Question 1 to 4

CLASS DOCTOR :

using System;

using System.Collections.Generic;

using System.Dynamic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Task1Doctor

{

internal class Doctor

{

/// <summary>

/// Constuctor for initial value of Doctor attributes

/// </summary>

public Doctor()

{

int Id = 0;

string Name=string.Empty;

int Age = 0;

string Qualification = string.Empty;

int Experience = 0;

string Speciality= string.Empty;

}

/// <summary>

/// constructor to set doctors id

/// </summary>

/// <param name="id"></param>

public Doctor(int id)

{

Id = id;

}

public int Id { get; set; }

public string Name { get; set; }

public int Age { get; set; }

public int Experience { get; set; }

public string Qualification { get; set; }

public string Speciality { get; set; }

/// <summary>

/// Method to Print details of the doctor

/// </summary>

public void PrintDoctorsDetail()

{

Console.WriteLine(" --------------------------------------------");

Console.WriteLine($"Doctors Id \t:\t {Id}");

Console.WriteLine($"Doctors Name \t:\t {Name}");

Console.WriteLine($"Doctors Age \t:\t {Age}");

Console.WriteLine($"Doctors Exp \t:\t {Experience}");

Console.WriteLine($"Doctots speciality \t:\t {Speciality}");

Console.WriteLine($"Doctots Qualification \t:\t {Qualification}");

}

}

}

CLASS PROGRAM

using System.Dynamic;

using System.Numerics;

namespace Task1Doctor

{

internal class Program

{

/// <summary>

/// Create a doctor and return a doctor object

/// </summary>

/// <param name="id"> take a id of doctor</param>

/// <returns></returns>

Doctor CreateDoctorViaConsole(int id)

{

Doctor doctor = new Doctor(id);

Console.WriteLine(" --------------------------------------------");

// get name

Console.WriteLine("Please enter Doctor's name ");

doctor.Name = Console.ReadLine();

// get age

int age;

Console.WriteLine("Enter Doctor's age");

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

{

Console.WriteLine("invalid entry , try again");

}

doctor.Age = age;

//get experience

int experience;

Console.WriteLine("Enter Doctor's experience");

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

{

Console.WriteLine("invalid entry , try again");

}

doctor.Experience = experience;

// get qualification

Console.WriteLine("Please enter doctor's Qualification");

doctor.Qualification = Console.ReadLine();

// get Speciality

Console.WriteLine("Please enter doctor's Speciality");

doctor.Speciality = Console.ReadLine();

return doctor;

}

/// <summary>

/// Get total number of Doctor Initial

/// </summary>

/// <returns> return the count for doctors array</returns>

int GetDoctorsCount()

{

//current using it because I dont know replacment of VECTOR , that can grow and shrink

Console.WriteLine("Please Enter total number of doctors ");

int count;

while (!int.TryParse(Console.ReadLine(), out count))

{

Console.WriteLine("invalid entry , try again");

}

return count;

}

/// <summary>

/// function to get the doctor by the user entered specialization

/// </summary>

/// <param name="doctors">complete array of doctors </param>

void GetDoctorBySpecialization(Doctor[] doctors)

{

Console.WriteLine(" --------------------------------------------");

Console.WriteLine("Please enter doctor's Speciality for searching");

string speciality = Console.ReadLine();

bool flag = false;

Console.WriteLine("All doctors detail with given Specialization are : ");

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

{

if (doctors[i].Speciality == speciality)

{

doctors[i].PrintDoctorsDetail();

flag = true;

}

}

if (!flag)

{

Console.WriteLine("Sorry!! There is no such doctor with given Specialization Try Again");

while (true)

{

Console.WriteLine(" --------------------------------------------");

Console.WriteLine("Do you want to try again YES or NO");

string inp = Console.ReadLine();

if (inp == "YES")

{

GetDoctorBySpecialization(doctors);

break;

}

else if (inp == "NO")

{

return;

}

else

{

Console.WriteLine("Wrong choice");

}

}

}

}

static void Main(string[] args)

{

Program program = new Program();

int count = program.GetDoctorsCount();

Doctor[] doctors = new Doctor[count];

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

{

doctors[i] = program.CreateDoctorViaConsole(100+i);

}

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

{

doctors[i].PrintDoctorsDetail();

