**IS 312 Web Design: TypeScript (TS) for Modern Web Application**

**HOP06: Object-Oriented Programming**

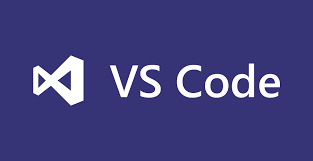
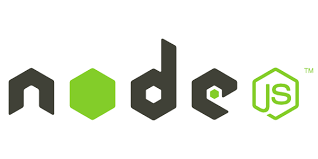
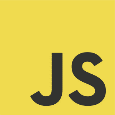
5/22/2019, Developed by Kevin Kuanting Chen, Class of 2020

5/18/2020, Revised by Amrutha Vaidyanathan, Class of 2020

9/23/2020, Revised by Kim Nguyen, Class of 2021

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* Some steps are not explained in the tutorial**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

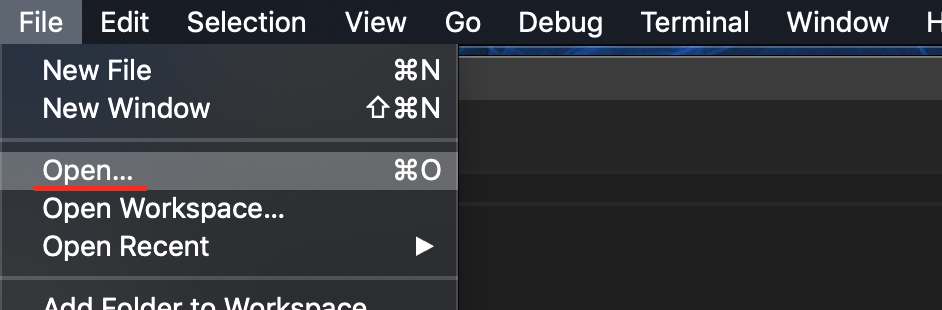
* Understand Object-Oriented programming concepts in TypeScript.

**Resources**

* [W3School Data Types](https://www.w3schools.com/js/js_datatypes.asp)
* [TypeScript Object-Oriented Concepts in a Nutshell](https://levelup.gitconnected.com/typescript-object-oriented-concepts-in-a-nutshell-cb2fdeeffe6e)

**Preparation**

1. Open the VS Code and open the repository you cloned from Github, if you have not cloned, go back to the Github repository generated when you accept the HOP assignment, read the instruction on how to clone the repository, before proceeding.



1. Open the terminal from the VSCode, check your current directory using the following command, if you are in **Module 4**, you are in the right place:

**Inheritance**

Inheritance provides the ability of a program to create a new class from an existing class. It is a mechanism which acquires the properties and behaviors of a class from another class. The class whose members are inherited is called the base class, and the class that inherits those members is called the derived/child/subclass. In child class, we can override or modify the behaviors of its parent class.

JavaScript uses functions and prototype-based inheritance, but TypeScript supports the class-based inheritance. The TypeScript uses class inheritance through the extends keyword. Super classes, (aka parent classes) allow for related objects to be grouped together so that they can inherit similar attributes.

TypeScript supports only single inheritance and multilevel inheritance. It doesn't support multiple and hybrid inheritance.

1. Create a file **InheritenceEg.ts** and type the following code.

A screen shot of a social media post

Description automatically generated

1. Complies and run the above file

>>> tsc InheritenceEg.ts

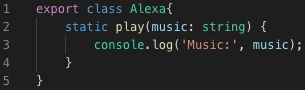
>>> node InheritenceEg.js

A close up of a sign

Description automatically generated

**Classes and Interfaces**

1. TypeScript borrows many object-oriented concepts from C# for its classes. You will see many similar class structures here. Create a file called **Alexa.ts** under the folder. The export keyword allows you to use the class in another file.



There are 2 modifiers: static and readonly.

If we want to access a property on a class without having to go through the trouble of returning an instance-object (calling the object with new), then we can mark it as static and it will be set on the class (function-object) itself.

The readonly keyword is used for class level variables and means that the value cannot be reassigned. Values that are initialized when the class is created, and if it should never be changed, specify readonly.

1. Create another file called **Vehicle.ts** under the same folder. The import keyword allows you to use exported classes from another file. Interface is used when an object being passed has a specific set of attributes.

