PRACTICA No 1

|  |  |
| --- | --- |
| Materia: Actualización Tecnológica SIS2420 "A" Docente:  Nombre: ricky ricardo revollo espinoza  Ing. Saul Mamani M.  Auxiliar: Egr. William Mucio Achabal Villalpando | Semestre: 1/2024  Fecha de Emisión: 27/04/2024  Fecha de Entrega: Primer Parcial |

Parte Teórica 30 pts.

EXPLICAR EN SUS PALABRAS CADA PREGUNTA

l. ¿Qué es un sistema?

Un sistema es un conjunto de elementos interrelacionados que trabajan juntos para lograr un objetivo común.

2. ¿Cuál es la diferencia entre una clase abstracta y una clase estática en 

Una clase abstracta se usa para definir una estructura común para clases derivadas y permite la herencia y la implementación de métodos abstractos, mientras que una clase estática se utiliza para agrupar miembros estáticos que son compartidos por todas las instancias de la aplicación y no requieren instanciación.

3. ¿Cuáles son las diferencias enfre la herencia y el polimorfismo en C#?

la herencia se centra en la relación de "es un" entre clases, mientras que el polimorfismo se centra en la capacidad de un método o propiedad para comportarse de diferentes maneras en función del contexto de ejecucion

4)

1. git init: Inicializa un repositorio Git en un directorio vacío o convierte un directorio existente en un repositorio Git.

2. Git status: Muestra el estado actual del directorio de trabajo y del área de preparación (staging).

3. git add: Agrega cambios al área de preparación (staging) para que puedan ser incluidos en el próximo commit.

4. Git commit -m "mensaje": Registra los cambios confirmados en el repositorio junto con un mensaje descriptivo.

5. git log: Muestra un registro de commits.

6. git checkout: Permite cambiar entre ramas o restaurar archivos.

7. git checkout -b NombreRama: Crea una nueva rama y se cambia a ella.

8. Git branch: Lista, crea o elimina ramas.

9. git push: Sube los commits locales a un repositorio remoto.

10. git pull: Obtiene cambios del repositorio remoto y los fusiona con la rama local actual.

11. git fetch: Descarga los últimos cambios del repositorio remoto, pero no los fusiona con la rama local actual.

12. git switch: Cambia entre ramas.

13. git merge: Fusiona una rama con otra.

14. git clone: Clona un repositorio Git existente en un nuevo directorio.

5¿Qué es un Requerimiento Funcional y No Funcional?

Requisitos Funcionales:

- Son las funciones o características específicas que el sistema debe realizar.

- Describen lo que el sistema debe hacer en términos de entradas, salidas y comportamiento esperado.

Requisitos No Funcionales:

- Son las restricciones o cualidades no relacionadas con la funcionalidad que el sistema debe cumplir.

- Se centran en aspectos como el rendimiento, la seguridad, la usabilidad, la escalabilidad, la fiabilidad, la disponibilidad, etc.

6¿Qué es SCRUM y cuáles son sus respectivos roles?

Scrum es un marco de trabajo ágil utilizado en el desarrollo de software y en proyectos que requieren flexibilidad y adaptabilidad. Sus roles principales son:

Product Owner

Scrum Master

Equipo de Desarrollo

Parte practico

1)

using System;

public static class NumerosALetras

{

private static string[] unidades = {"cero", "uno", "dos", "tres", "cuatro", "cinco", "seis", "siete", "ocho", "nueve"};

private static string[] especiales = {"diez", "once", "doce", "trece", "catorce", "quince", "dieciséis", "diecisiete", "dieciocho", "diecinueve"};

private static string[] decenas = {"", "diez", "veinte", "treinta", "cuarenta", "cincuenta", "sesenta", "setenta", "ochenta", "noventa"};

private static string[] centenas = {"", "ciento", "doscientos", "trescientos", "cuatrocientos", "quinientos", "seiscientos", "setecientos", "ochocientos", "novecientos"};

public static string ConvertirALetras(int numero)

{

if (numero < 10)

{

return unidades[numero];

}

else if (numero < 20)

{

return especiales[numero - 10];

}

else if (numero < 100)

{

int decena = numero / 10;

int unidad = numero % 10;

if (unidad == 0)

{

return decenas[decena];

}

else

{

return decenas[decena] + " y " + unidades[unidad];

}

}

else

{

int centena = numero / 100;

int decena = (numero % 100) / 10;

int unidad = numero % 10;

if (decena == 0 && unidad == 0)

{

return centenas[centena];

}

else

{

return centenas[centena] + " " + ConvertirALetras(numero % 100);

}

}

}

}

class Program

{

static void Main(string[] args)

{

Console.Write("Numero: ");

int numero = Convert.ToInt32(Console.ReadLine());

string resultado = NumerosALetras.ConvertirALetras(numero);

Console.WriteLine("Salida:");

Console.WriteLine(resultado);

}

}

2)

using System;

class Program

{

static void Main(string[] args)

{

Console.Write("Ingrese un número: ");

int numero = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Salida:");

Console.WriteLine("Serie 1: " + GenerarSerie1(numero));

Console.WriteLine("Serie 2: " + GenerarSerie2(numero));

Console.WriteLine("Serie 3: " + GenerarSerie3(numero));

}

static string GenerarSerie1(int n)

{

string serie = "";

int resultado = 1;

for (int i = 1; i <= n; i++)

{

resultado \*= i;

serie += resultado + ", ";

}

return serie.TrimEnd(',', ' ');

}

static string GenerarSerie2(int n)

{

string serie = "";

for (int i = 1; i <= n; i++)

{

serie += (i \* 3) + ", ";

}

return serie.TrimEnd(',', ' ');

}

static string GenerarSerie3(int n)

{

string serie = "";

for (int i = 1; i <= n; i++)

{

serie += (i \* 6) + ", ";

}

return serie.TrimEnd(',', ' ');

}

}

3)

using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Ingrese 10 números separados por comas:");

string entrada = Console.ReadLine();

List<int> numeros = entrada.Split(',').Select(int.Parse).ToList();

// Ordenar la lista utilizando expresiones lambda

List<int> listaOrdenadaLambda = numeros.OrderBy(x => x).ToList();

Console.WriteLine("Lista ordenada por expresiones lambda: " + string.Join(",", listaOrdenadaLambda));

// Ordenar la lista utilizando consultas LINQ

List<int> listaOrdenadaLinq = (from num in numeros

orderby num

select num).ToList();

Console.WriteLine("Lista ordenada por consultas LINQ: " + string.Join(",", listaOrdenadaLinq));

// Encontrar el menor y el mayor número en la lista

int menor = numeros.Min();

int mayor = numeros.Max();

Console.WriteLine("Menor: " + menor);

Console.WriteLine("Mayor: " + mayor);

}

}