private _capacidadHDD: number;
  private _instaladoSO: boolean;
  private _cliente: Array<Cliente>;
```

Como se ve en la imagen, debido a que tenemos que establecer una relación 1:N con la colección cliente, primero tenemos que importar desde **Cliente**, y después, en el encapsulamiento, incluir el “array” de clientes.

```
private _velocidadCPU: number;
```

Se usa “private” (encapsulamiento), lo cual nos obliga a crear un método para poder acceder:

```
get velocidadCPU() {
  return this._velocidadCPU;
}
```

Y con los distintos parámetros, creamos el constructor, que utilizaremos posteriormente:

```
constructor(
  numSerie: string,
  precioFabrica: number,
  velocidadCPU: number,
  cantidadRAM: number,
  capacidadHDD: number,
  instaladoSO: boolean
  ) {
  this._numSerie = numSerie;
  this._precioFabrica = precioFabrica;
  this._velocidadCPU = velocidadCPU;
  this._cantidadRAM = cantidadRAM;
  this._capacidadHDD = capacidadHDD;
  this._instaladoSO = instaladoSO;
  this._cliente = new Array<Cliente>();
}
```

Gracias a todo lo anterior, podemos crear **precio(),** haciendo cálculos dentro de la propia clase.

```
precio(): number { //hay clientes que quieren el ordenador sin S.O, ofrecemos un descuento del 20%
  let precio: number;
  precio = this._precioFabrica;
  if (this._instaladoSO == false) {
    precio -= 0.2 * precio;
  }
  return precio;
}
```

Y también método **completo()**

```
completo() {
  return `\nNúmero de serie: ${this._numSerie},
  Precio base: ${this._precioFabrica},
  GHz de la CPU: ${this._velocidadCPU},
  Número de GB de la RAM: ${this._cantidadRAM},
  Capacidad en TB del HDD: ${this._capacidadHDD},
  ¿Instalado el S.O.?: ${this._instaladoSO}`;
}
```

E incluso el método **gananciaTotal(),** y todo aprovechando la potencia de la clase

```
public gananciaTotal(){
  let gananciaT: number = 0
  for (let cliente of this._cliente){
    gananciaT += cliente.calcularGanancia()
  }
  return gananciaT
```

Además hemos aprovechado para añadir restricciones, dentro de la propia clase.

```
  set velocidadCPU(_velocidadCPU: number){

    if (this._velocidadCPU < 2){
        throw "\nVelocidad CPU insuficiente, debe ser al menos 2 GHz"
    }
    this._velocidadCPU = _velocidadCPU
}
set cantidadRAM(_cantidadRAM: number){

  if (this._cantidadRAM < 8){
      throw "\nCantidad de RAM pequeña, debe ser al menos 8 GB"
  }
  this._cantidadRAM = _cantidadRAM
}
set capacidadHDD(_capacidadHDD: number){

  if (this._capacidadHDD < 1){
      throw "\nCapacidad de Disco Duro escasa, debe ser al menos 1 TB"
  }
  this._capacidadHDD = _capacidadHDD
}
```

Y también hemos aprovechado en el **Schema**, para añadir otras restricciones, entre otras

```
import {Schema, model } from 'mongoose'
const clienteSchema = new Schema({
    _numIdentidad: {
        type: String,
        unique: true  //índice único
    },
```

Entre las opciones de menú más elaboradas, caben destacar:

a)La **opción 5**

```
console.log("\nHA ENTRADO EN OPCIÓN 5, ASIGNACIÓN DE UN ORDENADOR A CLIENTE(S)")
    let dOrdenador: tPort2
    let tmpPort: Portatil = new Portatil("", 0, 0, 0, 0, true)
    let tmpPersonal: Cliente
    await db.conectarBD()
    numSerie  = await leerTeclado('\nIntroduzca NÚMERO de SERIE del ordenador para asignar a cliente')
    let query4: any =  await Maquinas.findOne( {_numSerie: numSerie} )
    dOrdenador = query4
    // Creamos el objeto para el ordenador
    if (dOrdenador._tipoOrdenador == "PORT"){
      tmpPort = new Portatil(
                              dOrdenador._numSerie,
                              dOrdenador._precioFabrica,
                              dOrdenador._velocidadCPU,
                              dOrdenador._cantidadRAM,
                              dOrdenador._capacidadHDD,
                              dOrdenador._instaladoSO,
                              )
    }else{
      tmpPort = new Sobremesa(
                              dOrdenador._numSerie,
                              dOrdenador._precioFabrica,
                              dOrdenador._velocidadCPU,
                              dOrdenador._cantidadRAM,
                              dOrdenador._capacidadHDD,
                              dOrdenador._instaladoSO,
                              dOrdenador._tipoRefrig
                              )
    }
```

```
    // Leemos los clientes concretos, montamos los objetos correspondientes al número de serie
    // y asignamos los clientes al ordenador

    let v = await leerTeclado('\nCantidad de clientes que quiere asignar a un ordenador')
    let registro: number = parseInt(v);
    for(let i = 0; i < registro; i ++){
        numIdentidad  = await leerTeclado('\nIntroduzca DNI/CIF del cliente para asignar a dicha número serie del ordenador')
        let query3: any = await Humanos.findOne( {_numIdentidad: numIdentidad} )
        if (query3._tipoCliente == "PER"){
            tmpPersonal = new Cliente(query3._numIdentidad,query3._nombre, query3._pagoPrestamo,
                dOrdenador._numSerie, query3._fechaPrestado)
        }else {
            tmpPersonal = new Clientejuridico(query3._numIdentidad,query3._nombre, query3._pagoPrestamo,
                dOrdenador._numSerie, query3._idioma, query3._fechaPrestado)
        }

        tmpPort.incluirPersona(tmpPersonal)          }
```

```
    // Una vez montado el objeto ordenador con sus clientes hacemos persistenes los cambios en la BD.

