ASSIGNMENT 7

INTRODUCTION

Before you begin this assignment, make sure your **BowersHeading** file has been renamed to **LastNameHeading** (where *LastName* is your own last name).

ASSIGNMENT

PART 1

Create a *static* class named *MedicalHelper*. You will have to conduct some research to determine how to create a *static* class. I will not tell you. Once you have created your static class, create the methods shown ***and*** also write the code for each method. Assignment 4.15 on page 141 of your text discusses how to calculate heart rates; assignment 3.31 on pages 104 and 105 of your text discusses how to calculate body mass index (BMI).

```
public static double GetBMIValue (double weightPounds, double heightInches)
{
      //TODO: implementation
}
public static string GetBMIName(double weightPounds, double heightInches)
{
      //TODO: implementation
}
public static int GetMaximumHeartRate(int age)
{
    //TODO: implementation
}
public static int GetMaxTargetHeartRate(int age)
{
    //TODO: implementation
}
public static int GetMinTargetHeartRate(int age)
{
    //TODO: implementation
}
```

PART 2

Create a *non-static* class (e.g., a regular class) named *Patient*. Patient must implement the following properties:

• FirstName	(string; get/set)
• LastName	(string; get/set)
• Age	(int; get/set)
• WeightInPounds	(double; get/set)
• HeightInInches	(double; get/set)

PART 3

In your main application

1. Create a constant string named *APP_NAME*, which holds the name of your application, which will be *My Medical Helper*.

2. In your main application, make sure to maintain the existing *Main* method. Add the following methods, making sure to copy the implementation of the *displayPatient* method exactly as shown (do not copy and paste, otherwise you may end up with MS Word's *magic quotes* causing you problems).

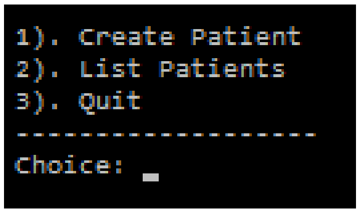
```
static int displayMainMenu()
{
    //TODO: Implementation
}

static Patient createPatient()
{
    //TODO: Implementation
}

static void listPatients(Patient[] patients)
{
    //TODO: Implementation
}

static void displayPatient(Patient p)
{
    Console.Clear();
    LastNameHeading.getHeading(APP_NAME); //Change this line to reflect your own heading class
    Console.WriteLine("  -------------------------");
    Console.WriteLine("  Patient Information");
    Console.WriteLine("  -------------------------");
    Console.WriteLine("  {0, -25}: {1}", "First Name", p.FirstName);
    Console.WriteLine("  {0, -25}: {1}", "Last Name", p.LastName);
    Console.WriteLine("  {0, -25}: {1}", "Age", p.Age);
    Console.WriteLine("  {0, -25}: {1}", "Height (inches)", p.HeightInInches);
    Console.WriteLine("  {0, -25}: {1}", "Weight (pounds)", p.WeightInPounds);
    Console.WriteLine("  -------------------------");
    Console.WriteLine("  Body Mass Index");
    Console.WriteLine("  -------------------------");
    Console.WriteLine("  {0, -25}: {1:0.00}", "BMI", MedicalHelper.GetBMIValue(p.WeightInPounds, p.HeightInInches));
    Console.WriteLine("  {0, -25}: {1}", "BMI is considered:", MedicalHelper.GetBMIName(p.WeightInPounds, p.HeightInInches));
    Console.WriteLine("  -------------------------");
    Console.WriteLine("  Heart Rate Info");
    Console.WriteLine("  -------------------------");
    Console.WriteLine("  {0, -25}: {1}", "Max Heart Rate", MedicalHelper.GetMaximumHeartRate(p.Age));
    Console.WriteLine("  {0, -25}: {1}", "Max Target Heart Rate", MedicalHelper.GetMaxTargetHeartRate(p.Age));
    Console.WriteLine("  {0, -25}: {1}", "Min Target Heart Rate" , MedicalHelper.GetMinTargetHeartRate(p.Age));
    Console.WriteLine();
    Console.WriteLine("  Press enter to return to main screen.");
    Console.ReadLine();
}
```

Continued on the next page.

3. In your main method:

   a. Create an array of *Patient* objects.

   b. Clear the screen and prompt the user as shown in Figure 1. Make sure to display your personal heading, passing the APP_NAME constant as the argument.



```
1). Create Patient
2). List Patients
3). Quit
-------------------
Choice: _
```

Figure 1

   c. When the user enters a value:

      i. Make sure the value is numeric. If the value is not numeric, display an error and prompt the user for a choice.