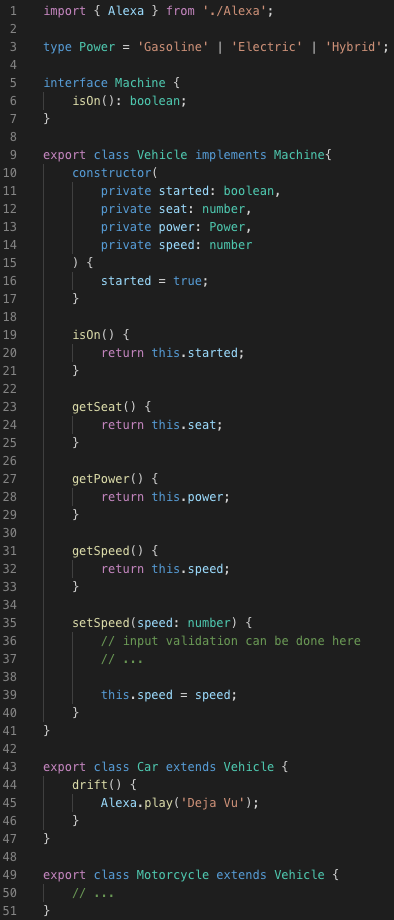
You will notice keyword private which makes a class variable/method only accessible within that class. There are other keyworks like public and protected.

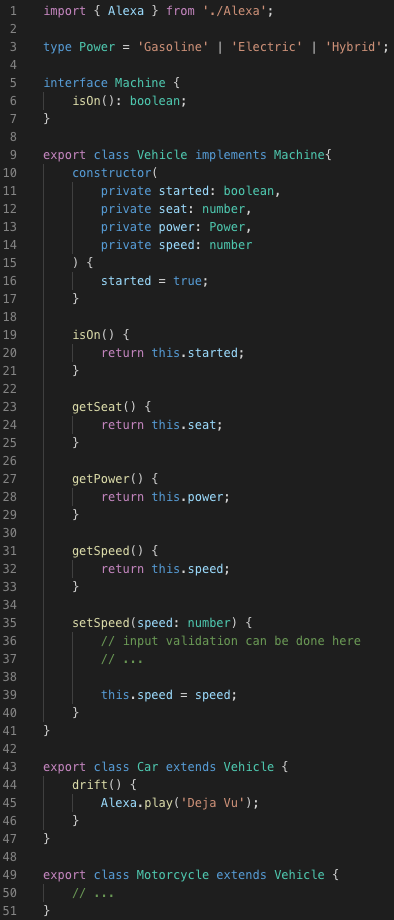
Public is used when a method or variable is public that means it can be accessed by another part our program. If there is no keyword specified on a variable or method, it means its public.

Protected means that a variable/method can only be accessed in child classes of the parent.

