

PRACTICA N1

Facultad Nacional

De Ingeniería

Universidad Técnica

de Oruro

**CI:** 7399800 Or.

**Auxiliar:** William Mucio Achabal Villalpando

**Fecha de entrega:** 17/04 /2023

**Estudiante:** Leon Quiñones Alain Guido

**Materia:** SIS-2420”B”

**Docente:** Ing. Saul Mamani Mamani

**Ejercicio 1.**

**1.1 R.-** Un sistema es un conjunto de elementos que trabajan juntos para realizar un objetivo o tarea especifica.

**1.2 R.-** Una clase es una plantilla o modelo que define un conjunto de atributos (datos) y métodos (funciones) que se pueden utilizar para crear objetos.

Una clase abstracta se utiliza para definir una plantilla para otras clases, mientras que una clase estática se utiliza para agrupar funcionalidades relacionadas que no necesitan tener estado y que se pueden acceder sin tener que crear objetos.

**1.3 R.-** Ambos herencia y polimorfismo son conceptos relacionados pero su uso y propósito es diferente.

El polimorfismo se utiliza para hacer que un objeto pueda actuar de manera diferente en función del contexto en el que se use.

La herencia se utiliza para poder heredar atributos y métodos de una clase a otra.

**1.4 R.-** Es un proceso estructurado y sistemático utilizado en la industria del software para diseñar, desarrollar y mantener aplicaciones de software. Es un marco de trabajo que describe las distintas fases y actividades involucradas en el desarrollo de software, desde la concepción de la idea hasta la implementación y mantenimiento del sistema.

**1.5 R.-**

**Git init:** sirve para inicializar un nuevo repositorio de Git en un directorio existente.

**Git status:** sirve para inicializar un nuevo repositorio de Git en un directorio existente.

**Git add:** agrega archivos al área de antes de confirmar los cambios.

**Git commit –m “Mensaje”:** se utiliza para guardar los cambios realizados en un repositorio de Git.

**Git log:** muestra un registro detallado de los commits(confirmaciones) realizados en un repositorio de Git.

**Git checkout:** se utiliza principalmente para cambiar entre ramas (branches) en un repositorio de Git.

**Git branch:** se utiliza para listar, crear y eliminar ramas (branches) en un repositorio de Git.

**Git push:** se utiliza para enviar los cambios realizados en un repositorio local a un repositorio remoto.

**Git pull:** se utiliza para actualizar el repositorio local con los cambios realizados en el repositorio remoto.

**Git clone:** se utiliza para crear una copia local de un repositorio remoto en la máquina local.

**1.6 R.-** La metodología tradicional se enfoca en una planificación detallada, una estructura jerárquica y una entrega única, mientras que la metodología ágil se enfoca en una planificación flexible, entregas incrementales y una colaboración cercana del equipo de trabajo.

**1.7 R.-**

**Metodologías tradicionales:**

1. Modelo de Ciclo de Vida en Cascada (Waterfall): como ya se mencionó, esta metodología se enfoca en una planificación detallada, una estructura jerárquica y una entrega única.
2. Modelo V: es una variante del modelo en cascada que enfatiza en la relación entre las pruebas y el desarrollo de software.
3. Modelo Iterativo: esta metodología se enfoca en realizar una serie de iteraciones para construir el producto final, cada iteración se planifica y ejecuta como un proyecto en miniatura.
4. Modelo Espiral: esta metodología es una combinación de la planificación en cascada y la iteración, y se centra en la evaluación continua del proyecto para identificar riesgos y oportunidades de mejora.
5. Modelo de Desarrollo en Prototipos: esta metodología se enfoca en la creación de prototipos para validar la solución, antes de comenzar el desarrollo completo del producto.

