Project 2

T-213-VEFF, Vefforritun

School/Student Phonebook

Your task is to create the classes School, Student and Phonebook, such that the following Main( ) function should compile and execute correctly:

```
namespace PhonebookApp
{
        class Program
        {
        static void Main(string[] args)
        {
                PhoneBook phonebook = PopulatePhoneBook();
                string command = string.Empty;

                Console.WriteLine("Welcome to phonebook 1.0");
                Console.WriteLine(Environment.NewLine);
                Console.WriteLine("Commands available are:");
                Console.WriteLine("addschool: Add new school");
                Console.WriteLine("addstudent: Add student to school");
                Console.WriteLine("editstudent: Edit student in school");
                Console.WriteLine("print: Print phonebook");
                Console.WriteLine("printhtml: Print phonebook as HTML");
                Console.WriteLine("quit: Quit program");
                Console.WriteLine(Environment.NewLine);

                while (true)
                {
                        Console.WriteLine("Please enter command: ");
                        command = Console.ReadLine();
                        Console.WriteLine(Environment.NewLine);
                        switch (command)
                        {
                                case "addschool":
                                        phonebook.AddNewSchool();
                                        break;
                                case "addstudent":
                                        phonebook.AddNewStudentToSchool();
                                        break;
                                case "editstudent":
                                        phonebook.EditStudentInSchool();
                                        break;
                                case "print":
                                        Console.WriteLine(phonebook);
                                        break;
                                case "printhtml":
                                        Console.WriteLine(phonebook.Html());
                                        break;
                                case "quit":
                                        return;
                                default:
                                {
                                        Console.WriteLine("Command not recognized");
                                        break;
                                }

                        }

                }
        }
        }
}
```

For example, when the program is running, after entering command „print“, the output should look like this:

```
School: HR
     0101054449 - Helena - 22
     0102054449 - Katrin - 21
     0103054449 - Jon - 23
Total number of students: 3

School: HÍ
     0104054449 - Sigurdur - 22
     0105054449 - Inga - 21
Total number of students: 2

School: LÍ
     0106054449 - Haukur - 20
     0107054449 - Baldur - 24
Total number of students: 2
```

The PopulatePhoneBook function looks as follows and can be added besides the Main function the program class:

```
static PhoneBook PopulatePhoneBook()
{
        PhoneBook phonebook = new PhoneBook();

        School school1 = new School("HR");
        school1.AddStudent(new Student { Ssn = "0101054449", Name = "Helena", Age = 22 });
        school1.AddStudent(new Student { Ssn = "0102054449", Name = "Katrin", Age = 21 });
        school1.AddStudent(new Student { Ssn = "0103054449", Name = "Jon", Age = 23 });
        phonebook.Schools.Add(school1);

        School school2 = new School("HÍ");
        school2.AddStudent(new Student { Ssn = "0104054449", Name = "Sigurdur", Age = 22 });
        school2.AddStudent(new Student { Ssn = "0105054449", Name = "Inga", Age = 21 });
        phonebook.Schools.Add(school2);

        School school3 = new School("LÍ");
        school3.AddStudent(new Student { Ssn = "0106054449", Name = "Haukur", Age = 20 });
        school3.AddStudent(new Student { Ssn = "0107054449", Name = "Baldur", Age = 24 });
        phonebook.Schools.Add(school3);
}
```

Implementation hints:

- Create the classes Phonebook, School and Student. You will need to write a constructor for School that accepts a single string which is the name of the school.

- Create properties for the class School. The class School needs only one property which stores the name of the school (of type string). It also needs a class member variable which stores a collection of students. The best way to do that is to use a class similar to C++'s std::vector, called List<>. It can be found in the namespace System.Collections.Generic, so you will need to add a using statement at the top of the School.cs file. Then, you should initialize this variable in the class constructor:

```
Students = new List<Student>( );
```

- Similarly you have to add one class member variable (property of the `List<School>`)to the class PhoneBook which stores a collection of schools. You could use the same method as for studens for this property.

- Add a function to the class School called AddStudent. It should accept a single parameter of type Student. It should simply store the student instance in the List<Student> member variable (use the Add( ) function which works similarly to the push_back( ) function in std::vector).

- Add properties to the class Student. You will need two properties: SSN, Name and Age. I'll leave it to you to figure out the type of these properties.

- Override the ToString functions in both classes (see the book for details). The Student version of ToString is quite simple, while the School implementation will need the following:

    - A local string variable that will contain the end result. You can then use the += operator to concatenate to this variable, and use System.Environment.NewLine; to add a newline to the string. Example:

```
string result = "Some string" + System.Environment.NewLine;
result += "some other string";
```

    - A loop that walks through all students in the collection (also, see the book about foreach statements), and adds each student to the string. Note: you can either call the ToString method for each student explicitly, or use a syntax similar to the one seen inside Main above, where the School instance is printed out explicitly.

- You will new to added the following functions to the PhoneBook class in order to comply with the main program given:
    - AddNewSchool:
        - Ask user for name. Create new school and add to PhoneBook schools.
    - AddNewStudentToSchool:
        - For this part you will have to figure out how to ask the user for which school to add the student to. One solution could be to ask the user for the index of the school and then ask user for ssn, name and age, and add the student to chosen school.
    - EditStudentInSchool:
        - Very similar for when adding new students to schools, you will have to figure out how to ask the user for which school to look for the student in as well as how to ask the user for which student to edit. Both could be solved by asking the user for index of school and then index of student in the corresponding list. Then just ask user for update ssn, name and age, and update the chosen student.
    - Override the ToString:
        - Simililar to the School class, you could have a loop that walks through all school in the phonebook and call the ToString method for each school explicitly. The printout example above should give you a good idea this should look like
    - Html:
        - This function should be very similar to sample printout, e.i. it should return a string containing the printout for the entire phonebook. The only difference being that output should be valid HTML.
        - This could be an example for the HTML function:

```
public string Html()
{
        StringBuilder result = new StringBuilder();
        //...
        result.Append(" append html some content ");
        //...
        //Create and save file a project root
        using (StreamWriter writer = new StreamWriter("result.html"))
        {
                writer.WriteLine(result);
        }
        //return HTML
        return result.ToString();
}
```

        - You will also need to implement the following requirements:

            - Make sure to have a single table in the body of the document
            - <html>, <head>, <title>, and <body> are mandatory
            - all tags must be properly nested
            - all tags are in lowercase (<h1> instead of <H1>)
            - all tags are properly closed, and empty elements have a trailing slash
            - (, and)
            - all attribute values are properly quoted (instead of)
            - all attribute names must be in lower case

Hand in

The deadline for this assignment can be seen on myschool. Hand in a single .zip/.rar/.7z file containing the .cs files plus the .exe file:

- School.cs
- Student.cs
- PhoneBook.cs
- Program.cs
- PhoneBookApp.exe

Please note that this is an individual assignment.