MET CS601: Module 4 Assignment

General Rules for Homework Assignments

- You are strongly encouraged to add comments to your source code. Doing so will help your facilitator to understand your logic/approach and grade your work more accurately.
- You must work on your assignments individually. You are ***not allowed*** to copy the answers from others. However, you are encouraged to discuss approaches to the homework assignment with your facilitator
- You are expected to write your own code for all assignments. You may use an IDE or advanced text editor for your assignments, but you *must* ***not*** use any auto generated code provided by such tools or other applications. So be sure to write your own code in the editor window, don't use the WYSIWYG builder (if applicable).
- Do not use any unapproved code libraries or frameworks.
- Each assignment has a strict deadline. However, you are still allowed to submit your assignment within ***two (2) days*** after the deadline with a penalty. 15% of the credit will be deducted unless you made previous arrangements with your facilitator. Assignments submitted 2 days after the deadline will not be graded.
- When the term *lastName* is referenced in an assignment's file or folder name, please replace it with ***your*** last name.

Create a new folder/directory named **CS601_HW4_*lastName***. Place your solution(s) to the assignment requirements in this folder.

NOTE: THIS DOCUMENT CONTAINS MULTIPLE PAGES

Building a Country Management System with TypeScript

This assignment focuses on using TypeScript to create a simple Country Management System application. You'll explore interfaces, classes, inheritance, type assertions, and type predicates to create a type-safe and flexible system. Imagine you're building a program to manage information about various countries around the world. The program needs to filter through a list of various countries with unique properties and build a list of country objects of a specified type. In this application, we will build a list of only snowy countries.

Requirements

Before you begin, follow the steps in Part I to create a development environment and install the TypeScript compiler.

Part I: Create a Project Directory

a. Download the **Module4_Assignment.zip** file from Blackboard and extract the content to your local drive. You will need these starter files to complete this assignment. Verify that you have all of the following files in the **country-app** directory:

```
│   package-lock.json
│   package.json
│   tsconfig.json
│
├───public
│       index.html
│
└───src
        app.ts
```

b. Make sure you have **Node.js** installed. You can download and install Node.js by visiting this site: https://nodejs.org/en/download

c. Open a terminal (or commend prompt) and navigate to the root level of the **country-app** directory. Issue the follow command to install the TypeScript compiler and other dependencies:

```
npm install
```

d. You should now have a new folder called "**node_modules**" added to your project folder.

e. Move on to Part II.

Part II: Start Coding

1. Use your favorite IDE/Code Editor, complete the following using TypeScript.
3. Open the **app.ts** file and complete as follows:
   a. Create an interface called **ICountry** that is comprised of:
      1) **name** – a string type property to store a country name
      2) **getInfo()** – a function that accepts a parameter of HTMLElement type and returns the element.

b. Create three classes **RainCountry**, **SnowyCountry**, and **IslandCountry** that implement the **ICountry** interface.

c. Each country class has the following properties and implementations:

   1) **RainyCountry**
      a) A property called **rainLevel** that stores annual inches of rain received.
      b) The **getInfo()** receives an HTMLElement object type (e.g. <p>).
      c) Construct a string that prints the rain level. For example, the element would contain a string similar to: "Brazil has a rain level of 120.50 inches."
      d) Return the element

   2) **SnowyCountry**
      a) A property called **snowLevel** that stores annual inches of snows received.
      b) The **getInfo()** – same as RainyCountry, except it prints **snowLevel** information.
      c) Return the element

   3) **IslandCountry**
      a) A property called **landSize** that stores the total land mass.
      b) The **getInfo()** – same as RainyCountry, except it prints **landSize** information.
      c) Return the element

