# Assignment 4 – OOP: Encapsulation

This is an exercise in the OOP concept of ***Encapsulation***, whereby a class contains both the properties of an object and the methods to manage those properties. It is also an exercise in ***abstraction***, which permits us to upgrade the database from text files to an RDBMS without modifying the user interface.

Where you see ***HK***, insert your initials instead.

1. Either create a new Windows project or continue with the current one, if you have an MDI main page and menu.
2. Add a folder to your project called ***HKClasses***.

### HKProvince Class

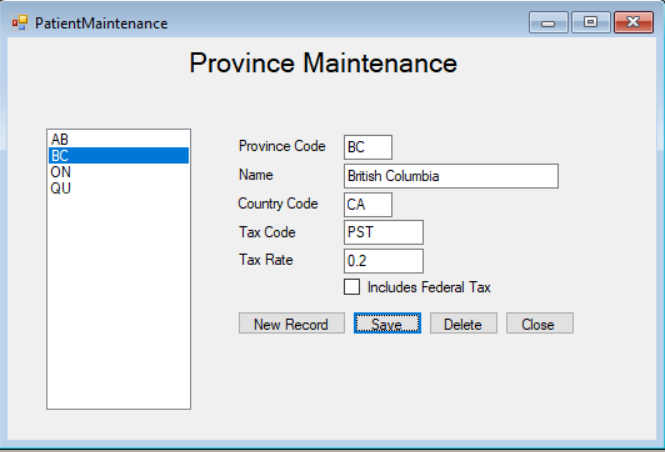
Create a class called ***HKProvince*** in the HKClasses folder. The following are *instance* fields and methods unless stated otherwise.

1. Add global private class-level StreamReader, StreamWriter variables and a string for the path to your output file … give this the value **“C:\PROG1815data\Province.txt”**. This is to stop you retyping it throughout the code
2. Add the following properties to the class (either as public properties or as private attributes & public properties):
   1. A string: *ProvinceCode*
   2. A string: *Name*
   3. A string: *CountryCode*
   4. A string: *TaxCode*
   5. A Double: *TaxRate*
   6. A Boolean: *IncludesFederalTax*
3. Override the ***ToString***() method inherited by your ***HKProvince*** class. This will create the output record required by every method that writes to the output file:
   1. Concatenate the properties of the current HKProvince object together into a single string. Use either a delimiter or fixed-length fields to differentiate them, so your parse routine can reconstruct the object again later.
   2. Return the string.
4. Add a class method called ***HKParseProvince***. This creates a Province object from a record read in from the Province file by other methods.
   1. Accept a string parameter and return an object of the ***HKProvince*** class.
   2. Parse the input string into individual fields
   3. Instantiate an object of the class, populate its properties from the fields in the string, and return it.
   4. Throw a “not a province record” exception (showing the incoming record) if the parameter does not contain the correct number of or type of fields.
5. Add a public class method called ***HKGetByProvinceCode*** which accepts a stringparameter called *provinceCode*, retrieves the province record with the given province code and returns it as an HKProvince object … or null if not found.
6. Add a public class method called ***HKGetByProvinceName*** which accepts a string parameter called *provinceName*, retrieves the province record with the given name and returns it as an HKProvince object … or null if not found.
7. Add a public class method called ***HKGetProvinces*** which has no parameters and returns a List<HKProvince> containing all provinces on file.
8. Add a private void method called ***HKEdit*** that takes the current HKProvince object and performs the following validations and reformats:
   1. Initialize an empty string to collect all error messages. Each error message should:
      1. Be delimited with a newline character.
      2. Include the field-name in the message.
   2. Trim all strings, converting nulls into an empty string.
   3. *Name* is required.
   4. *ProvinceCode* and *CountryCode* are required and must be exactly 2 **letters**. Accept either case, but shift them to upper case.
   5. Set a global private variable ***IsEdit*** to true if the object’s province code is already on file, false otherwise (a safeguard against duplicate keys).
   6. Reject the current record if it will create a duplicate *Name* on file. Keep in mind that this validation should not inhibit editing an existing record that is retaining its name.
   7. If the *TaxCode* is provided, shift it to upper case and ensure it is only letters. It can be empty/null. No needs to check TaxRate is zero or not?????, TaxRate can be zero?
   8. If the *TaxCode* is not provided, the *TaxRate* must be zero. Otherwise, it can be zero to one, inclusive.
   9. The Boolean *IncludesFederalTax* can be true or false, but must be false if *TaxRate* is zero (yes, we’re looking at you, Alberta).
   10. If no errors were found, allow the method to exit cleanly. Otherwise, throw an exception with the accumulated error messages.
9. Add a method called ***HKAdd*** that sends the current HKProvince object to its edit method. If it returns from the edit and ***IsEdit*** is false, convert the object to a string and add it to the file. If ***IsEdit*** is true, throw a “province code already on file” exception.
10. Add a method called ***HKDelete*** that removes the record on file with the same province code. If there is no record on file with the same province code, throw an exception with a pertinent message.
11. Add a method called ***HKUpdate*** that sends the current HKProvince object to its the edit method. If it returns from the edit, but the edit noticed there was no record with the same province code, throw an exception with a pertinent message. Otherwise, convert the current object to a string and replace the record on file with the current one.
    1. Hint: “update” is a “delete” followed by an “add”.

### HKProvinceMaintenance

You must use your ***HKProvince*** class to retrieve and maintain records in the Province file. Failure to do so will result in zero in the assignment.

1. Create a Windows form to maintain records on the Province file, similar to the example below.



1. The ListBox on the left lists all province codes currently on file … selecting one fires its *SelectedIndexChanged* event, with the selected province code as its *SelectedItem*.
   1. Fetch the selected province object using your HKProvince Class’ GetByProvinceCode method, load the input areas, and remember that you’re in Update.
2. When the Add button is clicked, clear the input areas and remember that you’re in Add.
3. The Save button saves the current record to file. If successful, update the ListBox’s contents and change its selected entry to the one just saved (which should fire SelectedIndexChanged).
4. When a province record is deleted, update ListBox’s contents, set its selected index to -1 and clear the input areas.
5. For all file I/O, use a try-catch to intercept and display exceptions thrown by your HKProvince class.

### Hand In

* + ou