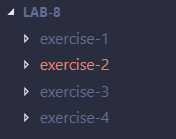
**COMP 3133 – Full Stack Development II**

**Lab 06 – TypeScript**

**Developer Note:**When working on your exercises, please create separate folder for your work. This way you won’t putting all your code in the same file, which can pollute the global name space. In short, it will prevent you from overwriting your own work and causing your code to compile incorrectly.

Organize your folder structure in this way.



**Exercise #1 – TypeScript First Start**

1. Create a folder named **Lab06\_studentID**   
   
2. Open a command prompt create a directory for **exercise-1**A black screen with yellow text

   AI-generated content may be incorrect.
3. Open Visual Studio Code and open the folder **exercise-1**A black screen with yellow text

   AI-generated content may be incorrect.
4. Run the following to install the ***TypeScript Compiler*** globally using NPM

  
A screen shot of a computer

AI-generated content may be incorrect.

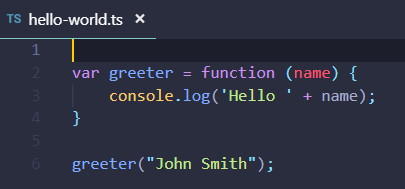
1. Once downloaded check the version of the TypeScript compiler using the following:



  
A black background with white text and purple letters

AI-generated content may be incorrect.

1. Create a TypeScript file named ***hello-world.ts*** and type the following code:



1. Execute the TypeScript compiler to compile the file into a JavaScript file.



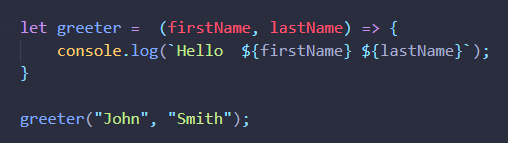
1. Type the following to open the new hello-world.js file in Visual Code.



A black screen with yellow text

AI-generated content may be incorrect.

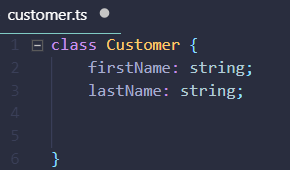
1. Rewrite this code block using TypeScript ES6 features use the let keyword, lambda (fat arrow) declaration for anonymous functions and the back tick (template literals). Change the function for two names, first and last.



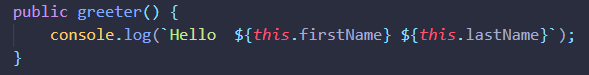
1. Repeat steps 7 & 8 and view the transpiled JavaScript code.

**Exercise #2 – Types, Classes and Objects**

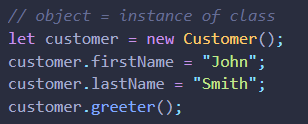
1. Open a command prompt create a directory for **exercise-2**
2. Open Visual Studio Code and open the folder **exercise-2**
3. Create a TypeScript file named ***customer.ts***. Create a class named Customer with two string typed properties.



1. In the ***Customer class***, add the ***greeter method*** and use the ***this*** keyword to output the properties.



1. In the same ***customer.ts*** file. Create a ***Customer object***, by creating an ***instance of the Customer class***. Then call the greet method to get the output.



1. Compile the ***customer.ts*** using TypeScript compiler



1. Run Node to execute the transpiled JavaScript file.



The expected output is the following:

  
  
A black screen with yellow text

AI-generated content may be incorrect.

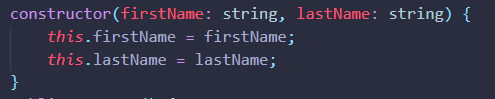
**Exercise #3 – Access Modifiers and Constructors**

1. Open a command prompt create a directory for **exercise-3**A black screen with yellow text

   AI-generated content may be incorrect.
2. Open Visual Studio Code and open the folder **exercise-3**
3. Create a ***TypeScript file*** named ***customer.ts***. Copy the contents from the previous Customer class.
4. Add ***a private keyword*** to both properties for the First and Last Name.



1. Add ***Constructor*** that will take two parameters for first and last Name. Then it will set the local properties using the ***this keyword***.



1. In the same file, change the way the Customer object instance is created to know pass in the two parameters for first and last name.



1. Compile the ***customer.ts*** using TypeScript compiler



1. Run Node to execute the transpiled JavaScript file.



The expected output is the following:

  
  
A black screen with yellow text

AI-generated content may be incorrect.

**Exercise #4 – Modules**

1. Open a command prompt create a directory for **exercise-4**
2. Open Visual Studio Code and open the folder **exercise-4**
3. Create a ***TypeScript file*** named ***customer.ts***. Copy the contents from the previous Customer class.
4. Add the ***export keyword*** to the ***Customer class*** to mark it as a module in the application.



1. Create a ***maint.ts typescript*** and import the ***Customer Module*** into the file.



1. In the same main.ts file, create the object using an instance of the Customer module and then call the greeter method.



1. Compile the ***customer.ts*** using TypeScript compiler



1. Run Node to execute the transpiled JavaScript file.



The expected output is the following:



1. Update the ***customer.ts*** to do the following:
   * Write a method named ***GetAge()*** that should return the age of the Customer and log it to the console.
   * Update the ***constructor*** to take a ***new parameter named Age and will set a private property \_age of type integer***.

Update the ***main.ts*** to do the following:

* + When creating an instance of the Customer object call the GetAge() method.  
      
    A black screen with yellow text

    AI-generated content may be incorrect.

**Submission**

* + **Screenshot(s)** of appropriate output. You may add all the screenshots in word document and upload it**. *[Do not put screenshots in project/zip file]****.*
  + Create **a zip file** of your entire project folder and submit.
  + Your **GitHub** project link.