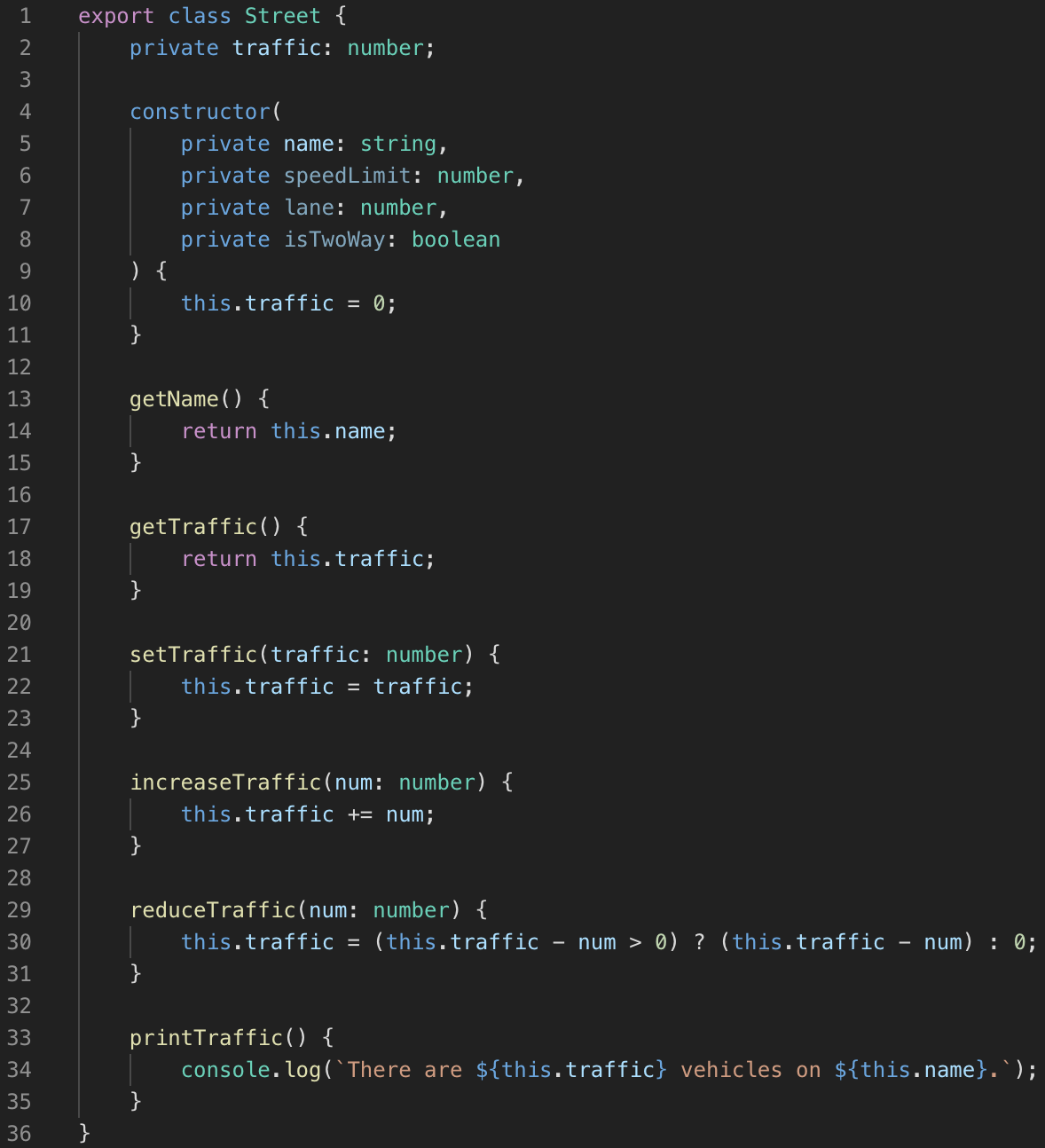
TypeScript classes have built in get and set modifiers which will trigger our getters and setters whenever a property is trying to be accessed.

Interface is used when an object being passed has a specific set of attributes.

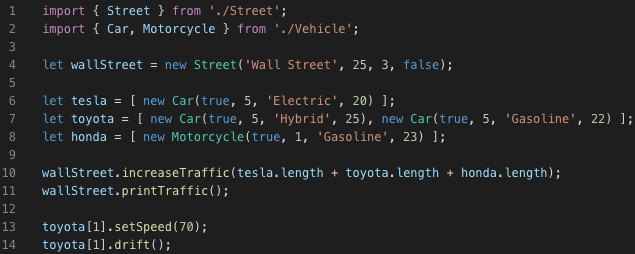




1. Create one more file called **Street.ts**.

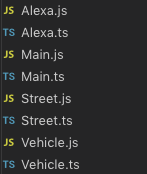


1. Finally, create a file called **Main.ts** for your main code.



1. Compile the Main.ts file.

>>> tsc Main.ts



Note: Notice that Alexa.ts, Street.ts, and Vehicle.ts are also compiled when you compile the Main.ts file. This is because you have imported classes from Street.ts and Vehicle.ts (and the imported Alexa class in Vehicle.ts) in Main.

1. Run the Main.js file.

>>> node Main.js



**Push your work to GitHub**

Run the following commands to push your work to the GitHub repository:

Open the terminal from the VSCode by hitting the control + ~ key and type the following command:

**>>> git add .**

**>>> git commit -m “Submission for Module 4 – YOUR NAME”**

**>>> git push origin master**