El empleo es de todos

Mintrabajo

APLICAR PROGRAMACIÓN ORIENTADA A OBJETO CON C#

José Fernando Galindo Suárez

SENA

PROGRAMACIÓN ORIENTADA A OBJETO CON C#

4

INTERFACES PROGRAMACIÓN ORIENTADA A OBJETO CON C#

JOSE FERNANDO GALINDO SUAREZ

OBJETIVOS

Después de completar esta lección usted estará en la capacidad de:

- **Entender los conceptos básicos de la utilización de las interfaces en la programación orientada a objetos con C#.**
- **Crear objetos heredados desde una interface, modificando o restringiendo el acceso, el estado y comportamiento.**

INTERFACES

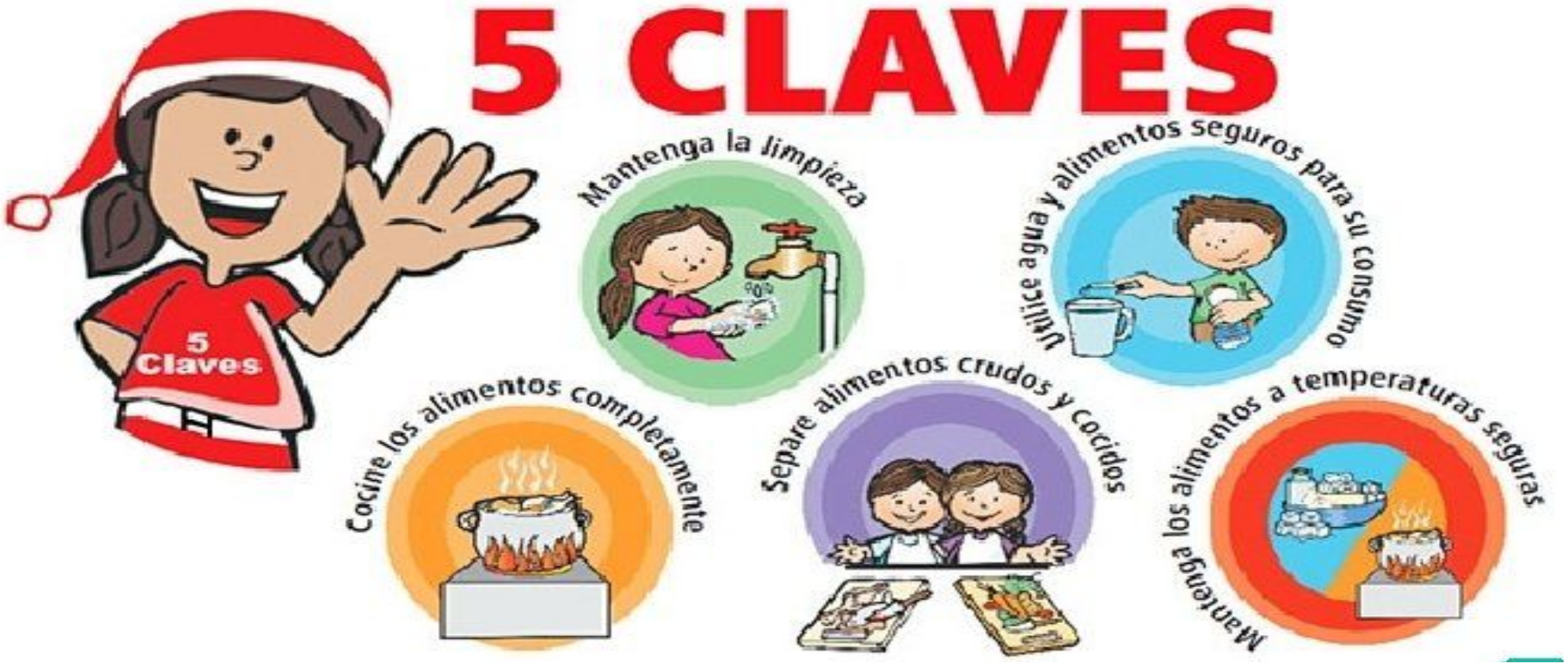
Una interfaz (interface) es sintácticamente similar a una clase abstracta, en la que puede especificar uno o más métodos que no tienen cuerpo ({}).

Esos métodos deben ser implementados por una clase para que se definan sus acciones.

Por lo tanto, una interfaz especifica qué se debe hacer, pero no cómo hacerlo. Una vez que se define una interfaz, cualquier cantidad de clases puede implementarla.

