PRACTICA N.º 1

Nombre: Burgos Mamani Iber Nelson **Materia**: Actualización Tecnológica SIS 2420 “A”

PARTE TEORICA.

1-. ¿Qué es un sistema?

es un marco conceptual que describe cómo un conjunto de componentes, operaciones y procesos se organizan para tomar información de entrada, realizar operaciones en esa información y producir una respuesta o salida.

Entrada: Esta es la parte del sistema donde se recopilan los datos, información o señales del entorno o de otros sistemas.

Proceso: En esta etapa, el sistema lleva a cabo una serie de operaciones, cálculos, transformaciones o acciones sobre la entrada recibida

Salida: La salida es el resultado o los datos procesados que el sistema genera como respuesta a la entrada y al proceso

2. ¿Qué es y qué diferencias tienen una clase abstracta y una clase estática en C#?

Clase Abstracta: Una clase abstracta es una clase que no se puede instanciar por sí sola, es decir, no se pueden crear objetos directamente a partir de ella.

Clase Estática: Una clase estática es una clase que no se puede instanciar en absoluto, y sus miembros (métodos y variables) son accesibles a través de la propia clase, en lugar de a través de instancias de la clase.

la principal diferencia entre una **clase abstracta** y una **clase estática** es que una clase abstracta se utiliza para definir una estructura común y obligar a las clases derivadas a implementar ciertos métodos, mientras que una clase estática contiene miembros que son compartidos y se acceden a través de la propia clase, sin necesidad de crear instancias de la misma.

3. ¿Qué es y qué diferencias tienen la herencia y polimorfismo en C#?

Herencia: La herencia es un mecanismo mediante el cual una clase (llamada clase base o superclase) puede heredar las características y comportamientos de otra clase (llamada clase derivada o subclase).

Polimorfismo: El polimorfismo se refiere a la capacidad de objetos de diferentes clases de actuar de manera uniforme a través de una interfaz común.

la herencia se centra en la relación de jerarquía entre clases y la reutilización de código, mientras que el polimorfismo se centra en la capacidad de objetos de diferentes clases de comportarse de manera similar a través de una interfaz común.

4. ¿Qué es un ciclo de vida del desarrollo de software (SDLC)?

el ciclo de vida del desarrollo de software es un conjunto de etapas y procesos que guían el desarrollo de una aplicación de software desde la concepción hasta su despliegue y mantenimiento. Las etapas típicas incluyen:

Requisitos: Identificación y documentación de lo que debe hacer el software.

Diseño: Creación de la arquitectura y diseño detallado.

Implementación: Codificación del software.

Pruebas: Verificación y corrección de errores.

Despliegue e Instalación: Instalación en el entorno de producción.

Mantenimiento: Actualizaciones y correcciones para garantizar su funcionamiento continuo.

5. Para qué sirven estos comandos de Git:

-Git init: Se utiliza para iniciar un nuevo repositorio de Git en un directorio local.

-Git status: se utiliza para obtener información sobre el estado actual de tu repositorio de Git

-Git add . : Sirve para salvar los cambios (stage)

-Git commit –m "Mensaje": Sirve para salvar los cambios en líneas de tiempo

-Git log: se utiliza en Git para ver el historial de commits en un repositorio.

-Git checkout: tiene varios usos y funcionalidades importantes, y su propósito principal es cambiar el estado del repositorio de Git en diversas formas.

-Git checkout – b NombreRama: Esto es útil cuando estás a punto de comenzar una nueva característica o corrección y quieres crear una rama separada para ello.

-Git Branch: se utiliza para listar, crear y gestionar ramas en un repositorio.

-Git push: se utiliza en Git para enviar (o "empujar") tus cambios locales a un repositorio remoto

-Git pull: se utiliza en Git para actualizar tu copia local de un repositorio desde su versión remota.

-Git merge: se utiliza para combinar los cambios de una rama en otra.

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

6. ¿Cuál es la diferencia entre una metodología tradicional y ágil?

la metodología ágil (**SCRUM**) se centra en la flexibilidad, la colaboración y la adaptación continua, permitiendo a los equipos responder de manera eficiente a los cambios y ofrecer valor al cliente de manera más rápida y frecuente, mientras que la metodología tradicional (**CASCADA**) se basa en una planificación detallada y fases secuenciales. La elección entre estas metodologías depende de las necesidades y las características específicas del proyecto.

5 ejemplos de una metodología tradicional y 5 ejemplos de una metodología tradicional ágil

Metodologías Tradicionales:

Modelo en Cascada (Waterfall): Es un enfoque secuencial donde cada fase (requisitos, diseño, implementación, pruebas, mantenimiento) se completa antes de pasar a la siguiente. Los cambios son difíciles de incorporar una vez que se inicia una fase.

Modelo en V: Similar al modelo en cascada, pero con un énfasis en pruebas que corresponden a cada fase de desarrollo. Las pruebas se realizan en el camino de regreso, formando una "V".

Modelo en Espiral: Combina la planificación y el diseño iterativo con elementos del enfoque secuencial. Cada ciclo incluye cuatro etapas: planificación, análisis de riesgos, ingeniería y evaluación.

Método de Desarrollo Estructurado (SDM): Se centra en la planificación, el análisis y el diseño antes de la implementación. Utiliza diagramas de flujo y técnicas estructuradas para modelar el sistema.

Proceso Unificado (Rational Unified Process - RUP): Un marco de trabajo que utiliza cuatro fases (iniciación, elaboración, construcción y transición) para desarrollar software de manera iterativa. Combina elementos de desarrollo ágil y tradicional.

Metodologías Ágiles:

Scrum: Un enfoque iterativo y colaborativo que se centra en equipos pequeños y autoorganizados. Los proyectos se dividen en sprints (ciclos de desarrollo) y se entregan incrementos funcionales en cada sprint.

Kanban: Se basa en la visualización de tareas y el flujo de trabajo. Las tareas se mueven a través de tableros Kanban, y se enfoca en la limitación del trabajo en progreso (WIP) para mejorar la eficiencia.

Extreme Programming (XP): Se enfoca en la calidad del software a través de prácticas como la programación en parejas, pruebas automatizadas y entregas frecuentes. Los cambios en los requisitos son bienvenidos.

Lean Software Development: Se basa en los principios del lean manufacturing para eliminar el desperdicio en el proceso de desarrollo. Prioriza la entrega rápida de valor al cliente.

Dynamic Systems Development Method (DSDM): Un marco de desarrollo ágil que se centra en la colaboración, la entrega temprana y la adaptabilidad. Incluye una serie de fases y principios para la gestión de proyectos.

8. ¿Qué es un Requerimiento Funcional y No Funcional?

Requerimiento funcional: Un requisito funcional, en el contexto de la ingeniería de software y la gestión de proyectos, es una declaración que describe una función específica que un sistema, software o aplicación debe ser capaz de realizar

Requerimiento no funcional: Un requerimiento no funcional, en el contexto de la ingeniería de software y la gestión de proyectos, es una especificación que describe las cualidades, características y restricciones que no están relacionadas con una función específica del sistema, sino que se refieren a atributos de calidad, características de rendimiento y restricciones del sistema en su conjunto.

9. ¿Qué es SCRUM?

Scrum es un marco de trabajo o framework ágil ampliamente utilizado en la gestión de proyectos y el desarrollo de software. Fue creado inicialmente para gestionar proyectos de desarrollo de software, pero se ha extendido a otros campos debido a su flexibilidad y eficacia. Scrum se basa en principios de colaboración, adaptabilidad y entrega continua de valor.

10. ¿Cuáles son los roles de SCRUM?

Product Owner:

-Dueño del producto

-Entrevista al cliente

-Captura los requisitos funcionales

-Sabe lo que quiere el cliente

-Organiza el product backlog

Scrum Master:

-Lider del equipo scrum

-Conoce bien scrum

-Facilita las herramientas

-Organiza al equipo

Developer team:

-Analisis, diseño e implementación

-QA

-pruebas

PARTE PRACTICA:

1. Realizar un programa utilizando una clase estática que permita ingresar un número por teclado y te muestre en su parte literal.

```
namespace ConsoleApp3
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Numero: ");
```

```
            if (int.TryParse(Console.ReadLine(), out int numero))
            {
                string literal = Numero.Convercion(numero);
                Console.WriteLine("literal: " + literal);
            }
            else
            {
                Console.WriteLine("ingresa un numero valido");
            }
            Console.ReadKey();
        }

    }
}

    public static class Numero
    {
        private static string[] unidad = {
 "", "uno", "dos", "tres", "cuatro", "cinco", "seis", "siete",
"ocho", "nueve"
 };
        private static string[] decena = {
 "", "diez", "veinte", "treinta", "cuarenta", "cincuenta",
"sesenta", "setenta", "ochenta", "noventa"
 };
        private static string[] NumEspecial = {
 "diez", "once", "doce", "trece", "catorce", "quince",
"dieciséis", "diecisiete", "dieciocho", "diecinueve"
 };
        public static string Convercion(int numero)
        {
            if (numero < 10)
            {
                return unidad[numero];
            }
            else if (numero < 20)
            {
                return NumEspecial[numero - 10];
            }
            else
            {
                int unidad = numero % 10;
                int decena = numero / 10;
                return decena[decenas] + (unidades > 0 ? " y " +
               unidad[unidades] : "");
            }
        }

    }
```

2. Realizar un programa utilizando listas que te permita ingresar n números por teclado donde cada número entre en las siguientes listas: Entrada ¿Cuantos números deseas añadir? ¬ 7 Añada 7 números: ¬ 2, 3, 5, 10, 5, 4, 6

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Pregunta_2
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Anhada los numeros: ");
            if (int.TryParse(Console.ReadLine(), out int n))
            {
                List<int> numeros = new List<int>();
                for (int i = 0; i < n; i++)
                {
                    Console.Write("Anhada los numeros [{i + 1}]: ");
                    if (int.TryParse(Console.ReadLine(), out int num))
                    {
                        numeros.Add(num);
                    }
                    else
                    {
                        Console.WriteLine("numero invalido");
                        i--;
                    }

                }
                List<int> multiploDos = numeros.Where(x => x % 2
==0).ToList();
                List<int> primos = numeros.Where(x => Primo(x)).ToList();
                Console.WriteLine("multiplo de dos: " + string.Join(", ",
               multiploDos));
                Console.WriteLine("Primos: " + string.Join(", ", primos));
            }
            else
            {
                Console.WriteLine("Ingrese un numero valido");
            }
        }
        static bool Primo(int numero)
        {
            if (numero <= 1)
                return false;
            for (int i = 2; i * i <= numero; i++)
            {
                if (numero % i == 0)
                    return false;
            }
            return true;
        }

    }

}
```

3. Realizar un programa que tenga las siguientes clases utilizando polimorfismo y herencia. La clase Celular debe ser una clase abstracta.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace ConsoleApp4
{
    internal class CelularController
    {
        List<CelularNuevo> celularesNuevos = new List<CelularNuevo>();
        public void Insertar()
        {
            celularesNuevos.Add(new CelularNuevo()
            {
                Marca = "Samsung",
                Modelo = "Galaxy s4",
                SistemaOperativo = "Android",
                RAM = 3,
                Almacenamiento = 128,
                Precio = 299,
                FechaIngreso = DateTime.Parse("01/08/2020")
            });
            celularesNuevos.Add(new CelularNuevo()
            {
                Marca = "Samsung",
                Modelo = "Galaxy s5",
                SistemaOperativo = "Android",
                RAM = 4,
                Almacenamiento = 128,
                Precio = 399,
                FechaIngreso = DateTime.Parse("01/03/2021")
            });
            celularesNuevos.Add(new CelularNuevo()
            {
                Marca = "Samsung",
                Modelo = "Galaxy s5",
                SistemaOperativo = "Android",
                RAM = 6,
                Almacenamiento = 256,
                Precio = 599,
                FechaIngreso = DateTime.Parse("01/03/2022")
            });
            celularesNuevos.Add(new CelularNuevo()
            {
                Marca = "Xiaomi",
                Modelo = "POCO F3",
                SistemaOperativo = "Android",
                RAM = 8,
                Almacenamiento = 256,
                Precio = 299,
                FechaIngreso = DateTime.Parse("01/09/2022")
            });
            celularesNuevos.Add(new CelularNuevo()
            {
                Marca = "Xiaomi",
```

```
            Modelo = "Redmi Note 10",
            SistemaOperativo = "Android",
            RAM = 8,
            Almacenamiento = 799,
            Precio = 499,
            FechaIngreso = DateTime.Parse("01/03/2022")
        });
        celularesNuevos.Add(new CelularNuevo()
        {
            Marca = "Apple",
            Modelo = "Iphone 10",
            SistemaOperativo = "IOS",
            RAM = 6,
            Almacenamiento = 128,
            Precio = 499,
            FechaIngreso = DateTime.Parse("01/05/2020")
        });
        celularesNuevos.Add(new CelularNuevo()
        {
            Marca = "Samsung",
            Modelo = "NOTE 10",
            SistemaOperativo = "Android",
            RAM = 8,
            Almacenamiento = 256,
            Precio = 399,
            FechaIngreso = DateTime.Parse("01/02/2020")
        });
        celularesNuevos.Add(new CelularNuevo()
        {
            Marca = "Apple",
            Modelo = "Iphone 15",
            SistemaOperativo = "Android",
            RAM = 12,
            Almacenamiento = 256,
            Precio = 999,
            FechaIngreso = DateTime.Parse("01/07/2023")
        });
        celularesNuevos.Add(new CelularNuevo()
        {
            Marca = "Xiaomi",
            Modelo = "Note 7",
            SistemaOperativo = "Android",
            RAM = 4,
            Almacenamiento = 128,
            Precio = 199,
            FechaIngreso = DateTime.Parse("09/03/2020")
        });
        celularesNuevos.Add(new CelularNuevo()
        {
            Marca = "Samsung",
            Modelo = "Galaxy s12",
            SistemaOperativo = "Android",
            RAM = 12,
            Almacenamiento = 512,
            Precio = 999,
            FechaIngreso = DateTime.Parse("01/03/2023")
        });

    }

    public void PromedioCelular()
    {
```

```
Console.WriteLine("*****************************************");

            double promedioPrecio = celularesNuevos.Average(c =>c.Precio);
            Console.WriteLine("PPOMEDIO PRECIOS DE CELULARES: \n" +
promedioPrecio);
        }
        public void CelularMarcaS()
        {

Console.WriteLine("*****************************************");

            var celularesSamsung = celularesNuevos.Where(c
=>c.Marca.Equals("Samsung")).ToList();
            Console.WriteLine("Celulares marca Samsung:");
            foreach (var celular in celularesSamsung)
            {
                Console.WriteLine("Modelo: " + celular.Modelo);
                Console.WriteLine("Precio: " + celular.Precio);
                Console.WriteLine();
            }
        }

        public void CelularRSA()
        {
            var resultado = from celular in celularesNuevos
                            where celular.RAM == 8 &&
                           celular.SistemaOperativo == "Android" &&
celular.Almacenamiento == 128
                            select celular;

Console.WriteLine("*****************************************");

            Console.WriteLine("RAM = 8GB, SO = Andreoid y Almacenanamiento
de 128: ");
            foreach (var celular in resultado)
            {
                Console.WriteLine("Modelo: " + celular.Modelo);
                Console.WriteLine("Precio: " + celular.Precio);
                Console.WriteLine();
            }
        }
        public void Celular_Ingreso()
        {
            var resultado = from celular in celularesNuevos
                            where celular.FechaIngreso.Year == 2005
                            select celular;

Console.WriteLine("*****************************************");

            Console.WriteLine("CELULARES QUE INGRESARON EL ANHO 2020:");
            foreach (var celular in resultado)
            {
                Console.WriteLine("Modelo: " + celular.Modelo);
                Console.WriteLine("Precio: " + celular.Precio);
                Console.WriteLine();
            }

        }
        public void ExpresionLambda()
        {
            var celularesApple = celularesNuevos.Where(c =>
           c.Marca.Equals("Apple"));
```

```
Console.WriteLine("*****************************************");

            Console.WriteLine("Modelo y Precio de los Celulares Apple ");
            foreach (var celular in celularesApple)
            {
                Console.WriteLine("Modelo: " + celular.Modelo);
                Console.WriteLine("Precio: " + celular.Precio);
                Console.WriteLine();
            }
        }
        public void ConsultaLinq()
        {
            var celularesApple = from celular in celularesNuevos
                                 where celular.Marca.Equals("Apple")
                                 select celular;

Console.WriteLine("*****************************************");

        }

    }
}


    internal class CelularNuevo : Celular
    {
        public DateTime FechaIngreso { get; set; }
        public double Precio { get; set; }

    }

    internal class Celular
    {
        public string Marca { get; set; }
        public string Modelo { get; set; }
        public string SistemaOperativo { get; set; }
        public int RAM { get; set; }
        public int Almacenamiento { get; set; }

    }
```

5. Realizar las Historias de Usuario y el Product Backlog para la empresa ChocoMax

La empresa ChocoMax está ubicada en la ciudad de Tarija, donde la empresa se dedica a la elaboración y venta de chocolates. Actualmente la empresa gestiona la venta de los chocolates de forma manuscrita, también se detectó que no cuenta con un buen control de los vendedores en que turno están o cuanto fue su venta en el día, lo cual genero perdidas económicas,

En la empresa ChocoMax existen dos turnos (turno mañana y turno tarde), cada vendedor trabaja solamente un turno. El Gerente general sólo le interesa la parte de reportes de las ventas por día, por mes y por vendedor, el vendedor requiere registrar las ventas, buscar o añadir los datos del cliente para emitir un recibo de la venta.

HU1: Informe de ventas diarias	
Como	Gerente
Quiero	tener acceso a informes de ventas diarias
Para	tomar decisiones basadas en datos

Tabla 1: Historia de usuario – Informe de ventas diarias

HU2: Informe de ventas mensuales	
Como	Gerente
Quiero	quiero obtener informes mensuales de ventas
Para	valuar el desempeño a lo largo del mes.

Tabla 2: Historia de usuario **–** Informe de ventas mensuales

HU3: Informe de ventas como vendedor	
Como	Gerente
Quiero	tener un resumen de ventas por vendedor
Para	identificar a los vendedores más exitosos.

Tabla 3: Historia de usuario – Informe de ventas por vendedor

HU4: Registrar venta de chocolates	
Como	Vendedor
Quiero	poder registrar mis ventas de chocolates de manera eficiente
Para	agilizar el proceso de ventas y reducir posibles errores.

Tabla 4: Historia de usuario – Registrar venta de chocolates

HU5: Emitir recibo de venta	
Como	Vendedor
Quiero	quiero buscar o añadir datos de clientes
Para	emitir recibos de ventas de manera eficiente y mejorar la satisfacción del cliente.

Tabla 5: Historia de usuario – Emitir recibo de venta

HU6: Generar recibo de venta	
Como	Vendedor
Quiero	quiero generar recibos de venta de manera automática
Para	entregar a los clientes, para que el proceso sea más eficiente y para mantener un registro de todas las transacciones.

Tabla 6: Historia de usuario – Generar recibo de venta

HU7: Iniciar sesión	
Como	Gerente
Quiero	Poder gestionar los perfiles de los vendedores para agregar o eliminar vendedores
Para	Administrar el acceso al sistema

Tabla 7: Historia de usuario – Iniciar sesión

PRODUCT BACKLOG

Historia de usuario	Descripción	Prioridad
HU1	**Informe de ventas diarias**	2
HU2	**Informe de ventas mensuales**	2
HU3	**Informe de ventas como vendedor**	2
HU4	**Registrar venta de chocolates**	1
HU5	**Emitir recibo de venta**	3
HU6	**Generar recibo de venta**	3
HU7	**Iniciar sesión**	1

Tabla 8: Product Backlog

6. Realizar las Historias de Usuario y el Product Backlog para mejorar el Sistema Dragón FNI

HU1: Paralelos disponibles	
Como	Estudiante
Quiero	Ver los paralelos disponibles de las materias y recibir una notificación cuando se habiliten
Para	Cambiar de paralelo

Tabla 1: Historia de usuario – Paralelos disponibles

HU2: Lista de docentes (laboratorio)	
Como	Estudiante
Quiero	Ver la lista de docentes (laboratorios)
Para	Para escoger un buen paralelo y horario

Tabla 2: Historia de usuario – Lista de docentes

HU3: Registrar notas	
Como	Docente
Quiero	Tener acceso a la planilla de calificaciones
Para	Cambiar las notas

Tabla 3: Historia de usuario – Registrar notas

HU4: Interfaz responsive	
Como	Estudiante
Quiero	que la interfaz del sistema sea responsive
Para	utilizar el sistema de manera efectiva desde cualquier dispositivo

Tabla 4: Historia de usuario – Interfaz responsive

HU5: Lista de auxiliares	
Como	Estudiante
Quiero	Poder ver la lista de auxiliares y contacto
Para	De manera sencilla para unirme a mis respectivos grupos

Tabla 5: Historia de usuario – Lista de auxiliares

PRODUCT BACKLOG

Historia de usuario	Descripción	Prioridad
HU1	**Paralelos disponibles**	1
HU2	**Lista de docentes (laboratorio)**	2
HU3	**Registrar notas**	1
HU4	**Interfaz responsive**	3
HU5	**Lista de auxiliares**	2

Tabla 6: Product Backlog