    let personales2 = tmpPort.conseguirPersonas()
    for (let personal of personales2){
        await Humanos.findOneAndUpdate(
          { _numIdentidad: personal.numIdentidad },
          {
              _numseriePrestado: personal.numseriePrestado,
          },
          {
              runValidators: true // para que se ejecuten las validaciones del Schema
          }
      )
      .then(() => console.log('Modificado Correctamente') )
      .catch( (err: any) => console.log('Error: '+err)) // concatenando con cadena muestra mensaje
    }
    await db.desconectarBD()
    console.log("\nEL CAMBIO DE ASIGNACIÓN DE ORDENADOR SE HA REALIZADO CON ÉXITO")

break
```

b)La **opción 9**

```
console.log("\nHA ENTRADO EN OPCION 9, CALCULAR SUMA DE TODAS LAS GANANCIAS ASOCIADAS A CADA ORDENADOR")
  let gananciaOrdenador = async () => {
    await db.conectarBD()
    let dOrdenador: tPort2
    let tmpOrdenadores: Portatil = new Portatil("", 0, 0, 0, 0, true)
    let tmpPersona: Cliente
    let query: any =  await Maquinas.find( {} )
    let gananciaTotal: number
    for (dOrdenador of query){
        gananciaTotal = 0
     if (dOrdenador._tipoOrdenador == "PORT"){
       tmpOrdenadores = new Portatil(
           dOrdenador._numSerie,
           dOrdenador._precioFabrica,
           dOrdenador._velocidadCPU,
           dOrdenador._cantidadRAM,
           dOrdenador._capacidadHDD,
           dOrdenador._instaladoSO,
           )
         }else {
           tmpOrdenadores = new Sobremesa(
           dOrdenador._numSerie,
           dOrdenador._precioFabrica,
           dOrdenador._velocidadCPU,
           dOrdenador._cantidadRAM,
           dOrdenador._capacidadHDD,
           dOrdenador._instaladoSO,
           dOrdenador._tipoRefrig)
          }
```

```
      let query2: any = await Humanos.find( {_numseriePrestado: dOrdenador._numSerie} )

      for (let dPersona of query2){

        if (dPersona._tipoCliente == "PER"){
            tmpPersona = new Cliente(
            dPersona._numIdentidad,
            dPersona._nombre,
            dPersona._pagoPrestamo,
            dPersona._numseriePrestado,
            dPersona._fechaPrestado
            )
        }else {
          tmpPersona = new Clientejuridico(
            dPersona._numIdentidad,
            dPersona._nombre,
            dPersona._pagoPrestamo,
            dPersona._numseriePrestado,
            dPersona._idioma,
            dPersona._fechaPrestado)
        }
          gananciaTotal += tmpPersona.calcularGanancia()
          tmpOrdenadores.incluirPersona(tmpPersona)                    }
    // Estamos pidiendo el valor al método de la clase. NO se calcula fuera de la clase
    //Aquí se usan 2 métodos (de dentro de la clase) a la vez
        console.log(tmpOrdenadores.abreviado()+'GANANCIA total gracias a este ORDENADOR: '+tmpOrdenadores.gananciaTotal())
  }

  await db.desconectarBD()
}
```

Finalmente concluimos

a) Con dos ejemplos de "**sobreescritura**" de método

```
// sobre escribimos los dos métodos de abajo
precio(): number {// refrigeración líquida tiene un sobrecoste del 10%
  let precio: number;
  precio = super.precio();
  if (this._tipoRefrig == 'líquida') {
    precio += 0.1 * precio;
  }
  return precio;
}

completo(){
  let resultado: string
  resultado = `\n${super.completo()}, Refrigeración: ${this._tipoRefrig} `
  return resultado
}
```

b) y ejemplos de **polimorfismo**

```
for (let a of portatiles) {
  // valores comunes ->
  dSchemaPort._numSerie = dSchemaPC._numSerie = a.numSerie
  dSchemaPort._precioFabrica = dSchemaPC._precioFabrica = a.precioFabrica
  dSchemaPort._velocidadCPU = dSchemaPC._velocidadCPU = a.velocidadCPU
  dSchemaPort._cantidadRAM = dSchemaPC._cantidadRAM = a.cantidadRAM
  dSchemaPort._capacidadHDD = dSchemaPC._capacidadHDD = a.capacidadHDD
  dSchemaPort._instaladoSO = dSchemaPC._instaladoSO = a.instaladoSO
  // Valores propios
  // Hay que preguntar primero por las subclases
  if (a instanceof Sobremesa) {
    dSchemaPC._tipoOrdenador = "PC"
    dSchemaPC._tipoRefrig = a.tipoRefrig
    aSchema = new Maquinas(dSchemaPC)
  } else if (a instanceof Portatil) {
    dSchemaPort._tipoOrdenador = "PORT"
    aSchema = new Maquinas(dSchemaPort)
  }
```