TypeScript - Clases

Objetivos: Crear clases con propiedades y métodos con tipos, Añadir modificadores de acceso a los miembros de la clase.

Ejercicio 5.1

Dado el siguiente codigo:

```
class MC {
  greet(event = 'party') {
    return `Bienvenido al ${event}`;
  }
}

const mc = new MC();
console.log('[Ejercicio 5.1]', mc.greet('espectaculo'));
```

1. Añadir tipo de parámetro de forma explícita en método 'greet'
2. Agregar el tipo de retorno explicito al metodo greet

Ejercicio 5.2

Dado el siguiente codigo:

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
}

const jane = new Person('Juan', 31);

console.log('[Ejercicio 5.2]', `El nombre de la nueva persona es ${jane.name}.`);
```

1. Añadir tipos de parámetros explícitos al constructor
2. Agregue parámetros con tipos para almacenar valores

Ejercicio 5.3

Dado el siguiente codigo:

```
class Employee {
  title: string;
  salary: number;
  constructor(title: string, salary: number) {
```

© 2019 SEA, CC BY-SA license

```
    this.title = title;
    this.salary = salary;
  }
}

const employee = new Employee('Ingeniero', 100000);

console.log('[Ejercicio 5.3]', `El titulo del nuevo empleado es ${employee.title} y gana
```

1 Hacer que las propiedades de title y salary estén explícitamente disponibles públicamente

2 Reduzca la clase a tres líneas de código manteniendo la funcionalidad

Ejercicio 5.4

Dado el siguiente codigo:

```
class Animal {
  constructor(name) { }
  move(meters) {
    console.log(`${this.name} se movio ${meters}m.`);
  }
}

class Snake {
  move(meters) {
    console.log('Deslizandose...');
    // debe invocar al metodo `move` del padre, con un deslizamiento predeterminado
    // de 5 metros
  }
}

class Pony {
  move(meters) {
    console.log('Galopando...');
    // debe invocar al metodo `move` del padre con un galope predeterminado
    // de 60 metros
  }
}

// La clase Animal no debe ser instanciable.
// Eliminar o comentar una vez que se logra el error deseado.
const andrew = new Animal("Andrew el Animal");
andrew.move(5);

const sammy = new Snake("Sammy la serpiente");
sammy.move();
console.log(sammy.name); // debe devolver error

const pokey = new Pony("Pokey el pony");
pokey.move(34);
console.log(pokey.name); // Should devolver error
```

 © 2019 SEA, CC BY-SA license

1 Añadir tipos

2 Hacer que la clase Snake herede de Animal

2 Hacer que la clase Pony herede Animal

2 Hacer que el miembro del nombre no pueda ser accedido públicamente

Ejercicio 5.5

Dado el siguiente codigo:

```
class Furniture {
  constructor(manufacturer: string = 'IKEA') { }
}

class Desk extends Furniture {
  kind() {
    console.log('[Ejercicio 5.5]', `Este es un escritorio hecho por ${this.manufactu
  }
}

class Chair extends Furniture {
  kind() {
    console.log('[Ejercicio 5.5]', `Esta es una silla hecha por ${this.manufacturer}
  }
}

const desk = new Desk();
desk.kind();
desk.manufacturer; // debe devolver error

const chair = new Chair();
chair.kind();
chair.manufacturer; // debe devolver error
}
```

1 Hacer que solo las clases Desk y Chair puedan ver el miembro del fabricante

Ejercicio 5.6

Dado el siguiente codigo:

```
class Student {
  public school: string = 'Harry Herpson High School';
  constructor(private name: string) { };
  introduction() {
    console.log('[Ejercicio 5.6]', `Hola, mi nombre es ${this.name} y asisto a ${Stu
  }
}

const student = new Student('Morty');
console.log(Student.school);
student.introduction();
```

© 2019 SEA, CC BY-SA license

1 Elimine el error sin cambiar las referencias a ‘Student.school‘

 © 2019 SEA, CC BY-SA license