Laboratorio III
TypeScript (parte 2)

Clase 02

Maximiliano Neiner

Temas a Tratar

- POO
- Namespaces

Clases (1/3)

- Los miembros de una clase poseen modificadores de visibilidad
  - public (defecto), protected y private.
- Los miembros de una clase pueden ser estáticos y no estáticos
  - Con el modificador ***static***, se marcan los miembros estáticos de una clase.
- Los constructores se definen utilizando la palabra reservada '***constructor***'.
- En TypeScript no existe la herencia múltiple.

Clases (2/3)

```
class Auto{

    public color : string;
    private precio : number;

    public GetPrecio() : number{
     return this.precio;
    }

    public constructor(color:string, precio:number)
    {
        this.color = color;
        this.precio = precio;
    }
}
```

Clases (3/3)

```
class Auto{

    private precio : number;

    public get Precio() : number{
      return this.precio;
    }

    public set Precio(value : number) {
      this.precio = value;
    }
}
```

Herencia

```
class Vehiculo{

    protected marca : string;

    public constructor(marca:string){
     this.marca = marca;
    }

}
class Auto extends Vehiculo{

    public constructor(marca : string) {
        super(marca);
        this.color = color;
    }
}
```

Interfaces

```
interface IAutoBase{

    GetColor() : string;

    SetColor(color:string) : void;
}

class Auto implements IAutoBase{
    //…….
    public GetColor() : string{
        return this.color;
    }
    public SetColor(color : string) : void{
     this.color = color;
    }
}
```

Clases Abstractas

```
abstract class Vehiculo{

    protected marca : string;

    public abstract Acelerar():void;

}

class Auto extends Vehiculo{
    //…….
    public Acelerar() : void{
      console.log("Acelerando…");
    }
}
```

Generics (1/2)

```
function Generica<T>(param:T) : T {

    console.log(typeof(param));

    return param;
}

let retStrring : string = Generica<string>("hola");

console.log(retStrring);

retStrring = Generica("mundo");

console.log(retStrring);
```

```
let autito = new Auto("ROJO",125000,"FERRARI");

let retAuto : Auto = Generica<Auto>(autito);

console.log(retAuto.Mostrar());

retAuto = Generica(new Auto("AMARILLO",200000,"SEAT"));

console.log(retAuto.Mostrar());
```

Generics (2/2)

```typescript
interface IGenerica<T>{

    param : T;

    Metodo() : T;

    OtroMetodo(param : T) : void;
}

class ClaseGenerica<T, U> {

    public paramUno : T;
    public paramDos : U;

    constructor(uno : T, dos : U) {
        this.paramUno = uno;
        this.paramDos = dos;
    }
}

let obj : ClaseGenerica<string, number> = new ClaseGenerica<string, number>("cadena", 10);

let obj2 : ClaseGenerica<boolean, string> = new ClaseGenerica(true, "otra cadena");
```

Temas a Tratar

- POO
- Namespaces

Namespaces (1/2)

- Concepto similar al que utiliza C#.
- Agrupaciones lógicas de elementos.
- Separación física de elementos de un mismo proyecto.
- Se pueden incluir clases, funciones, variables y otros namespaces.

Namespaces (2/2)

- Para poder acceder a los miembros de un namespace (por fuera del archivo actual) se debe agregar la palabra reservada ***export***.

```
namespace Entidades{

    export class Auto {

        public constructor(marca : string) {
            super(marca);
            this.color = color;
        }
    }
}
```

Ejercitación