WEB222 Assignment 2

Submission Deadline:

Sunday, February 19th, 2017 @ 11:59 PM

Assessment Weight:

5% of your final course Grade

Objective:

Practice Array Operations, Custom Objects & Working with Data.

Specification:

Write a custom object called customerDB that will serve as a object "database" for a strictly formatted set of sample data. To begin, access the following file and copy/paste the contents into scratchpad:

Shared file: assignment2.js

Be sure to save this file as "assignment2.js".

This file includes two blocks of code (you will write your customerDB object between them):

- The block labeled **TEST DATA** at the bottom is currently commented out, but you must uncomment it to test your solution when you're ready. After running your code with the TEST Data uncommented, the output in the console should look like this sample

- The second block of code at the top of the file (labeled **ALL DATA**) is an array of objects called "allData". All the objects in this array follow the same format:

{ type: "customer" */* or "store" or "address"*/*, data: { */* object with properties related to the "type" */* }

The "**type**" is one of "**customer**", "**store**", or "**address**", and the "**data**" is an object with properties that belong to the "store" or "customer" or "address". The properties in the "**data**" object will always have the same names and follow the following convention for each of the following "types":

type: "customer"	
customer_id	A primary key used to uniquely identify each customer.
store_id	A foreign key identifying the customer's "home store."
first_name	The customer's first name.
last_name	The customer's last name.
email	The customer's email address.
address_id	A foreign key identifying the address of the customer.
add_date	The date the customer was added to the system.

type: "store"	
store_id	A primary key that uniquely identifies the store.
address_id	A foreign key identifying the address of the store
name	The name of the store / branch.

type: "address"	
address_id	A primary key used to uniquely identify each address.
address	The street address.
city	The name of the city
province	The name of the province
postal_code	The postal code or ZIP code of the address (where applicable).

Part A: CustomerDB Object

In your assignment2.js file, underneath the "allData" array, declare an object called **CustomerDB** using "Object Literal Notation". This object will **contain** the following **properties** & **methods**

Array Properties

The following are internal arrays that will serve as the primary storage mechanism within the CustomerDB object for all of our sample data

- **customers**
  This is an array that will contain all the sample data objects that are of **type "customer".** It is initialized as an empty array (ie, []) and will be manipulated using the methods in the CustomerDB Object
- **addresses**
  This is an array that will contain all the sample data objects that are of **type "address".** It is initialized as an empty array (ie, []) and will be manipulated using the methods in the CustomerDB Object
- **stores**
  This is an array that will contain all the sample data objects that are of **type "store".** It is initialized as an empty array (ie, []) and will be manipulated using the methods in the CustomerDB Object

Main "insertData" Function (Method)

The insertData Method is the first method that will be invoked on your CustomerDB object. It is this method that takes all of the sample data and inserts it into the correct arrays (ie: "customers", "addresses", or "stores"). It takes one parameter, the **allData** array from the top of your assignment2.js file and processes it one array element at a time using the following rules:

- if type is "store", insert the "data" object into the "stores" array (**HINT**: there is a function we will write called **addStore(storeObj)** that will be perfect for this)
- if type is "customer", retrieve the "data" object and set it's "add_date" property to the current date and add it into the "customer" array (**HINT**: there is a function we will write called **addCustomer(customerObj)** that will be perfect for this)

- if type is "address", insert the "data" object into the "addresses" array (**HINT**: there is a function we will write called **addAddress(addressObj)** that will be perfect for this)

Once this method has run, your "customers" array should contain **8** "customer" data objects, your "addresses" array should contain **9** "address" data objects, and your "stores" array should contain **3** "store" data objects and your "database" is built.

Methods to work with "customer" data

The following are all methods that deal primarily with the "customers" array. Any output for these functions is meant for the web console, so whenever the term "output" is used, you may assume that we're outputting to the console with console.log(). HINT: to refer to the "customers" array from within these methods, use the "this" keyword, ie: "this.customers".

- **addCustomer (customerObj)**

  This method takes an object of type "customer", sets it's "add_date" property to the current date and adds it to the "customers" array.

