UNIVERSIDAD TÉCNICA DE ORURO
FACULTAD NACIONAL DE INGENIERIA
INGENIERIA DE SISTEMAS E INGENIERIA INFORMATICA

PRACTICA N° 1

Materia: Actualización Tecnológica SIS2420 “A”
Docente: Ing. Saul Mamani M.
Auxiliar: Egr. William Mucio Achabal Villalpando
Estudiante: Mamani Solares Gustavo

Parte Teórica 30 pts.
EXPLICAR EN SUS PALABRAS CADA PREGUNTA

1. ¿Qué es un sistema?

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

   En resumen:

   - Una clase abstracta se utiliza como una plantilla para clases derivadas, puede contener métodos abstractos y concretos, y se instancia a través de herencia.
   - Una clase estática no se puede instanciar en absoluto y contiene solo miembros estáticos; se utiliza para agrupar funciones relacionadas que no dependen del estado de un objeto y se accede a sus miembros a través del nombre de la clase.

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

   la herencia es un mecanismo que permite la creación de jerarquías de clases, mientras que el polimorfismo permite que objetos de diferentes clases respondan a un mismo mensaje de manera específica, lo que aumenta la flexibilidad y extensibilidad del código en C#.

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

   El Ciclo de Vida del Desarrollo de Software (SDLC) es un proceso estructurado que guía la planificación, diseño, desarrollo, pruebas, implementación y mantenimiento de aplicaciones de software. Se divide en fases como Requisitos, Diseño, Implementación, Pruebas, Despliegue y Mantenimiento. Existen diferentes enfoques (cascada, espiral, ágil) para llevar a cabo el SDLC, y su elección depende de las necesidades del proyecto. El SDLC es esencial para desarrollar software de manera ordenada, controlada y que cumpla con los requisitos del cliente y las normas de calidad.

5. Para qué sirven estos comandos de Git:
   - Git init
     Se utiliza para inicializar un nuevo repositorio git en un directorio local.

- Git status.- se utiliza para verificar el estado actual de un repositorio Git. Al ejecutar nos mostrara información sobre los archivos se a modificado etc.

- Git add.- nos permite añadir totos los cambios.

- Git commit -m”mensaje”.- nos sirve para guardar los cambios que se hizo en el proyecto.

- Git log.- este comando nos muestra todos los commit del repositorio.
- Git checkout.- permite recuperar archivos ,commit y ramas te permite moverte entre ramas o commit y ver las versiones del código.
- Git checkout.—b NombreRama.- se utiliza para crear una nueva rama en el repositorio y cambiar a esa rama en un solo paso.
- Git Branch.- este comando nos muestra las ramas que tiene nuestro repositorio.
- Git push.-nos permite subir nuestro proyecto a un repositorio remoto(Github)
- Git pull.- este comando extrae y descarga contenido desde un repositorio remoto y actuliza al instante el repositorio local para reflejar ese contenido.
- Git merge.- permite tomar las líneas independientes de desarrollo creadas por git Branch e integrarlas en una sola rama.
- Git clone.- apunta aun repositorio existente y clona o copia dicho repositorio en un nuevo directorio,en otra ubicación.

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

   las metodologías tradicionales siguen un enfoque lineal y planificado, mientras que las metodologías ágiles son flexibles, iterativas e incrementales. Las metodologías ágiles son ideales para proyectos en los que los requisitos pueden cambiar o no están completamente definidos desde el principio, y donde la entrega temprana de valor es crucial. Las metodologías tradicionales son adecuadas para proyectos donde los requisitos son estables y bien definidos desde el principio, y donde se prioriza una planificación y documentación exhaustiva. La elección entre ambos enfoques depende de las necesidades específicas del proyecto y las preferencias del equipo.

7. Dar 5 ejemplos de una metodología tradicional y 5 ejemplos de una metodología tradicional ágil
   Metodologías tradicionales:
   - Modelo en cascada
   - Modelo de desarrollo en espiral
   - Modelo de desarrollo en v
   - Modelo big bang
   - Modelo de desarrollo en fases

   Metodologias agiles:

- Scrum
- Kanban
- Extreme programming
- Lean
- Feature driven development

8. ¿Qué es un Requerimiento Funcional y No Funcional?
   Un requerimiento funcional es un requerimiento que el cliente teda mientras que un requerimiento no funcional es aquel que uno mismo como programador tiene que ver en el programa como un bug , contraseñas y demás.

9. ¿Qué es SCRUM?
   Scrum es un marco de trabajo ágil ampliamente utilizado en el desarrollo de software y en la gestión de proyectos se basa en ciclos de desarrollo llamado sprints.se basa en la adaptación del proyecto a cambios .

10. ¿Cuáles son los roles de SCRUM?
    Producto owner.-
    Representa a los interesados y define los ruquisitos del producto.es responsable de priorizar el tranbajo del equipo.
    Scrum master.-
    Facilita el proceso scrum elimina obstáculos y asegura que el equipo siga las practicas agiles
    Equipo de desarrollo(developers).- son los programadores que realiza el trabajo real de desarrollo.

Parte Práctica 70 pts.

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

   Entrada

   Numero: 55

   Salida

   Cincuenta y cinco

   Solución.-

```
using System;
using System.Collections.Generic;

public static class NumeroALetra
{
   private static Dictionary<int, string> mapeoNumerosALetras = new
Dictionary<int, string>
   {
      {0, "Cero"}, {1, "Uno"}, {2, "Dos"}, {3, "Tres"}, {4, "Cuatro"},
      {5, "Cinco"}, {6, "Seis"}, {7, "Siete"}, {8, "Ocho"}, {9, "Nueve"}
   };

   public static string ConvertirNumeroALetra(int numero)
   {
      if (mapeoNumerosALetras.ContainsKey(numero))
      {
         return mapeoNumerosALetras[numero];
      }
      else
      {
         return "Número no válido";
```

```
        }
    }
}

class Program
{
    static void Main()
    {
        Console.Write("Número: ");
        if (int.TryParse(Console.ReadLine(), out int numero))
        {
            string resultado = NumeroALetra.ConvertirNumeroALetra(numero);
            Console.WriteLine(resultado);
        }
        else
        {
            Console.WriteLine("Entrada inválida. Ingrese un número entero.");
        }
    }
}
```

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

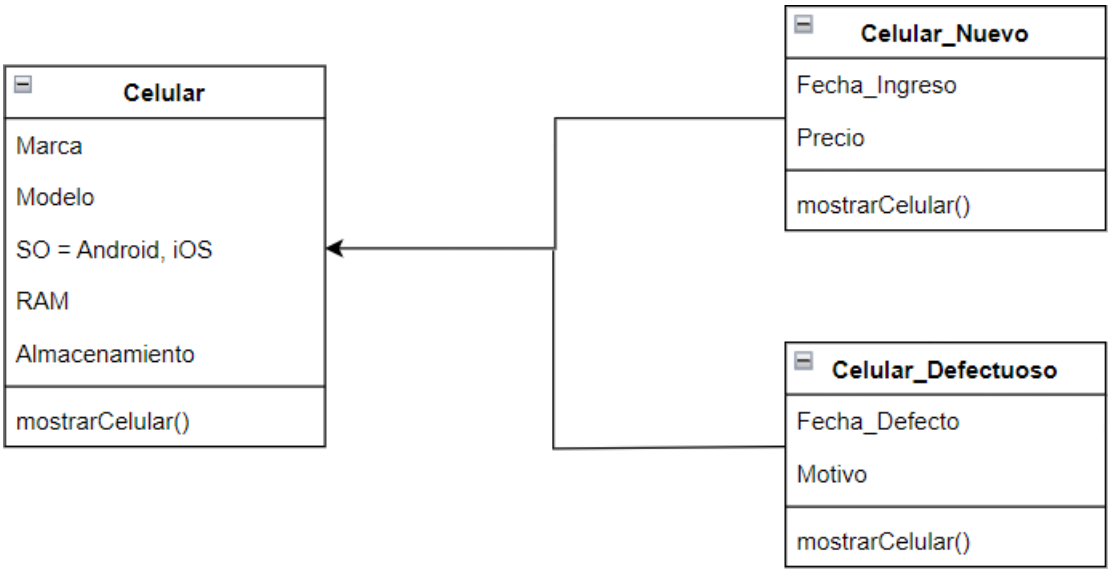
   Salida
   - Lista 1: 2, 4, 6,10 (Múltiplos de 2)
   - Lista 2: 2, 3, 5 (Primos)
   - Lista 3: 5, 10 (Múltiplos de 5)
   - Lista 4: 6 (Perfectos)

Solución:

```
using System;
using System.Collections.Generic;

class Program
{
    static bool EsPrimo(int numero)
    {
        if (numero <= 1)
            return false;

        for (int i = 2; i * i <= numero; i++)
        {
            if (numero % i == 0)
                return false;
        }
        return true;
    }

    static bool EsMultiploDe(int numero, int divisor)
    {
        return numero % divisor == 0;
    }
```

```
    static void Main()
    {
        Console.Write("¿Cuántos números deseas añadir? ");
        if (int.TryParse(Console.ReadLine(), out int n))
        {
            List<int> numeros = new List<int>();

            for (int i = 0; i < n; i++)
            {
                Console.Write($"Añade el número {i + 1}: ");
                if (int.TryParse(Console.ReadLine(), out int numero))
                {
                    numeros.Add(numero);
                }
                else
                {
                    Console.WriteLine("Entrada no válida. Por favor, ingresa un número entero.");
                    i--;
                }
            }

            List<int> multiplosDe2 = new List<int>();
            List<int> primos = new List<int>();
            List<int> multiplosDe5 = new List<int>();
            List<int> perfectos = new List<int>();

            foreach (int numero in numeros)
            {
                if (EsMultiploDe(numero, 2))
                    multiplosDe2.Add(numero);

                if (EsPrimo(numero))
                    primos.Add(numero);

                if (EsMultiploDe(numero, 5))
                    multiplosDe5.Add(numero);

                int sumaDivisores = 0;
                for (int i = 1; i < numero; i++)
                {
                    if (EsMultiploDe(numero, i))
                        sumaDivisores += i;
                }
                if (sumaDivisores == numero)
                    perfectos.Add(numero);
            }

            Console.WriteLine("Lista 1: " + string.Join(", ", multiplosDe2) + " (Múltiplos de 2)");
            Console.WriteLine("Lista 2: " + string.Join(", ", primos) + " (Primos)");
            Console.WriteLine("Lista 3: " + string.Join(", ", multiplosDe5) + " (Múltiplos de 5)");
            Console.WriteLine("Lista 4: " + string.Join(", ", perfectos) + " (Perfectos)");
        }
        else
        {
            Console.WriteLine("Entrada no válida. Por favor, ingresa un número entero.");
        }
    }
}
```

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




```
using System;

abstract class Celular
{
    public string Marca { get; set; }
    public string Modelo { get; set; }
    public int Ram { get; set; }
    public int Almacenamiento { get; set; }
    public string SistemaOperativo { get; set; }

    public Celular(string marca, string modelo, int ram, int almacenamiento, string so)
    {
        Marca = marca;
        Modelo = modelo;
        Ram = ram;
        Almacenamiento = almacenamiento;
        SistemaOperativo = so;
    }

    public abstract void MostrarDetalles();
}

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

    public CelularNuevo(string marca, string modelo, int ram, int almacenamiento, string so, DateTime fechaIngreso, decimal
precio)
        : base(marca, modelo, ram, almacenamiento, so)
    {
        FechaIngreso = fechaIngreso;
        Precio = precio;
    }

    public override void MostrarDetalles()
    {
        Console.WriteLine("Celular Nuevo:");
        Console.WriteLine($"Marca: {Marca}");
        Console.WriteLine($"Modelo: {Modelo}");
        Console.WriteLine($"RAM: {Ram} GB");
        Console.WriteLine($"Almacenamiento: {Almacenamiento} GB");
        Console.WriteLine($"Sistema Operativo: {SistemaOperativo}");
        Console.WriteLine($"Fecha de Ingreso: {FechaIngreso}");
        Console.WriteLine($"Precio: ${Precio}");
    }
}

class CelularDefectuoso : Celular
{
    public DateTime FechaDefecto { get; set; }
    public string Motivo { get; set; }

    public CelularDefectuoso(string marca, string modelo, int ram, int almacenamiento, string so, DateTime fechaDefecto,
string motivo)
```

```
        : base(marca, modelo, ram, almacenamiento, so)
    {
        FechaDefecto = fechaDefecto;
        Motivo = motivo;
    }

    public override void MostrarDetalles()
    {
        Console.WriteLine("Celular Defectuoso:");
        Console.WriteLine($"Marca: {Marca}");
        Console.WriteLine($"Modelo: {Modelo}");
        Console.WriteLine($"RAM: {Ram} GB");
        Console.WriteLine($"Almacenamiento: {Almacenamiento} GB");
        Console.WriteLine($"Sistema Operativo: {SistemaOperativo}");
        Console.WriteLine($"Fecha de Defecto: {FechaDefecto}");
        Console.WriteLine($"Motivo del Defecto: {Motivo}");
    }
}

class Program
{
    static void Main()
    {
        Celular celularNuevo = new CelularNuevo("Samsung", "Galaxy S21", 8, 128, "Android", DateTime.Now, 799.99m);
        Celular celularDefectuoso = new CelularDefectuoso("Apple", "iPhone 12", 6, 64, "iOS", DateTime.Now, "Pantalla
rota");

        MostrarDetalleCelular(celularNuevo);
        MostrarDetalleCelular(celularDefectuoso);
    }

    static void MostrarDetalleCelular(Celular celular)
    {
        celular.MostrarDetalles();
        Console.WriteLine();
    }
}
```

4. **Del ejercicio 3 crear una lista utilizando la clase Celular_Nuevo**
   - Añadir 10 celulares_nuevos.
   - Crear una función Prom_Celular, utilizando expresiones lambda sacar el promedio del precio de los celulares.
   - Crear una función Cel_MarcaS para buscar los celulares de Marca = Samsung, utilizando expresiones lambda.
   - Crear una función Celular_RSA, utilizando consultas LinQ mostrar los celulares que son de RAM = 8GB, SO = Android y Almacenamiento de 128 GB.
   - Crear una función Celular_Ingreso, utilizando consultas LinQ mostrar los celulares que ingresaron el año 2005.
   - Crear dos funciones, la primera función usando: Expresiones lambda y la segunda función: consultas LinQ, donde se debe mostrar el modelo y el precio de los celulares Apple.
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.

1. HISTORIAS DE USUARIO

Para la determinación de requerimientos del sistema se recolectarán las Historias de Usuario.

HU1: iniciar sesion	
Como	Gerente general
Quiero	Una validacion de usuario
Para	Ingresar al sistema wed

HU2: registro de venta	
Como	Vendedor
Quiero	Registrar una venta ,incluyendo el producto vendido
Para	Llevar un registro de mis transacciones

Tabla 2: Historia de usuario – Iniciar Sesión

HU3: Recibo de Venta	
Como	Vendedor
Quiero	Añadir datos del cliente como nombre dirección y nuemero de teléfono para generar recibo de venta
Para	Tener un control de los relojes disponibles

Tabla 3: Historia de usuario – Registrar Venta

HU4: Gestionar (CRUD) Usuarios	
Como	Vendedor
Quiero	Crear, Leer, Editar y Eliminar (CRUD) usuario
Para	Controlar las acciones que realizan

Tabla 4: Historia de usuario – Gestionar (CRUD) Usuarios

HU5: Generar Reportes	
Como	Gerente General
Quiero	Generar reportes de las ventas mensuales y de las ventas por dia,
Para	Tomar decisiones sobre el estado actual de la empresa

Tabla 5: Historia de usuario – Generar Reportes

HU6: Gestionar (CRUD) Clientes	
Como	Gerente general
Quiero	Crear, Leer, Editar y Eliminar (CRUD) clientes
Para	Acceder a sus datos para cada venta

Tabla 6: Historia de usuario – Gestionar (CRUD) Clientes

HU7: generar reportes mensuales	
Como	Gerente General
Quiero	Obtener un informe mensual sobre las ventas
Para	Comprender las tendencias alo largo del mes

Tabla 7: Historia de usuario – Gestionar (CRUD) Relojes

HU8: informe de ventas por vendedor	
Como	Gerente general
Quiero	Informe de ventas por vendedor
Para	evaluar el desempeño individual

2. PRODUCT BACKLOG

La pila del producto de pendientes a desarrollar está constituida por las historias de usuario (Requerimientos funcionales), y ordenada según prioridad de implementación.

Historia de usuario	Descripción	Prioridad
HU1	**Iniciar Sesión**	1
HU2	**Registrar Venta**	2
HU3	**Recibo de venta**	1
HU4	**Gestionar (CRUD)usuario**	4
HU5	**Generar reportes**	2
HU6	**Gestionar (CRUD) cliente**	1
HU7	**Generar reportes mensuales**	4
HU8	**Informe de ventas por vndedor**	3

Nota: Subir toda la Práctica en un repositorio público de su cuenta GitHub y enviar el enlace del repositorio al inbox por Slack, si entregas la práctica antes del parcial obtendrás +10 puntos a la nota obtenida. ¡Prácticas iguales serán anuladas!

Vo. Bo. Ing. Saul Mamani M.
Docente

Egr. William Mucio Achabal Villalpando
Auxiliar