}

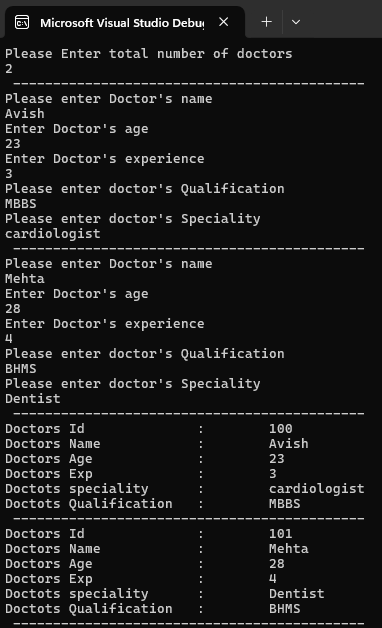
program.GetDoctorBySpecialization(doctors);

}

}

}

Output :



A screenshot of a computer

Description automatically generated

Task 5

using System.Net.Http.Headers;

namespace CardValidator

{

internal class Program

{

/// <summary>

/// function to take Card number from user

/// </summary>

/// <returns> the card number </returns>

string TakeUserInput()

{

Console.WriteLine("Enter a visa card number ");

string CardNumber=Console.ReadLine();

while (!ValidateInput(CardNumber))

{

Console.WriteLine("Not a correct input !!!! Try again");

CardNumber = Console.ReadLine();

}

return CardNumber;

}

/// <summary>

/// To check function to validate Cardnumber contains digits only

/// </summary>

/// <param name="input">Card number</param>

/// <returns> correct card number or not (true/false)</returns>

bool IsStringInteger(string input)

{

foreach (char c in input)

{

if (!char.IsDigit(c))

{

return false;

}

}

return true;

}

/// <summary>

/// check user input valid or not

/// </summary>

/// <param name="cardNumber"> Card Number </param>

/// <returns>true/false</returns>

bool ValidateInput(string cardNumber)

{

if(cardNumber.Length!=16)

return false;

if(string.IsNullOrEmpty(cardNumber))

return false;

if(!IsStringInteger(cardNumber))

return false ;

return true;

}

/// <summary>

/// function to reverse the card number

/// </summary>

/// <param name="input">Card number</param>

/// <returns> reverse of that card number</returns>

string ReverseCardNumber(string input)

{

string ReversedCardNumber = "";

for (int i = input.Length - 1; i >= 0; i--)

{

ReversedCardNumber += input[i];

}

return ReversedCardNumber;

}

/// <summary>

/// sum of the digits if greater than 9

/// </summary>

/// <param name="digit">sum when multiply by 2 </param>

/// <returns>return single digit number</returns>

int SumUntilSingleDigit(int digit)

{

int sum = 0;

while (digit != 0)

{

sum += digit % 10;

digit /= 10;

}

return sum;

}

/// <summary>

/// to validate the card number

/// </summary>

/// <param name="cardNumber"></param>

void ValidateCardNumber(string cardNumber)

{

string ReversedCardNumber=ReverseCardNumber(cardNumber);

int sum = 0;

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

{

int digit = int.Parse(ReversedCardNumber[i].ToString());

if (i%2!=0)

{

digit \*= 2;

while (digit > 9)

{

digit = SumUntilSingleDigit(digit);

}

}

sum += digit;

}

Console.WriteLine(sum);

bool result =sum % 10 == 0;

PrintResult(result);

}

/// <summary>

/// print the result

/// </summary>

/// <param name="result"> result i.e valid or not </param>

void PrintResult(bool result)

{

if (result)

{

Console.WriteLine("Yes it is a Valid Card Number");

return;

}

Console.WriteLine("Not a valid card number ");

}

static void Main(string[] args)

{

Program program = new Program();

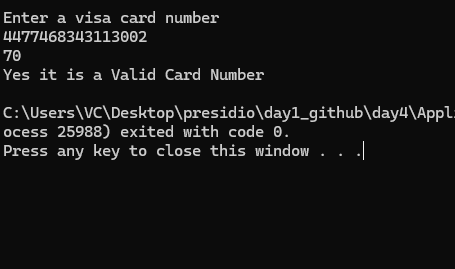
string CardNumber = program.TakeUserInput();

program.ValidateCardNumber(CardNumber);

}

}

}



A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated