# Exercises: Namespaces and modules

Problems for exercises and homework for the ["TypeScript Advanced" course @ Software University](https://softuni.bg/trainings/2696/typescript-advanced-december-2019).

## Geolocation

Create a class **Geolocation** that gets **latitude** and **longitude** from the constructor. Add a **showCoordinates()** method that opens a Google Maps search in your **default browser** and displays the location with the given coordinates. Export that class to a different file.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| let locator = new Geolocation(42.698334, 23,319941);  locator.showCoordinates() | 80071322_2733713916686369_962997668525637632_n.png |

### Hint:

### Look for opn node package.

## Greeter

Create a namespace **Greeter**, in one file, that exports a generic interface **Greeting**. The interface must support the following **two** **declarations**:

* introduction()
* **sayGoodbye(name)** – the name is of the generic type.

In a new file create class **Person** which implements the **Greeter** interface and receives two **private** properties from the constructor: **name** (a string) and **age** (a number) and add the following logic to the methods:

* **introduction()** – returns a string in the following format: "**My name is {name} and I am {age} years old."**
* **sayGoodbye(name)** – returns a string in the following format: "**Dear {name}, it was a pleasure meeting you!"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| let p = new Person('Ivan Ivanov', 25);  console.log(p.introduction());  console.log(p.sayGoodbye('Petar Petrov')); | My name is Ivan Ivanov and i am 25 years old  Dear Petar Petrov, it was a pleasure meeting you! |

## Deliveries

Create a namespace **FoodAndBeverages**, in one file, that exports an interface **Delivery**, which supports the following **declarations**:

* **newCustomer(customerName: string, visited: boolean)**
* **visitCustomer(customerName: string)** - this method returns a string
* **showCustomers()** - this method returns a string

In a new file create a class **Courier** that implements Delivery and has one property of its own:

* **protected** property **placesToVisit** which is set through the constructor and is array of objects.

Make the following three methods:

* **newCustomer(customerName: string, visited: boolean)** - the **visited** by default is **false**. Check if the **customerName** is already a customer of yours, if so throw a new error: "**{customerName} is already a customer of yours!"**. Otherwise push an object with key **customerName** and value – **visited** to the **placesToVisit** array and return a message: "**{customerName} just became your client."**
* **visitCustomer(customerName: string)** – check if the customer is client of yours, if not throw a new error: "**{clientName} is not your customer"**. Otherwise swap the value of **visited to true**.
* **showCustomers()** – print all the customers in the following format:

"{customerName} -> visited

{customerName} -> visited

… "

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| let courier = new Courier([{ customerName: 'DHL', visited: false }]);  courier.newCustomer('Speedy');  courier.newCustomer('MTM');  courier.newCustomer('TipTop');  courier.visitCustomer('DHL');  courier.visitCustomer('MTM');  courier.visitCustomer('MTM');  console.log(courier.showCustomers()); | DHL -> true  Speedy -> false  MTM -> true  TipTop -> false |

## Shops

Create an **abstract class Shop** that takes two **protected** **properties**: **shopName** and **shopAddress**, from the constructor and an abstract method **showInventory()** and export that class to a different file.

In a new file create a class **PCStore** that extends **Shop** and has three properties of its own:

* **protected** property budget (number) - set through the constructor
* **public** property **pcParts** (object) which is initially set to empty object
* **public** property **pcCollection** which is initially set to an empty array of objects

Make the following four methods:

* **buyParts(part: string, price: number)** – check if you have enough money to buy the part. If so also check if the part is already in the **pcParts** object. If it is not set the part as a **key and value to 1**. Otherwise, **just increase the value with 1**. If you don’t have enough money throw new error: "**{shopName} cannot afford that part."** When you successfully buy a part decrease the budget with its price.
* **buildPC(neededParts: string[], pcName: string, pcPrice: number)** – check if you have all of the needed parts in your inventory. If so build a new PC, add it to the **pcCollection** in the following format: **{pcName, neededParts, pcPrice}** and return a message "**You have just built {pcName}”**. Make sure you decrease the quantity of all of the needed products in your inventory. If a PC part reaches 0, delete the entire key and value pairing. If you don’t have one or more part, throw new error: **“You do not have all needed parts to build this PC"**
* **sellPC(desiredPCName: string, clientBudget: number)** – check if you have the **desiredPCName** in your **pcCollection**. If so, check if the client budget is enough to pay for the PC. If it is delete the PC from the collection and increase the budget with the **pcPrice**. If the client`s budget is higher than the **pcPrice** you should return the rest of the money back and print the following message: "**You successfully bought {desiredPCName} and you get {moneyInReturn} BGN in return** **".**
* **showInventory()** – print all the information in the following format:

**"Shop name: {shopName}**

**Shop address: {shopAddress}**

**Shop budget: {budget}**

**PC parts in store: {pcPart, pcPart, pcPart...}**

**PCs in store: {pcName, pcName...}"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| let pcStore = new PCStore('AllBestPCs', 'Sofia', 10000);    pcStore.buyParts('CPU', 300);  pcStore.buyParts('CPU', 300);  pcStore.buyParts('CPU', 300);  pcStore.buyParts('CPU', 300);  pcStore.buyParts('Graphic card', 200);  pcStore.buyParts('Graphic card', 200);  pcStore.buyParts('Graphic card', 200);  pcStore.buyParts('RAM', 70);  pcStore.buyParts('RAM', 70);  pcStore.buyParts('HDD', 30);  pcStore.buyParts('HDD', 30);  pcStore.buyParts('HDD', 30);  pcStore.buyParts('HDD', 30);  pcStore.buyParts('HDD', 30);  pcStore.buyParts('SSD', 60);  pcStore.buyParts('SSD', 60);  pcStore.buyParts('SSD', 60);    pcStore.buildPC(['HDD', 'CPU', 'Graphic card', 'RAM'], 'Regular PC', 1000);  pcStore.buildPC(['HDD', 'SSD', 'CPU', 'Graphic card', 'RAM'], 'Gaming PC', 1700);    console.log(pcStore.sellPC('Regular PC', 1300));  console.log(pcStore.showInventory()); | You successfully bought Regular PC and you get 300 BGN in return  Shop name: AllBestPCs  Shop address: Sofia  Shop budget: 8730  PC parts in store: CPU, Graphic card, HDD, SSD  PCs in store: Gaming PC |