Program.cs

namespace CardVerifier

{

internal class Program

{

static void Main(string[] args)

{

CardVerifierHub cardVerifierHub = new(16);

if (cardVerifierHub.VerifyCard("abc"))

{

Console.WriteLine("Card Valid");

}

else {

Console.WriteLine("Card not Valid");

}

}

}

}

CardVerifierHub.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace CardVerifier

{

internal class CardVerifierHub

{

public int Length { get; set; }

public int[] Digits { get; set; }

public CardVerifierHub (int length) {

Length = length;

Digits = new int[length];

PopulateDigits ();

}

private void PopulateDigits () {

string cardNumber = string.Empty;

do

{

Console.WriteLine("Enter card number");

cardNumber = Console.ReadLine();

}

while (!(cardNumber.Length == Length) && !isDigits(cardNumber));

cardNumber = new string(cardNumber.ToCharArray().Reverse().ToArray());

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

{

Digits[i] = int.Parse(cardNumber[i].ToString());

}

}

private void DoubleEvenPlacesStep() {

for (int i = 1; i < Length; i+=2)

{

Digits[i] \*= 2;

}

}

private void ReduceEvenPlaceToDigitSumStep() {

for (int i = 1; i<Length; i+=2)

{

if (Digits[i].ToString().Length >= 2) {

int num = Digits[i];

int sum = 0;

while (num != 0)

{

sum += num % 10;

num /= 10;

}

Digits[i] = sum;

}

}

}

private static int GetSumOfArr(int[] arr) {

int sum = 0;

foreach (int i in arr) {

sum += arr[i];

}

return sum;

}

/// <summary>

/// Verifies the Card with supported algorithms

/// </summary>

/// <param name="algorithm">Algorithm name 'abc'</param>

/// <returns>bool whether the card is valid or not</returns>

public bool VerifyCard(string algorithm) {

bool isCardValid = false;

if (algorithm == "abc") {

DoubleEvenPlacesStep();

ReduceEvenPlaceToDigitSumStep();

isCardValid = GetSumOfArr(Digits) % 10 == 0;

}

return isCardValid;

}

private static bool isDigits(string digit) {

bool containsNumbers = true;

if (!Regex.IsMatch(digit, @"/d"))

{

containsNumbers = false;

}

return containsNumbers;

}

}

}

Output



