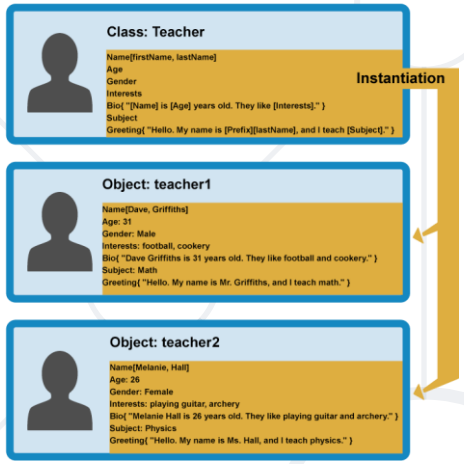
TypeScript OOP

Classes, inheritance, abstraction



SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

Software University

http://softuni.bg

Table of Content

Have a Question?

SoftUni Foundation

sli.do

#tbd

TypeScript Classes

Definition

- **Classes** in TypeScript can contain:
  - **Data**, defined by its **properties**
  - Who can use the data - **access modifiers**
  - Some **actions** by using **methods**
- **One class** may have **many heirs** – **inheritance**
- **Abstract classes** cannot be instantiated directly. They are the **ancestor** class which starts the **inheritance chain**

Overview

```
class Dog {
    private name: string;
    private age: number;

    constructor(n: string, a: number) {
        this.name = n;
        this.age = a;
    }

    bark() {
        return `${this.name} woofed friendly`;
    }
}

let tommy = new Dog('Tommy', 6);

console.log(tommy); //Dog { name: 'Tommy', age: 6 }
console.log(tommy.bark()); //Tommy woofed friendly
```

Class initialization

Class properties

Class constructor

Class method

Breakdown: Properties

SoftUni Foundation

- The **properties** in TypeScript are used to **store data**
  - They are defined **before** the constructor in the **body** of the class
  - The **data is passed** to them **afterwards**

```
class ContactList {
    private name: string;
    private email: string;
    private phone: number;
}
```

Property declarations

Breakdown: Constructor

SoftUni Foundation

- The **constructor** is used to give properties **values**
  - Each **class** can have only **one constructor**
  - The constructor creates **new object** with the defined properties

```
class ContactList {
    //property declarations
    constructor(n: string, e: string, p: number) {
        this.name = n;
        this.email = e;
        this.phone = p;
    }
}
```

Breakdown: Methods

- The **methods** are used to define functionalities
  - Each **class** can have **lots of methods**
  - Generally speaking, each **method** should do **one thing** only

```
class ContactList {
    //property declarations
    //constructor
  call() {
    return `Calling Mr. ${this.name}`
  }
  showContact() {
    return `Name: ${this.name} Email: ${this.email} Number: ${this.phone}`
  }
}
```

Methods

Access modifiers

SoftUni Foundation

- Unlike JavaScript, TypeScript has **access modifiers**
- Used to **define** who can **use** the class elements
- **Types** of access modifiers:
  - Public
  - Private
  - Readonly
  - Protected

Public

- By **default** each element is defined **as public**
- Gives **access** to the element
- Not only **properties** may be public, but **constructors** as well

```
class Zoo {
    public type: string;
    public name: string;

    public constructor(t: string, n: string) {
        this.type = t;
        this.name = n;
    }
}
```

Private

SoftUni Foundation

- Element marked as **private** cannot be accessed **outside** the declaration

```
class Zoo {
    private type: string;
    private name: string;

    constructor(t: string, n: string) {
        this.type = t;
        this.name = n;
    }
}
let animal = new Zoo('bear', 'Martha');
console.log(animal.name); //Error: name is private.
```

Readonly

- **Readonly** protects the value from being **modified**
- No unexpected data mutation

```
class Zoo {
    readonly name: string;

    constructor(n: string) {
        this.name = n;
    }
}
let animal = new Zoo('Martha');
animal.name = 'Thomas'; //Error: name is read-only.
```

Protected

- Element marked as **protected** can be accessed **only** within the **declaration class** and **the subclasses**

```
class Zoo {
    protected name: string;
    constructor(n: string) { this.name = n; }
}
class Bear extends Zoo {
    private color: string;
    constructor (name, c: string) {
        super(name);
        this.color = c;
    }
}
let martha = new Bear('Martha', 'Brown');
```