**Metodologías ágiles:**

1. Scrum: es una metodología ágil que se enfoca en el trabajo en equipo, entregas incrementales, reuniones diarias y una planificación flexible.
2. Kanban: esta metodología se enfoca en el flujo de trabajo y la limitación del trabajo en proceso (WIP) para evitar el exceso de trabajo y los cuellos de botella.
3. XP (Programación Extrema): es una metodología ágil que se enfoca en la calidad del software, la comunicación del equipo y la adaptabilidad a los cambios.
4. Crystal: esta metodología se enfoca en la colaboración, la comunicación y la entrega rápida de software de alta calidad.
5. Lean: esta metodología se enfoca en la eliminación de desperdicios, la optimización del flujo de trabajo y la entrega continua de valor al cliente.

**Ejercicio 2.1**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

public static class NumeroLiteral

{

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

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

private static string[] decenas2 = { "", "", "veinti", "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 < 0 || numero > 999)

throw new ArgumentException("El número debe estar entre 0 y 999");

if (numero == 0)

return unidades[numero];

string letras = "";

if (numero >= 100)

{

if (numero == 120)

{

letras = "ciento veinte";

numero %= 10;

}

else

{

int centena = numero / 100;

letras = centenas[centena] + " ";

numero %= 100;

}

}

if (numero >= 10 && numero <= 19)

{

letras += decenas[numero - 10] + " ";

return letras.Trim();

}

if (numero >= 20 && numero < 30)

{

if (numero == 20)

{

letras = "veinte";

numero %= 10;

}

else

{

int decena = numero / 10;

letras += decenas2[decena];

numero %= 10;

}

}

if (numero >= 30)

{

int decena = numero / 10;

letras += decenas2[decena] + " y ";

numero %= 10;

}

if (numero > 0)

{

letras += unidades[numero];

}

return letras.Trim();

}

}

**Ejercicio 2.2.-**

internal class Program

{

public static void Main()

{

Console.WriteLine("Ingrese un número entre 0 y 999:");

int numero = int.Parse(Console.ReadLine());

string literal = NumeroLiteral.ConvertirALetras(numero);

Console.WriteLine("El número {0} se escribe como: {1}", numero, literal);

Console.ReadKey();

}

}

}

internal class Program

{

static void Main(string[] args)

{

Numero.GenerarSecuencia1(5);

Numero.GenerarSecuencia2(5);

Console.ReadKey();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

public static class Numero

{

public static void GenerarSecuencia1(int n)

{

int x = 2;

string cad = "";

int num = 1;

cad = cad + num + " ";

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

{

num = num \* x;

cad = cad + num + " ";

x++;

}

Console.WriteLine(cad);

}

public static void GenerarSecuencia2(int n)

{

int num = 2;

string cad = "";

cad = cad + num + " ";

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

{

num = num \* 2;

cad = cad + num + " ";

}

Console.WriteLine(cad);

}

}

internal class Program

{

static void Main(string[] args)

{

Numero.GenerarSecuencia1(5);

Numero.GenerarSecuencia2(5);

Console.ReadKey();

}

}

}

**Ejercicio 2.3.-**

|  |
| --- |
| **HU1: Registrar ventas** |
| Como: vendedor  Quiero: poder registrar mis ventas de manera digital  Para: un manejo mas fácil de los registros de mis ventas |
| **HU2: Registrar clientes** |
| Como: vendedor  Quiero: registrar y buscar los datos de mis clientes  Para: facilitarme la ubicación y contacto de los mismos |
| **HU3: Imprimir y emitir recibos de venta** |
| Como: vendedor  Quiero: poder imprimir y emitir recibos de venta  Para: ahorrarme tiempo de hacerlo manualmente |
| **HU4: Control total del sistema al gerente** |
| Como: gerente  Quiero: tener control total del sistema  Para: mantenerme informado sobre la misma |
| **HU5: ver los reportes de ventas** |
| Como: gerente  Quiero: ver los reportes de ventas por dia por mes y por vendedor  Para: ver el progreso de las ventas |

|  |  |
| --- | --- |
| **Prioridad** | **Historias de Usuario** |
| 1 | Control total del sistema al gerente |
| 1 | Registrar Clientes |
| 1 | Registrar ventas |
| 2 | Imprimir y emitir recibos de ventas |
| 3 | Ver los reportes de ventas |

**Ejercicio 2.4 y 2.5.-**

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

namespace ejercicio4

{

internal class Program

{

static void Main(string[] args)

