Weekend Assignment

Sunday, January 21, 2024 4:48 PM

1. what is .NET and why we need to use .NET ?

Ans :

.NET is an open source framework. It is developed by Microsoft to build desktop, web and mobile applications. It support 70+ languages like C++, C#, Java, Python, F#, Visual Basic etc.

2. How many versions are there in .NET framework as of today ?

Ans :

.NET Framework 2.0: Released in 2005.
.NET Framework 3.0: Released in 2006.
.NET Framework 3.5: Released in 2007.
.NET Framework 4.0: Released in 2010.
.NET Framework 4.5: Released in 2012.
.NET Framework 4.5.1, 4.5.2: Released in subsequent years.
.NET Framework 4.6: Released in 2015.
.NET Framework 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2: Released in subsequent years.
.NET Core 1.0: Released in 2016.
.NET Core 1.1: Released in 2016.
.NET Core 2.0: Released in 2017.
.NET Core 2.1: Released in 2018.
.NET Core 2.2: Released in 2018.
.NET Core 3.0: Released in 2019.
.NET Core 3.1: Released in 2019.
.NET 5: Released in 2020 (this version marked the convergence of .NET Core and .NET Framework).
.NET 6: Released in 2021.
.NET 7: 8 November 2022
.NET 8: 14 November 2023

3. What is Base class libraries and use of them ?

Ans :

Base Class Libraries is the subpart of the framework that provide the library support to the Common Language runtime. It is a collection of classes, interfaces, and value types that provide a wide range of functionalities essential for building various types of applications.

4. What are the namespaces ?

Ans :

A namespace is a way to organize and encapsulate related classes, structures, interfaces etc. It helps in avoiding naming conflicts. Namespaces are used to group logically related code and to prevent naming collisions between different parts of a program or between different libraries.

It provide hierarchical procedure of accessing any class or method.

For example :

System.Console.WriteLine();

Here,
System is Namespace
Console is Class
WriteLine is Method

5. What type of applications we can develop using .NET ?

Ans :

With the help of .NET we can develop various types of applications like Desktop Applications, Web Applications, Mobile Applications, Games, Cloud Applications, Command Line Application etc.

6. What is front-end, back-end ?

Ans :

Front-end and back-end are two components of application.

Frontend is known as client-side application that provide direct interaction between user and application. In the web application frontend uses HTML, CSS and JavaScript.

Backend is known as the server-side application. This is the part of application where user can't interact directly. In the backend basically involves database and servers.

7. What are constructors ?

Ans :

In object oriented programming constructors are the special types of method which have not any return type and it will be called automatically when we create an object of the class. Constructors have the same name as the class name. The purpose of a constructor is to initialize the object's state, set initial values for its properties, and perform any necessary setup tasks.

8. Build a class with the following

- 2 properties which are readonly

- 2 methods which are overloaded

Program :

```
public class Patient
{
    // Properties
    public string Name { get; }
    public int Age { get; }

    // Constructor to initialize patient information
    public Patient(string patientName, int patientAge)
    {
        Name = patientName;
        Age = patientAge;
    }

    // Display basic patient information
    public void DisplayBasicInfo()
    {
        Console.WriteLine($"Patient Name: {Name}, Age: {Age}");
    }

    // Display patient information with additional medical details
    public void DisplayMedicalInfo(string medicalCondition)
    {
        Console.WriteLine($"Patient Name: {Name}, Age: {Age}, Medical Condition:
{medicalCondition}");
    }
}

class HospitalProgram
{
    static void Main()
    {
        // Create a patient instance
        Patient patient1 = new Patient("John Doe", 45);

        // Display basic patient information
        Console.WriteLine("Basic Patient Information:");
        patient1.DisplayBasicInfo();

        // Display patient information with medical details
        Console.WriteLine("\nPatient Information with Medical Condition:");
        patient1.DisplayMedicalInfo("Undergoing treatment for a respiratory
infection.");
    }
}
```

9. Build a derived class where in

- override the base class method functionality

- Program which can call base class method and derived class method and check the output

Program :

```
using System;

// Base class
public class Person
{
    // Properties
    public string Name { get; }
    public int Age { get; }

    // Constructor to initialize person information
    public Person(string name, int age)
    {
        Name = name;
        Age = age;
    }

    // Display basic person information
    public virtual void DisplayInfo()
    {
        Console.WriteLine($"Person Name: {Name}, Age: {Age}");
    }
}

// Derived class
public class Patient : Person
{
    // Properties specific to Patient
    public string MedicalCondition { get; }

    // Constructor to initialize patient information
    public Patient(string name, int age, string medicalCondition)
        : base(name, age)
    {
        MedicalCondition = medicalCondition;
    }

    // Override the base class method to include medical details
    public override void DisplayInfo()
    {
        base.DisplayInfo(); // Call the base class method
        Console.WriteLine($"Medical Condition: {MedicalCondition}");
    }
}

class HospitalProgram
{
    static void Main()
    {
        // Create a person instance
        Person person = new Person("John Doe", 30);

        // Call the base class method
        Console.WriteLine("Calling base class method for a person:");
        person.DisplayInfo();
        Console.WriteLine();

        // Create a patient instance
        Patient patient = new Patient("Alice Smith", 45, "Undergoing treatment for a
respiratory infection.");

        // Call the derived class method
        Console.WriteLine("Calling derived class method for a patient:");
        patient.DisplayInfo();
    }
}
```

10. Buid a class with the following

- 2 properties which are integer types

- 1 property should accept only values which are postives

- 1 property should accept only values which are negatives

Program :

```
using System;

public class NumberProperties
{
    private int positiveNumber;
    private int negativeNumber;

    // Property for positive values
    public int PositiveNumber
    {
        get { return positiveNumber; }
        set
        {
            if (value >= 0)
            {
                positiveNumber = value;
            }
            else
            {
                Console.WriteLine("Error: PositiveNumber must be a non-negative
value.");
            }
        }
    }

    // Property for negative values
    public int NegativeNumber
    {
        get { return negativeNumber; }
        set
        {
            if (value < 0)
            {
                negativeNumber = value;
            }
            else
            {
                Console.WriteLine("Error: NegativeNumber must be a negative value.");
            }
        }
    }
}

class Program
{
    static void Main()
    {
        // Create an instance of the class
        NumberProperties numbers = new NumberProperties();

        // Set positive and negative values
        numbers.PositiveNumber = 42;
        numbers.NegativeNumber = -10;

        // Try setting invalid values
        numbers.PositiveNumber = -5; // This should display an error message
        numbers.NegativeNumber = 8;  // This should display an error message
    }
}
```

11. Write a program to accept a string and check whether its palindrome.

Program :

```
internal class Program
{
    static bool IsPalindrome(string str)
    {
        // Remove spaces and convert to lowercase for a case-insensitive check
        string cleanedStr = str.Replace(" ", "").ToLower();

        int length = cleanedStr.Length;

        for (int i = 0; i < length / 2; i++)
        {
            if (cleanedStr[i] != cleanedStr[length - 1 - i])
            {
                return false;
            }
        }

        return true;
    }

    private static void Main(string[] args)
    {
        Console.Write("Enter a string to check whether it's a palindrome or not: ");

        string input = Console.ReadLine();

        if (IsPalindrome(input))
        {
            Console.WriteLine($"{input} is a palindrome.");
        }
        else
        {
            Console.WriteLine($"{input} is not a palindrome.");
        }
    }
}
```

12. Write a program where in a sentence is accepted and each word's first alphabet to be converted to upper case

eg: Input: welcome to simplify3x software development company

Output: Welcome To Simplify3x Software Development Company

Program:

```
internal class Program
{
    static string CapitalizeFirstLetterOfWord(string word)
    {
         char[] letters = word.ToCharArray();
        letters[0] = char.ToUpper(letters[0]);
        return new string(letters);
    }

    static string CapitalizeFirstLetterOfEachWord(string input)
    {
        string[] words = input.Split(' ');

        for (int i = 0; i < words.Length; i++)
        {
               words[i] = CapitalizeFirstLetterOfWord(words[i]);

        }

        return string.Join(" ", words);
    }

    private static void Main(string[] args)
    {
        Console.Write("Enter a sentence: ");
        string inputSentence = Console.ReadLine();

        string capitalizedSentence =
CapitalizeFirstLetterOfEachWord(inputSentence);

        Console.Write("Output: ");
        Console.WriteLine(capitalizedSentence);
    }
}
```

13. Write a program where in a sentence is accepted and each words alternative character needs to be removed

eg: Input: welcome to simplify3x software development company

Output: wloe t smlf3 sfwr dvlpet cmay

Program :

```
    internal class Program
    {
        static string RemoveChar(string word)
        {

            string temp = "";
            for (int i = 0; i < word.Length; i++)
            {
                if (i % 2 == 0)
                {
                    temp += word[i];
                }
            }

            return temp;
        }

        static string AlternativeChar(string input)
        {
            string[] words = input.Split(' ');

            for (int i = 0; i < words.Length; i++)
            {
                   words[i] = RemoveChar(words[i]);

            }

            return string.Join(" ", words);
        }

        private static void Main(string[] args)
        {
            Console.Write("Enter a sentence: ");
            string inputSentence = Console.ReadLine();
            string alternativeSentence = AlternativeChar(inputSentence);

            Console.Write("Output: ");
            Console.WriteLine(alternativeSentence);
        }
    }
}
```

14. write a program and take 2 classes (1 base class and 1 derived class) and each class should

have overload and override methods and call them and it should work appropriately

Program :

```
using System;

class MathOperation
{
    public int Add(int a, int b)
    {
        return a + b;
    }

    public double Add(double a, double b)
    {
        return a + b;
    }

    public virtual void DisplayMessage()
    {
        Console.WriteLine("MathOperation - Base Class Message");
    }
}

class AdvancedMathOperation : MathOperation
{
    public int Multiply(int a, int b)
    {
        return a * b;
    }

    public double Multiply(double a, double b)
    {
        return a * b;
    }

    public override void DisplayMessage()
    {
        Console.WriteLine("AdvancedMathOperation - Derived Class Message");
    }
}

class Program
{
    static void Main()
    {
        MathOperation mathObj = new MathOperation();
        AdvancedMathOperation advancedMathObj = new
AdvancedMathOperation();

        // Overload methods from the base class
        Console.WriteLine("Overloaded methods from MathOperation (Base
Class):");
        Console.WriteLine($"Add(int): {mathObj.Add(5, 3)}");
        Console.WriteLine($"Add(double): {mathObj.Add(2.5, 3.5)}");
        Console.WriteLine();

        // Override methods in the derived class
        Console.WriteLine("Overridden methods from AdvancedMathOperation
(Derived Class):");
        Console.WriteLine($"Add(int): {advancedMathObj.Add(5, 3)}");
        Console.WriteLine($"Add(double): {advancedMathObj.Add(2.5, 3.5)}");
        Console.WriteLine($"Multiply(int): {advancedMathObj.Multiply(5, 3)}");
        Console.WriteLine($"Multiply(double): {advancedMathObj.Multiply(2.5, 3.5)}");
        Console.WriteLine();

        // Call overridden method from the base class
        Console.WriteLine("Overridden method from the base class:");
        mathObj.DisplayMessage();
        Console.WriteLine();

        // Call overridden method from the derived class
        Console.WriteLine("Overridden method from the derived class:");
        advancedMathObj.DisplayMessage();
    }
}
```

15. Write a program to check whether a date will fall in which week of the year

eg: Input: 19/1/2024

Output: 3rd week

Program :

```
    internal class Program
    {
        private static bool IsValidDate(string inputDate)
        {
            string[] date = inputDate.Split('/');
            if (date.Length != 3)
            {
                return false;
            }

            int day = Convert.ToInt32(date[0]);
            int month = Convert.ToInt32(date[1]);
            int year = Convert.ToInt32(date[2]);

            return IsValidDayMonthYear(day, month, year);
        }

        private static bool IsValidDayMonthYear(int day, int month, int year)
        {
            if (year < 1 || month < 1 || month > 12)
            {
                return false;
            }

            int[] daysInMonth = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

            if (IsLeapYear(year))
            {
                daysInMonth[2] = 29;
            }

            return day >= 1 && day <= daysInMonth[month];
        }

        private static bool IsLeapYear(int year)
        {
            return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
        }

        private static int GetWeekNumber(string inputDate)
        {
            string[] date = inputDate.Split('/');
            int day = Convert.ToInt32(date[0]);
            int month = Convert.ToInt32(date[1]);
            int year = Convert.ToInt32(date[2]);

            int[] daysInMonth = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

            if (IsLeapYear(year))
            {
                daysInMonth[2] = 29;
            }

            int totalDays = day;
            for (int i = 1; i < month; i++)
            {
                totalDays += daysInMonth[i];
            }

            int weekNumber = (totalDays - 1) / 7 + 1;

            return weekNumber;
        }

        static  void Main()
        {
            Console.Write("Enter a date (in the format dd/MM/yyyy): ");
            string inputDate = Console.ReadLine();

            if (IsValidDate(inputDate))
            {
                int weekNumber = GetWeekNumber(inputDate);
                Console.WriteLine($"Output: {weekNumber} week");
            }
            else
            {
                Console.WriteLine("Invalid date format. Please enter a valid date in the
format dd/MM/yyyy.");
            }
        }
    }
```