- **outputCustomerById(customer_id)**

  This method takes a number representing a customer_id and outputs all of the customer data for the corresponding customer_id from the "customers" array (including their address from the "addresses" array (**HINT**: there is a function we will write called **getAddressAddressById(address_id)** that will be perfect for this), in the format:

  Customer ***customer_id***: ***first_name last_name*** (***email***)
  Home Address: ***address city***, ***province***. ***postal_code***
  Joined: ***add_date***

  For Example, **CustomerDB.outputCustomerById(26)**:

  Customer 26: Dave Bennett (dbennett@gmail.com)
  Home Address: 3945 John St. Ajax, ON. L7M4T9
  Joined: Wed Feb 01 2017 22:13:22 GMT-0500 (EST)

- **outputAllCustomers( )**

  This method takes no parameters and simply outputs all customers in the **same format as above**, including a header at the top of the output stating "All Customers", ie:

  All Customers

  Customer 26: Dave Bennett (dbennett@gmail.com)
  Home Address: 3945 John St. Ajax, ON. L7M4T9
  Joined: Wed Feb 01 2017 22:13:22 GMT-0500 (EST)

  Customer 59: John Stevens (jstevens22@hotmail.com)
  Home Address: 391 Baker St. Apt 231 Mississauga, ON. M4T8S3

Joined: Wed Feb 01 2017 22:13:22 GMT-0500 (EST)

…etc…

- **outputCustomersByStore (store_id)**

  This method takes a number representing a store_id and outputs all of the customer data for the corresponding store_id from the "customers" array in the **same format as above**, including a header at the top of the output in the format: Customers in Store ***name***

  For Example, **CustomerDB.outputCustomersByStore(297);**

  Customers in Store: Scotiabank - Main Branch

  Customer 26: Dave Bennett (dbennett@gmail.com)
  Home Address: 3945 John St. Ajax, ON. L7M4T9
  Joined: Wed Feb 01 2017 22:29:06 GMT-0500 (EST)

  Customer 63: Steven Edwards (steven2231@hotmail.com)
  Home Address: 67 Rhymer Ave. Stouffville, ON. L3C8H4
  Joined: Wed Feb 01 2017 22:29:06 GMT-0500 (EST)

  Customer 73: Melissa Bennett (mbennett@gmail.com)
  Home Address: 3945 John St. Ajax, ON. L7M4T9
  Joined: Wed Feb 01 2017 22:29:06 GMT-0500 (EST)

  HINT: there is a function we will write called **getStoreById(store_id)** that can help to get the name of the store with id 297 (for example).

- **removeCustomerById (customer_id)**

  This method takes a number representing a customer_id and searches through the customers array and removes the customer with the matching "customer_id" property from the array.

  This method must also ensure that **the corresponding address is removed** from the addresses array **only if** there are no other "customer" or "store" objects still using it. For example, if we remove Customer 26: Dave Bennett, we **cannot remove** his corresponding address (address_id: 4536) because his wife, Melissa is still registered and still uses that address (address_id: 4536).

  However, If we choose to remove Customer 59: John Stevens, we **can remove** his address (address_id: 2473), because nobody else is using it.

  HINT: there is a function we will write called **removeAddressAddressById(address_id)** that will be perfect for this.

  HINT: to remove elements from the middle of an array, you can either build a new array one element at a time, making sure to NOT include the element you don't want, or check out the Array.prototype.splice() method.

Methods to work with "address" data

The following are all methods that deal primarily with the "addresses" array. Any output for these functions is meant for the web console, so whenever the term "output" is used, you may assume that we're outputting to the console with console.log(). HINT: to refer to the "addresses" array from within these methods, use the "this" keyword, ie: "this.addresses".

- **addAddress (addressObj)**

  This method takes an object of type "address", and adds it to the "addresses" array.

- **getAddressById (address_id)**

  This method takes a number representing an address_id and searches through the "addresses" array looking for an address object that has a matching "address_id". This method will return the corresponding address object, for example: **CustomerDB.getAddressById(2727);** will return the object with address " 287 Brant St. Apt 4A" in the city "Waterdown".