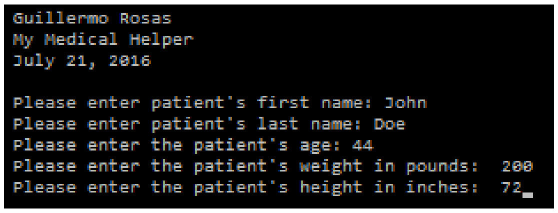
      ii. Make sure the value is valid. For example, Figure 1 displays 3 options: 1, 2, and 3. If the user enters a value other than the valid options, display an error and prompt the user for a choice.

      iii. If the user enters 1:

         1. Call the *createPatient()* method. The *createPatient()* method will return a Patient object. Store the patient object in the *Patients* array, resizing the array as needed. In the body of *createPatient()*:

            a. Clear the screen and display your personal heading, passing the APP_NAME constant as an argument.

            b. Prompt the user to enter the patient's first name, last name, age, weight in pounds, and height in inches as shown in Figure 2. For numeric values (age, weight, and pounds) – make sure the value is a number that is greater than zero. If the input value is not a number that is greater than zero, display an error and prompt for the information again.



```
Guillermo Rosas
My Medical Helper
July 21, 2016

Please enter patient's first name: John
Please enter patient's last name: Doe
Please enter the patient's age: 44
Please enter the patient's weight in pounds:  200
Please enter the patient's height in inches:  72_
```

Figure 2

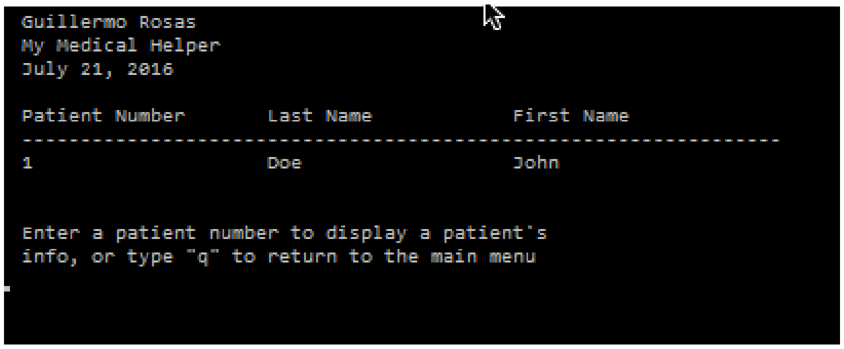
Continued on the next page.

c. Once all user information is input, return to the main menu.

iv. At the main menu, if the user enters 2:

1. Call the *listPatients* method. In the body of *listPatients*, display a listing of all the patients stored in the *patients* array.

2. Prompt the user to enter a patient number to view the individual patient's information, or enter "q" to quit – an example of this is shown in Figure 3.



```
Guillermo Rosas
My Medical Helper
July 21, 2016

Patient Number      Last Name           First Name
-----------------------------------------------------------------
1                   Doe                 John


Enter a patient number to display a patient's
info, or type "q" to return to the main menu
```

Figure 3

3. If the user enters "q", return to the main menu. If the user enters a value other than "q":

   a. Check that the value is a number. If it isn't, display an error and prompt the user to input a valid value.

   b. If the value is a number, check to make sure it's greater than zero. If it is zero or less, display an error and prompt the user to input a valid.

   c. If the user enters a value that is greater than zero, check to make sure the value is within the bounds of the array. If the value exceeds the bounds of the array, display an error and prompt the user to input a valid that cannot exceed #, where # is the upper bound of the array.

   d. If the user enters a valid number within the bounds of the array, call the *displayPatient* method, passing the individual *patient* object to the method. Do not change the implementation of the *displayPatient* method – this method is a test method.

Continued on the next page.

NOTES

- ***FOR EACH OF THE METHODS PROVIDED,*** *do not*:
    - Change the method name, parameter names or types, or return types.
    - Do not modify the implementation of *displayPatient* – your code will work with *displayPatient* and if it doesn't, modify your code.
- You have been provided a sample built application in Blackboard.

COMPLETION

- When complete, please ZIP the entire solution into a single file named **LastName_7.zip** (where *LastName* is your own last name).
- Upload and submit the ZIP file to the Assignment 3 drop box in Blackboard.