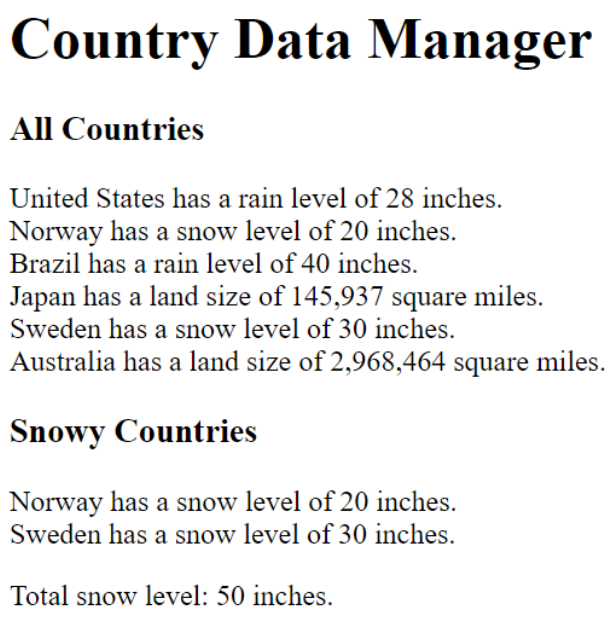
4. Prepare a list of various types of country objects as follows to test your app:

```
// Sample data
const countries: ICountry[] = [
    new RainyCountry("United States", 28),
    new SnowyCountry("Norway", 20),
    new RainyCountry("Brazil", 40),
    new IslandCountry("Japan", 145937),
    new SnowyCountry("Sweden", 30),
    new IslandCountry("Australia", 2968464)
];
```

5. Create an empty list called **snowyCountriesList** for storing only **SnowyCountry** objects.
6. Perform the following operations to build the **snowyCountriesList**:
   a. Create a function to perform the following:
      1) The function must receive a country object of type **ICountry** (e.g. `country:ICountry`).
      2) It checks the country's "**snowLevel**" property
         a) If the property is present in this object, return the country object
         b) Otherwise, return the **null** value
   b. Use your function to filter the list of countries.
   c. If the object returned by this function is a **SnowyCountry**, then add this country to the **snowyCountriesList**.

   d. Use type predicates and type assertions where required and necessary in your implementations.
   e. Create additional functions as needed (if any).

7. Construct and render the following information to the DOM (the **index.html**).
   a. Display all the country information from the **countries** list.
   b. Display all the snowy country information and the total annual snow level from the **snowyCountriesList**.



Country Data Manager

All Countries

United States has a rain level of 28 inches.
Norway has a snow level of 20 inches.
Brazil has a rain level of 40 inches.
Japan has a land size of 145,937 square miles.
Sweden has a snow level of 30 inches.
Australia has a land size of 2,968,464 square miles.

Snowy Countries

Norway has a snow level of 20 inches.
Sweden has a snow level of 30 inches.

Total snow level: 50 inches.

Part III: Compilation and Running

1. Open the command terminal and navigate to the root directory of **country-app**.
2. Compile your TypeScript using the command below:

```
C:\country-app> npx tsc
```

3. This will generate the corresponding `.js` file to the **/public** folder according to your `tsconfig.json` settings.
4. Verify that your **public** folder has the **app.js** file.
5. Run the **index.html** file using your favorite browser.

Remember that your HTML and JS files may still run normally even if there were errors (other than syntax errors) during the compilation process. The focus in this assignment is on TypeScript, so it's important that your TypeScript code compiles successfully without any errors.

Assessment/Grading

Your assignment submission will be scored by the following criteria:

1. Strict adherence to the requirements stated above: 70%
2. Code validates without errors (warnings are OK): 10%
3. Overall quality of work and effort as determined by your facilitator: 20%
    a. Includes your project/assignment also have a README.md file

It is important that your code passes validation; you should use http://validator.w3.org for assistance.

You must also validate your CSS code as well, that can be done here: http://jigsaw.w3.org/css-validator/

Submission

Export your **CS601_HW4_*lastName*** folder containing all the relevant sub-folders and files as a zip file and upload the zip file to the appropriate assignment submission area.

Note: Please DO NOT include the **node_modules** in your zip file.