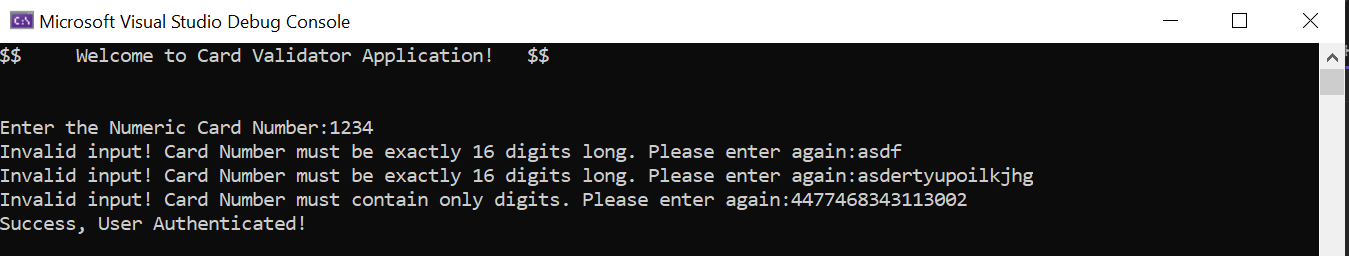
**Assignment 1: Access Card Validator Application**

using System;

using System.Linq;

namespace AccessCardValidator

{

internal class Program

{

/// <summary>

/// Takes a string input from the user and validates it.

/// </summary>

/// <param name="PrintValue">The prompt message to be displayed to the user.</param>

/// <returns>The valid 16-digit numeric string entered by the user.</returns>

static string TakingStringInput(string PrintValue)

{

string? str;

do

{

str = Console.ReadLine();

if (string.IsNullOrWhiteSpace(str) || str.Length != 16)

{

Console.Write($"Invalid input! {PrintValue} must be exactly 16 digits long. Please enter again:");

continue;

}

if (!str.All(char.IsDigit))

{

Console.Write($"Invalid input! {PrintValue} must contain only digits. Please enter again:");

continue;

}

} while (string.IsNullOrWhiteSpace(str) || str.Length != 16 || !str.All(char.IsDigit));

return str;

}

/// <summary>

/// Reverses the order of the digits in the card number.

/// </summary>

/// <param name="CardNumber">The 16-digit numeric card number.</param>

/// <returns>The reversed card number.</returns>

static string ReverseCardNumber(string CardNumber)

{

return new string(CardNumber.Reverse().ToArray());

}

/// <summary>

/// Checks the validity of the card number using the Luhn algorithm.

/// </summary>

/// <param name="CardNumber">The 16-digit numeric card number.</param>

/// <returns>True if the card number is valid, false otherwise.</returns>

static bool CheckValidation(string CardNumber)

{

// Ensuring the card number is not all zeros, all zeros will fail this algorithm

if (CardNumber.All(c => c == '0'))

{

return false;

}

string NewCardNumber = ReverseCardNumber(CardNumber);

int result = CalculateChecksum(NewCardNumber);

return result % 10 == 0;

}

/// <summary>

/// Calculates the checksum of the reversed card number using the Luhn algorithm.

/// </summary>

/// <param name="input">The reversed card number.</param>

/// <returns>The checksum calculated.</returns>

static int CalculateChecksum(string input)

{

int sum = 0;

for (int i = 0; i < input.Length; i++)

{

int digit = (int)Char.GetNumericValue(input[i]);

if (i % 2 != 0) // Even position

{

digit \*= 2;

if (digit > 9) // Check if the result is two digits

{

sum += SumOfDigits(digit.ToString());

}

else

{

sum += digit;

}

}

else // Odd position

{

sum += digit;

}

}

return sum;

}

/// <summary>

/// To find Sum of digits of given string input

/// </summary>

/// <param name="input"> CardNumber in string</param>

/// <returns></returns>

static int SumOfDigits(string input)

{

int sum = 0;

foreach (char c in input)

{

if (char.IsDigit(c))

{

sum += (int)Char.GetNumericValue(c);

}

}

return sum;

}

static void Main(string[] args)

{

Console.WriteLine("$$ Welcome to Card Validator Application! $$\n\n");

Console.Write("Enter the Numeric Card Number:");

string CardNumber = TakingStringInput("Card Number");

bool IsValid = CheckValidation(CardNumber);

if (IsValid)

{

Console.WriteLine("Success, User Authenticated!");

}

else

{

Console.WriteLine("User Authentication Failed, Invalid User!");

