**Travel-o-matic blog**

You enjoy traveling to different places, visiting events and eating good food in restaurants. You want to create a TypeScript-based system that presents all of your experiences in a web-page.

What is common to a place, event and restaurant? A **location**. Since you know about classes, you decided that your base(main) class Location should hold information about the **city, ZIP-code, address** (single line like “Kettenbrueckengasse 23”), and a**teaser image**. That base class has the function **display()** which is used for displaying the object’s properties on the screen as HTML.

A **restaurant** must also display a**telephone number, type (“chinese”, “indian”, “viennese”, …)  and a web address**. Restaurant objects **inherit their**basic properties (like ZIP-code) from the Location class. The display function must of course be updated.

Same goes for the **events** - they have their additional properties like **EventDate** (“12.10.2020”) and **EventTime** (“17:00”) and ticket price (in EUR) that also need to be displayed in addition to the base class’ location properties.

For the regular points of this CodeReview, you need to create a structure of TypeScript/JavaScript classes, their respective constructors and their display() function(s) that will display the relevant data of places, events and restaurants that you have visited.

Regular points:

* (5 points) Creation of a GitHub project (Repository name: **CFLMS-YourName-CodeReview-06**), successful “push” to the repository and correct information to CodeFactory through the learning management system (https://lms.codefactory.live/) with the GitHub link for the cloning procedure.
* (15 points) Create a **data model of Location** based on the specification above.
* (20 points) Create at least **2 location objects** by invoking a **constructor call** and save them in an **array** (class constructors usage is mandatory, do not use JSON and parsing).
* (10 points) Add the **display() function** to the Location class that can display Location objects as an HTML string. Use Bootstrap to solve design and responsiveness issues in an easy manner.
* (10 points) Use **looping functionality** to display the objects saved in the array on the screen.
* (20 points) Create the classes **Restaurant and Events** with their **respective properties**. Invoke constructors, and save resulting objects in the same array used for the location objects. Check the rendering (i.e. how you display it).
* (10 points) Create a **display() function** for the classes Restaurant and Event respectively. display() is capable of displaying relevant information of its class as an HTML. Again, use Bootstrap to solve design and responsiveness.
* (10 points) Assure that rendering of the array is done in the following manner: for **small screens**, you see the **teaser of one object (without the image)** in a row. On **medium screens**, you see **two objects in a row (with image)**. On**large screens**, you see **four objects**in a row (**with image**).

**Bonus Points**:

* (10 points) You have decided to add additional**date/time** for every location that you visited. You add that information “**Created: 24.10.2020 12:45**” to the respective teaser displays.
* (10 points)  You want to use the newly added entry to add two additional pages to your system **index-asc.html  and index-desc.html** Those pages are sorting the array of locations (places, events, restaurant) ascending/descending based on the “Created” date/time property.

**Summary:** Create 3 different classes. The main class is the Location class, followed by a Restaurant class and Event class. Both the Restaurant class and Event class will inherit the basic properties from the Location class and will have extra properties specific only to themselves. Each class will have a display() function, displaying their specific content. You need to create at least 2 objects per class. All objects created should be placed in an array, that should be looped through to display each object on your page(s).