- **outputAllAddresses( )**

  This method takes no parameters and simply outputs all addresses in the following format, including a header at the top of the output stating "All Addresses": ie:

  All Addresses

  Address ***address_id***: ***address city***, ***province***. ***postal_code***

  For Example, **outputAllAddresses();**

  All Addresses

  Address 1023: 2895 Yonge St. Toronto, ON. L4C02G

  Address 1984: 3611 Main St. West Hamilton, ON. R5O8H5

  Address 1757: 1177 Ontario St. Unit 8 Mississauga, ON. L9H6B3

  …etc…

- **removeAddressById (address_id)**

  This method takes a number representing an address_id and searches through the addresses array and removes the address with the matching "address_id" property **only if** the "address_id" is not referenced by any "customer" objects in the customer array, or "store" objects in the "store" array.

  For example, if we try to remove the address object with address_id 4536 after Customer 26: Dave Bennett, has been removed, we **cannot remove** it because his wife Melissa is still a registered "customer" and still uses that address.

  However, If we choose to remove the address object with address_id 2727 after Customer 71: Martin Scott, has been removed, we **can remove** it, because nobody else is using it.

  HINT: to remove elements from the middle of an array, you can either build a new array one element at a time, making sure to NOT include the element you don't want, or check out the Array.prototype.splice() method.

Methods to work with "store" data

The following are all methods that deal primarily with the "stores" array. Any output for these functions is meant for the web console, so whenever the term "output" is used, you may assume that we're outputting to the console with console.log(). HINT: to refer to the "stores" array from within these methods, use the "this" keyword, ie: "this.stores".

- **addStore (storeObj)**

  This method takes an object of type "store", and adds it to the "stores" array.

- **getStoreById (store_id)**

  This method takes a number representing a store_id and searches through the "stores" array looking for a store object that has a matching "store_id". This method will return the corresponding store object, for example: **CustomerDB.getStoreById(297);** will return the object with name " Scotiabank - Main Branch".

- **outputAllStores( )**

  This method takes no parameters and simply outputs all stores (including their address from the "addresses" array - **HINT**: the function **getAddressAddressById(address_id)** is perfect for this), in the following format, including a header at the top of the output stating "All Stores": ie:

  All Stores

  Store ***store_id***: ***name***
  Location: ***address_id***: ***address city***, ***province***. ***postal_code***

  For Example, **outputAllStores ();**

  All Stores

  Store 297: Scotiabank - Main Branch
  Location: 2895 Yonge St. Toronto, ON. L4C02G

  Store 614: Scotiabank - Hamilton
  Location: 3611 Main St. West Hamilton, ON. R5O8H5

  Store 193: Scotiabank - Mississauga
  Location: 1177 Ontario St. Unit 8 Mississauga, ON. L9H6B3

TEST DATA Output

Once you're ready to test your solution, uncomment the "TEST DATA" block of code and run your code. If your console output looks like this Sample, everything should be working properly.

Please Note, your "Joined:" dates will be different, as you will run your program at a different time.

Assignment Submission:

- Make sure that the "TEST DATA" block of code is uncommented and that your solution gives the correct output (see this sample) in the web console when run
- Add the following declaration at the top of your code:

```
/*********************************************************************************
* WEB222 – Assignment 02
* I declare that this assignment is my own work in accordance with Seneca  Academic Policy.
* No part of this assignment has been copied manually or electronically from any other source
* (including web sites) or distributed to other students.
*
* Name: _____ Student ID: _____ Date: _____
*
*********************************************************************************/
```

- Submit your **assignment2.js** file to My.Seneca under **Assignments** -> **Assignment** 2

Important Note:

- **NO LATE SUBMISSIONS** for assignments. Late assignment submissions will not be accepted and will receive a **grade of zero (0)**.
- After the end (11:59PM) of the due date, the assignment submission link on My.Seneca will no longer be available.