{

List<Mascota\_adoptada> Listamascotas = new List<Mascota\_adoptada>();

//AGREGANDO

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "Duque",

Edad = 10,

Raza = "pastor\_aleman",

Sexo = "macho",

Nombre\_Dueño = "Alejandro",

Pago\_Mascota = 100,

Fecha\_Adopcion = DateTime.Parse("10/06/2013")

});

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "Oso",

Edad = 5,

Raza = "mestizo",

Sexo = "macho",

Nombre\_Dueño = "Miguel",

Pago\_Mascota = 20,

Fecha\_Adopcion = DateTime.Parse("11/06/2018")

});

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "Linda",

Edad = 3,

Raza = "Cocker",

Sexo = "hembra",

Nombre\_Dueño = "Esteban",

Pago\_Mascota = 200,

Fecha\_Adopcion = DateTime.Parse("23/05/2020")

});

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "Cachuchin",

Edad = 7,

Raza = "Chapi",

Sexo = "macho",

Nombre\_Dueño = "Selena",

Pago\_Mascota = 100,

Fecha\_Adopcion = DateTime.Parse("18/10/2016")

});

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "Rex",

Edad = 10,

Raza = "Golden",

Sexo = "macho",

Nombre\_Dueño = "Jhoana",

Pago\_Mascota = 1000,

Fecha\_Adopcion = DateTime.Parse("25/03/2013")

});

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "Dana",

Edad = 2,

Raza = "Chihuahua",

Sexo = "femenino",

Nombre\_Dueño = "Jhoana",

Pago\_Mascota = 100,

Fecha\_Adopcion = DateTime.Parse("17/05/2021")

});

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "Pria",

Edad = 1,

Raza = "cocker",

Sexo = "femenino",

Nombre\_Dueño = "Isabel",

Pago\_Mascota = 300,

Fecha\_Adopcion = DateTime.Parse("13/01/2022")

});

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "Satan",

Edad = 8,

Raza = "Rottweiler",

Sexo = "macho",

Nombre\_Dueño = "Matias",

Pago\_Mascota = 2000,

Fecha\_Adopcion = DateTime.Parse("23/04/2015")

});

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "Juli",

Edad = 6,

Raza = "cocker",

Sexo = "femenino",

Nombre\_Dueño = "Anabel",

Pago\_Mascota = 150,

Fecha\_Adopcion = DateTime.Parse("09/09/2018")

});

Listamascotas.Add(new Mascota\_adoptada()

{

Nombre\_Mascota = "rocky",

Edad = 10,

Raza = "mestizo",

Sexo = "macho",

Nombre\_Dueño = "Alain",

Pago\_Mascota = 20,

Fecha\_Adopcion = DateTime.Parse("29/11/2013")

});

//expresiones lambda

//prom de edad de mascotas

Prom\_Mascota(Listamascotas);

//prom de pago de mascotas

Prom\_Pago(Listamascotas);

//consultas linq

//mostrar Raza chihuahua y femenino

Raza\_chiwawa(Listamascotas);

//mostrar mascotas menores a 2 años

Edad\_mascota(Listamascotas);

Console.ReadKey();

}

private static void Edad\_mascota(List<Mascota\_adoptada> listamascotas)

{

var mostrar = from Mascota\_adoptada in listamascotas

where Mascota\_adoptada.Edad < 2

select Mascota\_adoptada;

Console.WriteLine("Mascotas menores a 2 años son: ");

foreach (var Mascota\_Adoptada in mostrar)

{

Console.WriteLine("El nombre de la mascota es: " + Mascota\_Adoptada.Nombre\_Mascota + " su edad es: " + Mascota\_Adoptada.Edad + " su raza es: " + Mascota\_Adoptada.Raza + " su sexo es: " + Mascota\_Adoptada.Sexo);

}

}

private static void Raza\_chiwawa(List<Mascota\_adoptada> listamascotas)

{

var mostrar = from Mascota\_adoptada in listamascotas

where Mascota\_adoptada.Raza == "Chihuahua" && Mascota\_adoptada.Sexo == "femenino"

select Mascota\_adoptada;