Inheritance

SoftUni Foundation

- Used to **extend** existing classes to **new ones**
- To do so we use the **extend** key word
- The "basic" class is often called **superclass** and the extended - **subclasses**
- To inherit the superclass's **constructor** to the subclass we use the keyword **super**

Example of inheritance

```
class Company {
    public name: string;
    constructor(n: string) { this.name = n; }
}
class Department extends Company {
    private depName: string;
    constructor(name, dN) {
        super(name);
        this.depName = dN;
    }
}
class Employee extends Department {
     //Some code logic
}
```

Class Department inherits the Company class

Class Employee inherits the Department class

Inheritance

SoftUni Foundation

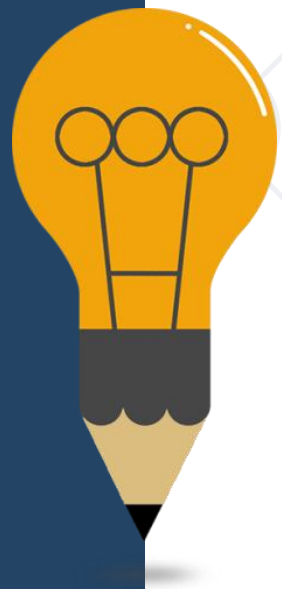
- Not only **properties** might be inherited, but **methods** as well.

```
class Vehicle {
    public color: string;
    constructor(c: string) { this.color = c; }
    showColor() { return `The car is ${this.color}`; }
}
class PassengerCar extends Vehicle {
    public model: string;
    constructor(color, m: string) {
        super(color);
        this.model = m;
    }
    details() {
        return `${super.showColor()} and is ${this.model}`
    }
}
```

Accessors

SoftUni Foundation

- In order to use accessors your compiler output should be set to **ES6** or higher
- **Get** and **Set**
  - Get method comes when you want to **access** any class property
  - Set method comes when you want to **change** any class property



Example of accessors

SoftUni Foundation

```
const fullNameMaxLength = 10;

class Employee {
    private _fullName: string;

    get fullName(): string {
        return this._fullName;
    }

    set fullName(newName: string) {
        if (newName && newName.length > fullNameMaxLength) {
            throw new Error("fullName has a max length of " + fullNameMaxLength);
        }

        this._fullName = newName;
    }
}
```

Abstract class

- Defined by keyword **abstract**
- They are **superclasses** but **cannot** be **instantiated directly**
- Methods inside abstract classes and marked as such **do not contain implementations** but **must be** implemented **in derived classes**

Example of abstract class

SoftUni Foundation

```
abstract class Department {
    public depName: string;
    constructor(n: string) { this.depName = n; }
    abstract sayHello(): void;
}
class Engineering extends Department {
    public employee: string;
    constructor (depName: string, e:string) {
        super(depName)
        this.employee = e;
    }
    sayHello() {
        return `${this.employee} of ${this.depName} department says hi!`;
    }
}
let dep = new Department('Test') //Cannot create instance of abstract class
```

Static properties

- Defined by keyword **static**
- The **property** belongs to the class itself, so it **cannot be accessed** outside of the class
- We can only access the properties directly **by referencing** the class itself

Example of abstract class

SoftUni Foundation

```
class Manufacturing {
    public maker: string;
    public model: string;
    public static vehiclesCount = 0;

    constructor(maker: string, model: string, ) {
        this.maker = maker;
        this.model = model;
    }
    createVehicle() {
        Manufacturing.vehiclesCount++;
        return `Created cars: ${Manufacturing.vehiclesCount} of
        ${this.maker} ${this.model}`;
    }
}
```

Live Exercises

Summary

- Classes in TypeScript consist of
  - Properties
  - Constructor
  - Methods
- You can restrict or allow access to properties by using access modifiers
- Using get and set methods

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SoftwareGroup
doing it right

NETPEAK

SmartIT

INDEAVR
Serving the high achievers

INFRAGISTICS®

STEMO®
Computer Systems & Software

SUPERHOSTING.BG

SoftUni Organizational Partners

SoftUni Foundation

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

Trainings @ Software University (SoftUni)

SoftUni Foundation

- Software University - High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg