PRACTICA 1

SIS 2420 “A”

Apellidos y Nombres: **Escobar Siles Dylan**

C.I:12549550

EXPLICAR EN SUS PALABRAS CADA PREGUNTA

1. **¿Qué es un sistema?**
   Un sistema es un conjunto de pasos que vienen siendo, Input (Entrada), las operaciones que se harán con el Input y el resultado ósea Output (Salida).
2. ¿Qué es y qué diferencias tienen una clase abstracta y una clase estática en C#?
   Una clase abstracta se utiliza como plantilla para clases derivadas utilizando la herencia, estas deben implementar métodos o propiedades abstractas para sus clases hijas mientras que una clase estática no permite la herencia y no se pueden instanciar
3. ¿Qué es y qué diferencias tienen la herencia y polimorfismo en C#?
   La diferencia radica que cuando una clase hereda a otra clase esta clase pasa ser un tipo de clase padre teniendo todos los métodos y atributos heredados, mientras que el polimorfismo se trata de como puede actuar las clases hijas en base a la clase padre
4. ¿Qué es un ciclo de vida del desarrollo de software (SDLC)?
   Es un conjunto de procesos y fases sistemáticamente planificados que guian al desarrollo del software desde su concepción hasta su implementación, el objetivo es proporcionar software de alta calidad, las fases son:
   Requerimientos.- Lo que el software necesitara a la hora de construir el sistema
   Analisis.- Después de la recopilación de requisitos, se realiza un análisis más detallado de estos para comprender completamente los procesos comerciales, las restricciones y las expectativas del cliente. Esta fase ayuda a refinar y definir aún más los requisitos.
   Diseño.- En aquí entra el como haremos el software que el cliente solicita
   Implementacion.- Empezamos a escribir código según las especificaciones y lo acordado con el equipo
   Pruebas.- Probamos el sistema si es que tiene errores, fallos lógicos y pruebas de seguridad
5. Para qué sirven estos comandos de Git:
   Git init: Es para inicializar un repositorio vacio
   Git status: Nos muestra el estado actual de nuestro repositorio
   Git add .: Este comando agrega todos los archivos modificados y nuevos en el directorio de trabajo al área de preparación (staging area) para ser incluidos en el próximo commit.
   Git commit –m “Mensaje: Permite crear un nuevo commit con los cambios que se encuentran en el área de preparación. El mensaje entre comillas describe los cambios realizados en el commit.

   Git log: Muestra un registro de todos los commits en el repositorio, incluyendo información como el autor, la fecha y el mensaje del commit.

   Git checkout: Se utiliza principalmente para cambiar entre ramas en Git. También se puede utilizar para deshacer cambios en archivos específicos en el directorio de trabajo.

Git checkout -b NombreRama: Crea una nueva rama con el nombre especificado y cambia a esa rama en un solo paso. Útil cuando deseas crear y trabajar en una nueva rama.

Git branch: Muestra una lista de todas las ramas en el repositorio. La rama actual se resalta con un asterisco (*).

Git push: Sube los cambios locales al repositorio remoto en la rama especificada. Esto es útil para compartir tu trabajo con otros y mantener el repositorio remoto actualizado.

Git pull: Descarga los cambios desde el repositorio remoto y los fusiona con la rama actual en tu repositorio local. Se utiliza para mantener tu repositorio local actualizado con los cambios realizados por otros colaboradores.

Git merge: Combina los cambios de una rama en otra. Se utiliza para fusionar una rama de desarrollo con la rama principal.

Git clone: Crea una copia exacta de un repositorio remoto en tu máquina local. Se utiliza para obtener una copia de un proyecto existente y comenzar a trabajar en él.

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

   Las metodologías tradicionales son secuenciales y orientadas a la planificación detallada, y se espera que el proyecto no cambie hasta su finalización, mientras que las metodologías agiles son agiles y iterativas, se realiza una reunión con el cliente para hacer un seguimiento, para poder realizar cambios.

7. Dar 5 ejemplos de una metodología tradicional y 5 ejemplos de una metodología tradicional ágil

   Metodología tradicional:

   Metodología de Cascada.- En este enfoque la fase se debe completar para poder pasar a la siguiente

   Modelo en V.- Es similar a la metodología de la cascada solo que este tiene un enfoque mas hacia las pruebas donde cada fase tiene su fase correspondiente de pruebas

   Modelo de desarrollo en Espiral.- Este modelo combina elementos de desarrollo iterativo con enfoque en la gestión de riesgos. Cada ciclo de desarrollo es una "espiral" que incluye planificación, riesgos, desarrollo y pruebas, donde cada vez el ciclo es mas largo como una espiral

   Modelo de Desarrollo de Prototipos: Se crea un prototipo inicial del software para ayudar a los stakeholders a comprender los requisitos. Luego, se desarrolla el producto final basado en el prototipo.

   Modelo en Herramienta de Desarrollo Rápido (Rapid Application Development - RAD): Se enfoca en la construcción rápida de prototipos funcionales y la iteración continua. Se prioriza la velocidad de desarrollo.

   Metodologías Agiles:

   Scrum: Scrum es un marco de trabajo ágil que utiliza iteraciones de tiempo fijo (llamadas "sprints") para entregar incrementos de software. Se basa en roles definidos (Scrum Master, Product Owner, Equipo de Desarrollo) y prácticas como reuniones diarias de seguimiento (daily standup) y planificación de sprint.

Kanban: Kanban es un enfoque visual de gestión de tareas que se centra en la visibilidad del trabajo y la limitación del trabajo en curso. Las tareas se mueven a través de columnas en un tablero Kanban a medida que avanzan.

Extreme Programming (XP): XP es una metodología ágil que se centra en la programación de parejas, pruebas unitarias, integración continua y otras prácticas para garantizar la calidad del software.

Lean Software Development: Inspirado en los principios del Lean Manufacturing, se enfoca en la eliminación de desperdicio, la entrega de valor al cliente y la mejora continua.

Dynamic Systems Development Method (DSDM): DSDM es un marco de trabajo ágil que se centra en la entrega temprana de productos y la colaboración estrecha con los usuarios finales. Utiliza fases y ciclos iterativos para lograr estos objetivos.

8. ¿Qué es un Requerimiento Funcional y No Funcional?
   Un requerimiento funcional es alguna característica que el cliente quiere que el equipo añada al sistema, mientras que la no funcional es una que ya debería estar en el sistema como por ejemplo, seguridad, buen rendimiento, flexibilidad,etc.
9. ¿Qué es SCRUM?
   Es una metodología ágil, se basa en una serie de principios y roles que se centra en la entrega de valor de forma iterativa y colaborativa.
   Donde hay sprints que vienen siendo periodos cortos de tiempo donde se realiza el desarrollo del proyecto, roles, reuniones y además cuenta con un tablero
10. ¿Cuáles son los roles de SCRUM?
    Scrum define roles claros, que incluyen el Scrum Master, el Product Owner y el Equipo de Desarrollo. Cada uno de estos roles tiene responsabilidades específicas en el proceso.
    El Scrum Master es responsable de facilitar el proceso Scrum, eliminar obstáculos y asegurarse de que el equipo siga las prácticas de Scrum.
    El Product Owner es responsable de definir los requisitos y prioridades del proyecto, y toma decisiones sobre el contenido del producto.
    El Equipo de Desarrollo es el grupo de personas que realiza el trabajo real en el proyecto.

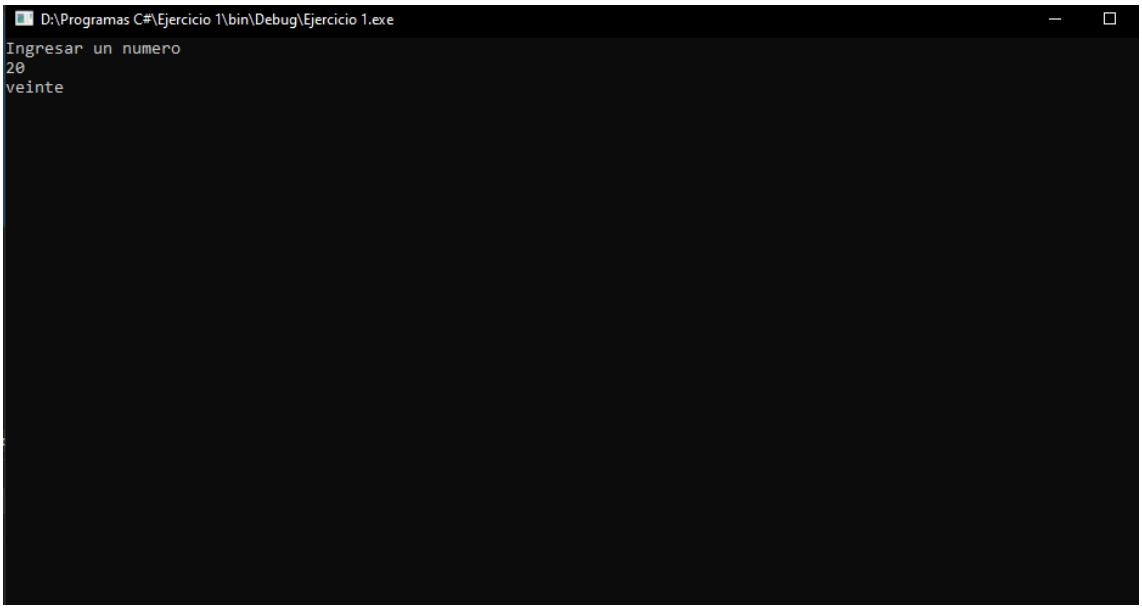
Parte Practica

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

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

namespace Ejercicio_1
{
    static class Program
    {
       static string ConvertirLiteral(int dato)
        {
            string[] unidades = {"uno", "dos", "tres", "cuatro", "cinco",
"seis", "siete", "ocho", "nueve" };
            string[] decenas = {"diez","veinte", "treinta", "cuarenta",
"cincuenta", "sesenta", "setenta", "ochenta", "noventa" };
```

```
    int resto;
    string uni="";
    string dec="";
    if(dato==10)
    {
        return "diez";
    }
    if(dato==11)
    {
        return "once";
    }
    if(dato==12)
    {
        return "doce";
    }
    if(dato==13)
    {
        return "trece";
    }
    if(dato==14)
    {
        return "catorce";
    }
    if(dato==15)
    {
        return "quince";
    }
for(int i=0;i<3;i++)
{
        resto = dato % 10;
        dato = dato / 10;
    if(i==0 && resto!=0)
        {
           for(int j=0;j<resto;j++)
            {
                uni=unidades[j];
            }
        }
    if(i==1 && resto!=0)
        {
            for(int j=0;j<resto;j++)
            {
                if (uni=="")
                {
                    dec = decenas[j];
                }
                else
                {
                    dec = decenas[j] + " y ";
                }
            }
        }


}
    return dec + uni;
}
static void Main(string[] args)
{
    int dato;
    string convertido;
    Console.WriteLine("Ingresar un numero");
    dato = int.Parse(Console.ReadLine());
```

```
            convertido=ConvertirLiteral(dato);
            Console.WriteLine(convertido);

            Console.ReadKey();
        }
    }
}
```

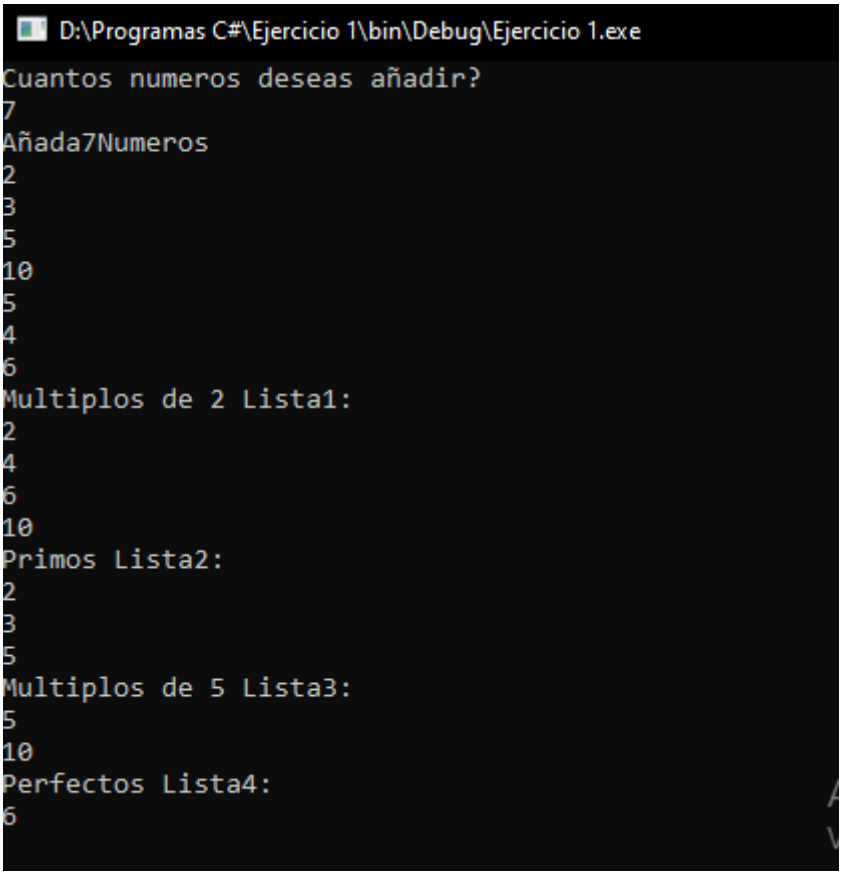


2. Realizar un programa utilizando listas que te permita ingresar n números por teclado donde cada número entre en las siguientes listas:

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

namespace Ejercicio_1
{
    static class Program
    {
        static void multiplosde2(int[] vec,int n)
        {
            SortedSet<int> vec1 = new SortedSet<int>();
            for (int i=0;i<n;i++)
            {
                if (vec[i]%2==0)
                {
                    vec1.Add(vec[i]);
                }
            }

            Console.WriteLine("Multiplos de 2 Lista1: ");
            foreach(int i in vec1)
            {
                Console.WriteLine(i);
            }
        }
        static void primos(int[] vec,int n)
```

```
{
    SortedSet<int> vec1 = new SortedSet<int>();
    int cont = 2;
    for(int i=0;i<n;i++)
    {
        for(int j=2;j<=9;j++)
        {
            if (vec[i]%j!=0)
            {
                cont++;
            }
        }
        if(cont==9)
        {
            vec1.Add(vec[i]);
        }
        cont = 2;
    }

    Console.WriteLine("Primos Lista2: ");
    foreach (int i in vec1)
    {
        Console.WriteLine(i);
    }

}
static void multiplosde5(int[] vec,int n)
{
    SortedSet<int> vec1 = new SortedSet<int>();
    for (int i = 0; i < n; i++)
    {
        if (vec[i] % 5 == 0)
        {
            vec1.Add(vec[i]);
        }
    }

    Console.WriteLine("Multiplos de 5 Lista3: ");
    foreach (int i in vec1)
    {
        Console.WriteLine(i);
    }
}
static void perfectos(int[] vec,int n)
{
    SortedSet<int> vec1 = new SortedSet<int>();
    int sum = 0;
    for(int i=0;i<n;i++)
    {
        for(int j = 1; j < vec[i];j++)
        {
            if (vec[i]%j==0)
            {
                sum += j;
            }
        }
        if (vec[i]==sum)
        {
            vec1.Add(vec[i]);
        }
        sum = 0;
    }
```

```
            Console.WriteLine("Perfectos Lista4: ");
            foreach (int i in vec1)
            {
                Console.WriteLine(i);
            }
        }
        static void Main(string[] args)
        {
            int n;
            Console.WriteLine("Cuantos numeros deseas añadir?");
            n = int.Parse(Console.ReadLine());
            int[] vec = new int[n];
            Console.WriteLine("Añada" + n + "Numeros");
            for(int i=0;i<n;i++)
            {
                vec[i] = int.Parse(Console.ReadLine());
            }
            multiplosde2(vec,n);
            primos(vec,n);
            multiplosde5(vec,n);
            perfectos(vec,n);



            Console.ReadKey();
        }
}
    }
```



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

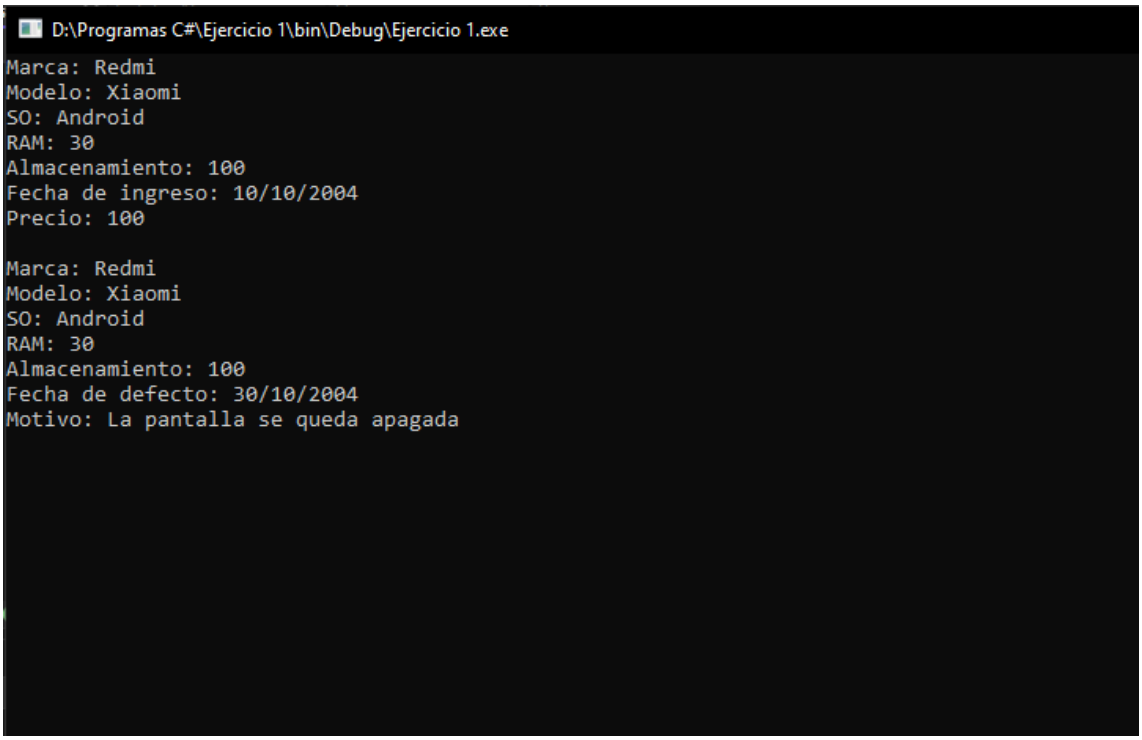
Main

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

namespace Ejercicio_1
{
    static class Program
    {

        static void Main(string[] args)
        {
            Celular obj1 = new Celular_Nuevo("10/10/2004",100);
            Celular obj2 = new Celular_Defectuoso("30/10/2004","La pantalla
se queda apagada");

            obj1.mostrarCelular();
            Console.WriteLine();
            obj2.mostrarCelular();


            Console.ReadKey();
        }
    }
      }
      Clase padre

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

namespace Ejercicio_1
{
    abstract class Celular
    {
        public string Marca="Redmi";
        public string Modelo="Xiaomi";
        public string SO="Android";
        public int RAM=30,Almacenamiento=100;

        public abstract void mostrarCelular();

    }
}
```

Clase hija

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

namespace Ejercicio_1
{
    class Celular_Nuevo : Celular
    {
        private string Fecha_Ingreso;
        private int Precio;
```

```
        public Celular_Nuevo(string Fecha,int Precio)
        {
            Fecha_Ingreso = Fecha;
            this.Precio = Precio;
        }

        public override void mostrarCelular()
        {
            Console.WriteLine("Marca: " + Marca);
            Console.WriteLine("Modelo: " + Modelo);
            Console.WriteLine("SO: " + SO);
            Console.WriteLine("RAM: " + RAM);
            Console.WriteLine("Almacenamiento: " + Almacenamiento);
            Console.WriteLine("Fecha de ingreso: "+Fecha_Ingreso);
            Console.WriteLine("Precio: " + Precio);
        }
    }
      }
      Clase hija

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

namespace Ejercicio_1
{
    internal class Celular_Defectuoso:Celular
    {
        private string Fecha_Defecto;
        private string motivo;

        public Celular_Defectuoso(string Fecha,string motivo)
        {
            Fecha_Defecto = Fecha;
            this.motivo = motivo;
        }
        public override void mostrarCelular()
        {
            Console.WriteLine("Marca: " + Marca);
            Console.WriteLine("Modelo: " + Modelo);
            Console.WriteLine("SO: " + SO);
            Console.WriteLine("RAM: " + RAM);
            Console.WriteLine("Almacenamiento: " + Almacenamiento);
            Console.WriteLine("Fecha de defecto: " + Fecha_Defecto);
            Console.WriteLine("Motivo: " + motivo);
        }
    }
}
```



4. **Del ejercicio 3 crear una lista utilizando la clase Celular_Nuevo**

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

namespace Ejercicio_1
{
    static class Program
    {

        static void Main(string[] args)
        {
            double promPrecio;
            Celular_Nuevo[] obj1= new Celular_Nuevo[10];
            Console.WriteLine("Celulares Nuevos");
           for(int i=0;i<10;i++)
            {
                Console.WriteLine("\n");
                obj1[i] = new Celular_Nuevo();
                Console.Write("Marca: ");
                obj1[i].Marca=(Console.ReadLine());
                Console.Write("Modelo: ");
                obj1[i].Modelo = (Console.ReadLine());
                Console.Write("SO: ");
                obj1[i].SO = (Console.ReadLine());
                Console.Write("RAM: ");
                obj1[i].RAM = int.Parse(Console.ReadLine());
                Console.Write("Almacenamiento: ");
                obj1[i].Almacenamiento = int.Parse(Console.ReadLine());
                Console.Write("Fecha_Ingreso: ");
                obj1[i].Fecha_Ingreso = (Console.ReadLine());
                Console.Write("Precio: ");
                obj1[i].Precio = int.Parse(Console.ReadLine());
```

```
            }

            double promedioPrecio = Prom_Celular(obj1, 10, celular =>
celular.Precio);


            double Prom_Celular(Celular_Nuevo[] celulares, int cantidad,
Func<Celular_Nuevo, int> selector)
            {
                double total = 0;
                for (int i = 0; i < cantidad; i++)
                {
                    total += selector(celulares[i]);
                }
                return total / cantidad;
            }
            Console.WriteLine("\nCelulares Samnsung");
            var marca = obj1.Where(celular => celular.Marca == "Samnsung");
            foreach (var celular in marca)
            {
                celular.mostrarCelular();
            }
            var resultados = obj1.Where(celular => celular.RAM == 8 &&
celular.SO == "Android" && celular.Almacenamiento == 128);

            Console.WriteLine("\nCelulares con RAM de 8GB, SO Android y
Almacenamiento de 128GB:");
            foreach (var celular in resultados)
            {
                celular.mostrarCelular();
            }
            Console.WriteLine("\nEL PROMEDIO ES: " + promedioPrecio);
            MostrarCelularesAppleExpresionLambda(obj1);
            MostrarCelularesAppleConsultaLinq(obj1);



         void MostrarCelularesAppleExpresionLambda(Celular_Nuevo[]
celulares)
        {
            Console.WriteLine("Celulares Apple (Expresiones Lambda):");
            var celularesApple = celulares.Where(c => c.Marca == "Apple");
            foreach (var celular in celularesApple)
            {
                Console.WriteLine($"Modelo: {celular.Modelo}, Precio:
{celular.Precio}");
            }
        }

         void MostrarCelularesAppleConsultaLinq(Celular_Nuevo[] celulares)
        {
            Console.WriteLine("Celulares Apple (Consultas LINQ):");
            var celularesApple = from c in celulares
                                 where c.Marca == "Apple"
                                 select c;
            foreach (var celular in celularesApple)
            {
                Console.WriteLine($"Modelo: {celular.Modelo}, Precio:
{celular.Precio}");
            }
        }

        Console.ReadKey();
```

```
        }
    }
}


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

namespace Ejercicio_1
{
    abstract class Celular
    {
        public string Marca;
        public string Modelo;
        public string SO;
        public int RAM,Almacenamiento;

        public abstract void mostrarCelular();

    }
}

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

namespace Ejercicio_1
{
    class Celular_Nuevo : Celular
    {
        public string Fecha_Ingreso { get; set; }
        public int Precio { get; set; }


        public override void mostrarCelular()
        {
            Console.WriteLine("\n LISTA DE CELULARES");
            Console.WriteLine("Marca: " + Marca);
            Console.WriteLine("Modelo: " + Modelo);
            Console.WriteLine("SO: " + SO);
            Console.WriteLine("RAM: " + RAM);
            Console.WriteLine("Almacenamiento: " + Almacenamiento);
            Console.WriteLine("Fecha de ingreso: "+Fecha_Ingreso);
            Console.WriteLine("Precio: " + Precio);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ejercicio_1
{
    internal class Celular_Defectuoso:Celular
    {
        private string Fecha_Defecto;
```

```
        private string motivo;

        public Celular_Defectuoso(string Fecha,string motivo)
        {
            Fecha_Defecto = Fecha;
            this.motivo = motivo;
        }
        public override void mostrarCelular()
        {
            Console.WriteLine("Marca: " + Marca);
            Console.WriteLine("Modelo: " + Modelo);
            Console.WriteLine("SO: " + SO);
            Console.WriteLine("RAM: " + RAM);
            Console.WriteLine("Almacenamiento: " + Almacenamiento);
            Console.WriteLine("Fecha de defecto: " + Fecha_Defecto);
            Console.WriteLine("Motivo: " + motivo);
        }
    }
}
```