Console.WriteLine("Mascotas de raza Chihuahua y de sexo femenino son: ");

foreach (var Mascota\_Adoptada in mostrar)

{

Console.WriteLine("El nombre de la mascota es: " + Mascota\_Adoptada.Nombre\_Mascota + " su edad es: " + Mascota\_Adoptada.Edad + " su raza es: " + Mascota\_Adoptada.Raza + " su sexo es: " + Mascota\_Adoptada.Sexo);

}

}

private static void Prom\_Pago(List<Mascota\_adoptada> listamascotas)

{

var Mostrar = listamascotas.Average(x => x.Pago\_Mascota);

Console.WriteLine("El promedio por el pago de las mascotas es: " + Mostrar);

}

private static void Prom\_Mascota(List<Mascota\_adoptada> listamascotas)

{

var Mostrar = listamascotas.Average(Mascota\_adoptada => Mascota\_adoptada.Edad);

Console.WriteLine("el promedio de edad de las mascotas es: " + Mostrar);

}

}

}

Mascota.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ejercicio4

{

internal class mascota

{

//ATRIBUTOS

public string Nombre\_Mascota { get; set; }

public int Edad { get; set; }

public string Raza { get; set; }

public string Sexo { get; set; }

//FUNCIONES

public void Mostrar\_Mascota() {

Console.WriteLine("El nombre de la mascota es: " + Nombre\_Mascota + "su edad es: " + Edad + "su raza es: " + Raza + "su sexo es: " + Sexo);

}

}

}

Mascota\_adoptada.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Timers;

namespace ejercicio4

{

internal class Mascota\_adoptada : mascota

{

//ATRIBUTOS

public string Nombre\_Dueño { get; set; }

public float Pago\_Mascota { get; set; }

public DateTime Fecha\_Adopcion { get; set; }

//FUNCIONES

public void Mostrar\_Mascota()

{

Console.WriteLine("El nombre del dueño es: "+Nombre\_Dueño+"el pago por la mascota es: "+Pago\_Mascota+"la fecha de adopcion es: "+Fecha\_Adopcion);

}

}

}

Mascota\_perdida.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ejercicio4

{

internal class Mascota\_Perdida : mascota

{

//ATRIBUTOS

public string Motivo { get; set; }

public DateTime Fecha\_Perdida { get; set; }

//FUNCIONES

public void Mostrar\_Mascota()

{

Console.WriteLine("Se perdio por: "+Motivo+" en fecha: "+Fecha\_Perdida);

}

}

}

**Ejercicio 2.6.-**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public int n{ get; set; }

public Form1()

{

InitializeComponent();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

bool esPrimo(int num)

{

if (num <= 1)

{

return false;

}

}

return true;

}

void mostrarPrimos(int n)

{

int contador = 0;

int num = 2;

while (contador < n)

{

if (esPrimo(num))

{

listBox1.Items.Add(num);

contador++;

}

num++;

}

}

void GenerarNumerosPerfectos(int n)

{

int cont = 1;

int num = 2;

while (cont <= n)

{

if (EsNumeroPerfecto(num) == 1)

{

listBox1.Items.Add(num);

cont++;

}

num++;

}

}

int EsNumeroPerfecto(int numero)

{

int cont = 0;

int i = 1;

while (i < n)

{

if (numero % i == 0)

{

cont = cont + i;

}

}

if (i == numero) return 1;

else return 0;

}

public void Fibonacci(int n)

{

int a = 0, b = 1, c;

for (int i = 0; i <= n; i++)

{

if (i <= 1)

{

c = i;

}

else

{

c = a + b;

a = b;

b = c;

}

if (c != 0)

{

listBox1.Items.Add(c);

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

n = Convert.ToInt32(textBox1.Text);

if (comboBox1.SelectedItem == "Numeros Primos")

{

mostrarPrimos(n);

}

if (comboBox1.SelectedItem == "Numeros Perfectos")

{

GenerarNumerosPerfectos(n);

}

if (comboBox1.SelectedItem == "Serie Fibonacci")

{

Fibonacci(n);

}

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}