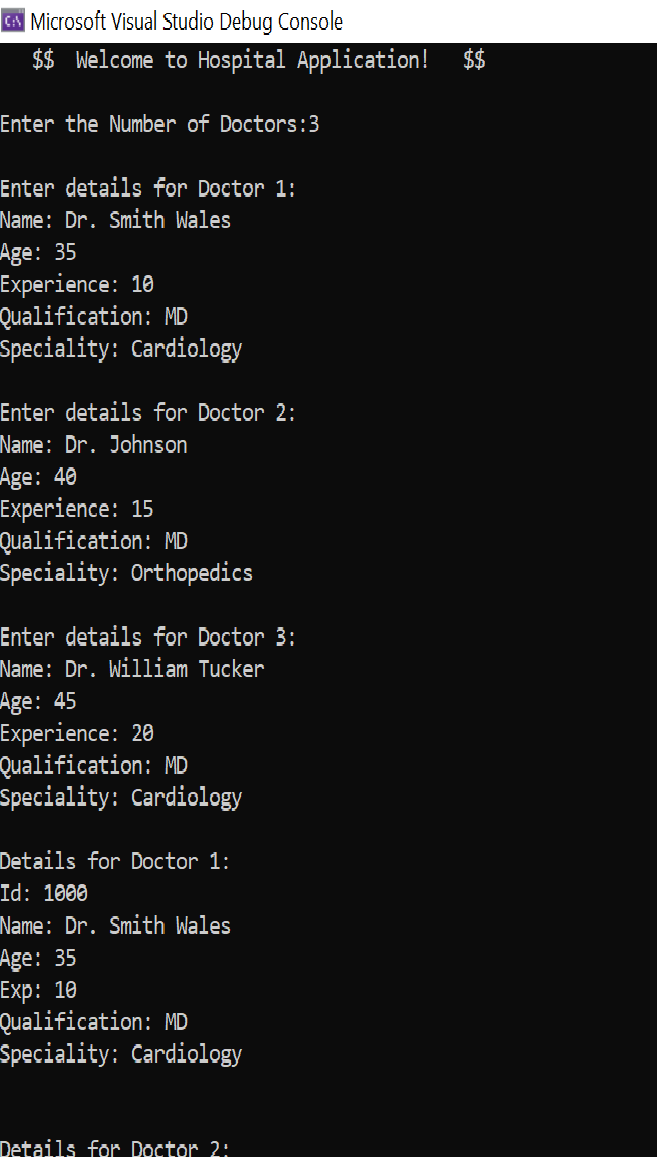
}

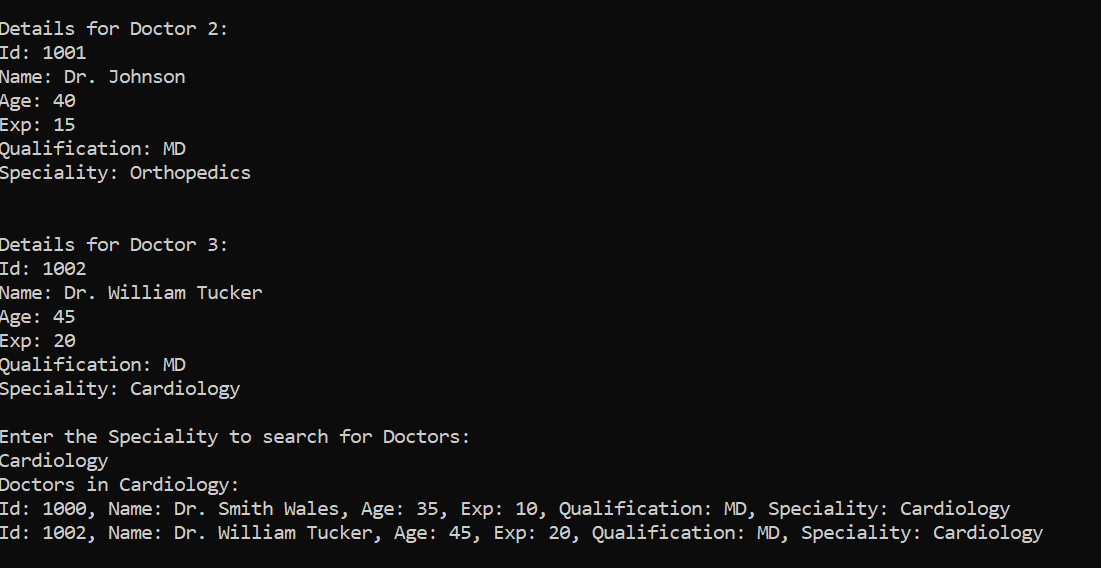
}

}

}

**Assignment 2: Create Doctors, Print them, and find by speciality Application.**

****

****

**Program.cs**

using System;

using System.Linq;

using System.Xml.Linq;

namespace Assignment

{

/// <summary>

/// A simple program to create and manage doctor records.

/// </summary>

internal class Program

{

/// <summary>

/// Takes a string input from the user.

/// </summary>

/// <param name="PrintValue">The prompt message to be displayed to the user.</param>

/// <returns>The string input provided by the user.</returns>

string TakingStringInput(string PrintValue)

{

string? str;

do

{

str = Console.ReadLine();

if (string.IsNullOrWhiteSpace(str))

Console.Write($"Invalid input, Please Enter {PrintValue} again:\n");

} while (string.IsNullOrWhiteSpace(str));

return str;

}

/// <summary>

/// Takes an integer input from the user.

/// </summary>

/// <param name="PrintValue">The prompt message to be displayed to the user.</param>

/// <returns>The integer input provided by the user.</returns>

int TakingIntInput(string PrintValue)

{

int value;

while (!int.TryParse(Console.ReadLine(), out value) || value <= 0)

{

Console.Write($"Invalid input! Please Enter {PrintValue} again:");

}

return value;

}

/// <summary>

/// Creates a new Doctor instance by taking input from the console.

/// </summary>

/// <returns>A new Doctor instance created with user input.</returns>

Doctor CreateDoctorThroughConsole(int id)

{

string name;

int age;

int exp;

string qualification;

string speciality;

Console.Write("Name: ");

name = TakingStringInput("Name");

Console.Write("Age: ");

age = TakingIntInput("Age");

Console.Write("Experience: ");

exp = TakingIntInput("Experience");

Console.Write("Qualification: ");

qualification = TakingStringInput("Qualification");

Console.Write("Speciality: ");

speciality = TakingStringInput("Speciality");

Doctor TempDoctor = new Doctor(id, name, age, exp, qualification, speciality);

return TempDoctor;

}

/// <summary>

/// Prints details of doctors by speciality.

/// </summary>

/// <param name="doctors">Array of Doctor objects.</param>

/// <param name="speciality">The speciality to search for.</param>

public void PrintDoctorsBySpeciality(Doctor[] doctors, string speciality)

{

Console.WriteLine($"Doctors in {speciality}:");

bool found = false;

foreach (var doctor in doctors)

{

if (doctor.Speciality == speciality)

{

Console.WriteLine($"Id: {doctor.Id}, Name: {doctor.Name}, Age: {doctor.Age}, Exp: {doctor.Exp}, Qualification: {doctor.Qualification}, Speciality: {doctor.Speciality}");

found = true;

}

}

if (!found)

{

Console.WriteLine("No doctors found with the given speciality.");

}

}

/// <summary>

/// The entry point of the program.

/// </summary>

/// <param name="args">Command-line arguments.</param>

static void Main(string[] args)

{

Console.WriteLine(" $$ Welcome to Hospital Application! $$ \n");

Program p = new Program();

Console.Write("Enter the Number of Doctors:");

int NumberOfDoctors = p.TakingIntInput("Number of Doctors");

Doctor[] doctors = new Doctor[NumberOfDoctors];

for (int i = 0; i < doctors.Length; i++)

{

Console.WriteLine($"\nEnter details for Doctor {i + 1}:");

doctors[i] = p.CreateDoctorThroughConsole(1000 + i);

}

for (int i = 0; i < doctors.Length; i++)

{

Console.WriteLine($"\nDetails for Doctor {i + 1}:");

doctors[i].PrintDoctorDetails();

}

Console.WriteLine("Enter the Speciality to search for Doctors: ");

string speciality = p.TakingStringInput("Speciality");

p.PrintDoctorsBySpeciality(doctors, speciality);

}

}

}

**Doctor.cs**

/// <summary>

/// Represents a Doctor entity with basic information such as ID, name, age, experience, qualification, and speciality.

/// </summary>

internal class Doctor

{

/// <summary>

/// Gets the ID of the doctor.

/// </summary>

public int Id { get; private set; }

/// <summary>

/// Gets the name of the doctor.

/// </summary>

public string Name { get; private set; }

/// <summary>

/// Gets the age of the doctor.

/// </summary>

public int Age { get; private set; }

/// <summary>

/// Gets the years of experience of the doctor.

/// </summary>

public int Exp { get; private set; }

/// <summary>

/// Gets the qualification of the doctor.

/// </summary>

public string Qualification { get; private set; }

/// <summary>

/// Gets the speciality of the doctor.

/// </summary>

public string Speciality { get; private set; }

/// <summary>

/// Default constructor for Doctor class.

/// </summary>

public Doctor()

{

Id = 0;

Name = string.Empty;

Age = 0;

Exp = 0;

Qualification = string.Empty;

Speciality = string.Empty;

}

/// <summary>

/// Parameterized constructor for Doctor class.

/// </summary>

/// <param name="id">ID of the doctor.</param>

/// <param name="name">Name of the doctor.</param>

/// <param name="age">Age of the doctor.</param>

/// <param name="exp">Years of experience of the doctor.</param>

/// <param name="qualification">Qualification of the doctor.</param>

/// <param name="speciality">Speciality of the doctor.</param>

public Doctor(int id, string name, int age, int exp, string qualification, string speciality)

{

Id = id;

Name = name;

Age = age;

Exp = exp;

Qualification = qualification;

Speciality = speciality;

}

/// <summary>

/// Prints the details of the doctor to the console.

/// </summary>

public void PrintDoctorDetails()

{

Console.WriteLine($"Id: {Id}\nName: {Name}\nAge: {Age}\n" +

$"Exp: {Exp}\nQualification: {Qualification}\n" +

$"Speciality: {Speciality}\n");

}

}