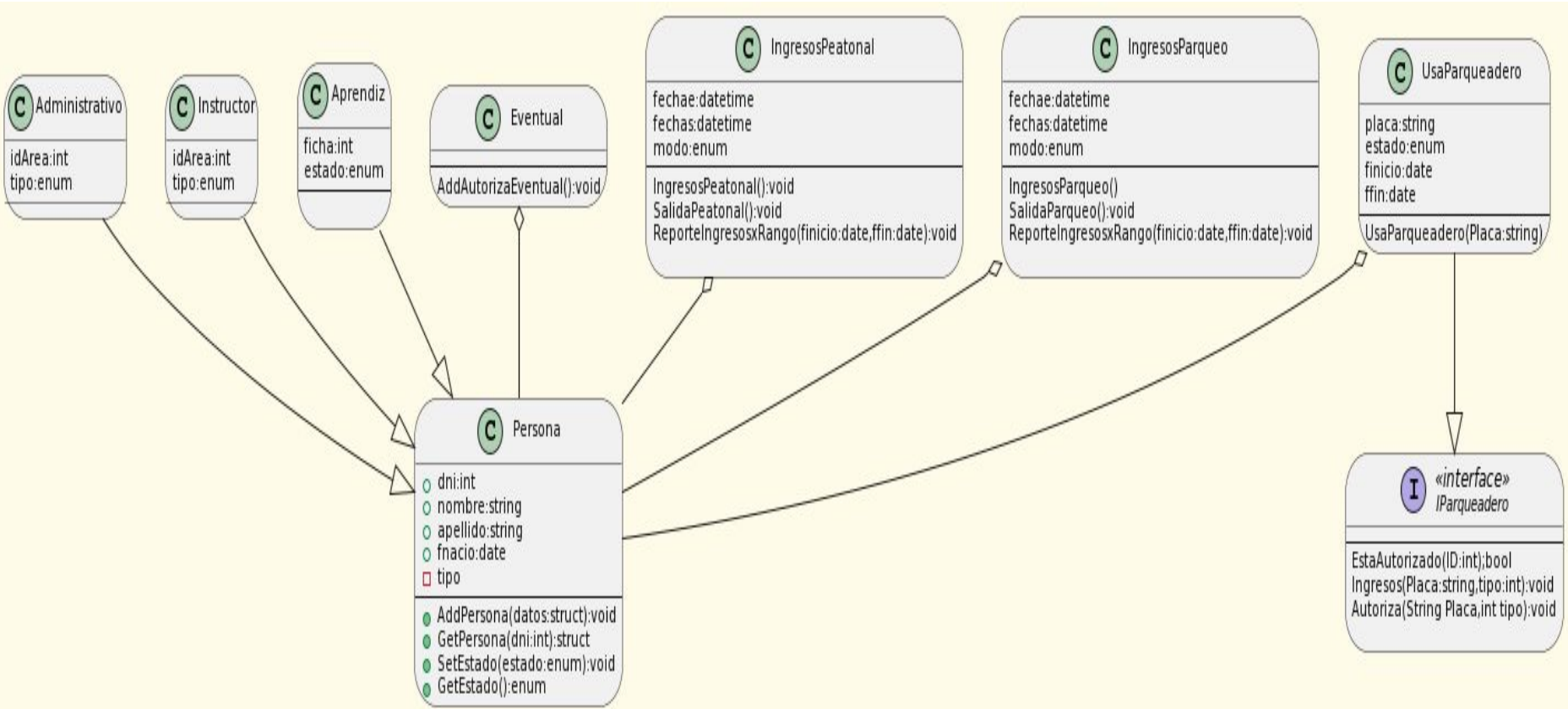
Además, una clase puede implementar cualquier cantidad de interfaces.

ACCIONES A REALIZAR EN LA COCINA

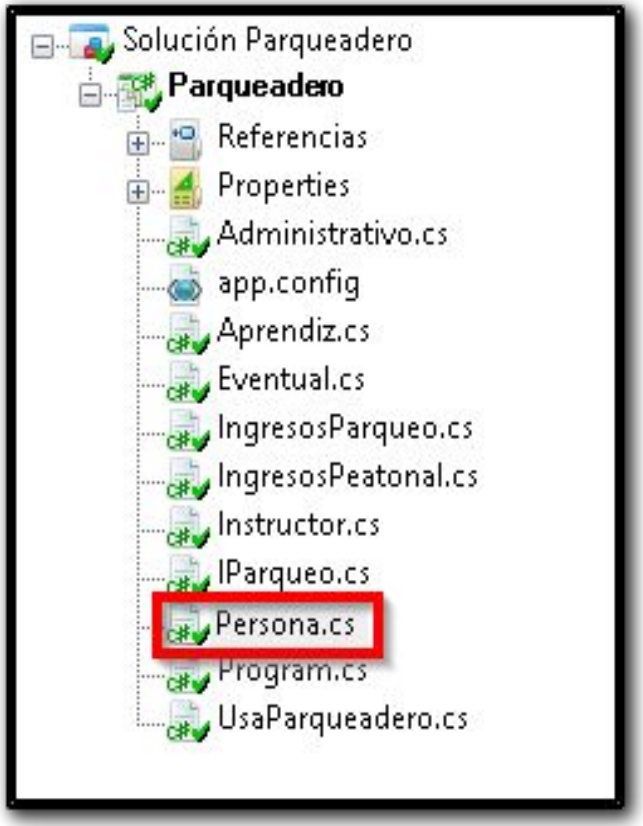


PRÁCTICA EN EL AMBIENTE DE APRENDIZAJE

INTERFACES - CASO DE ESTUDIO

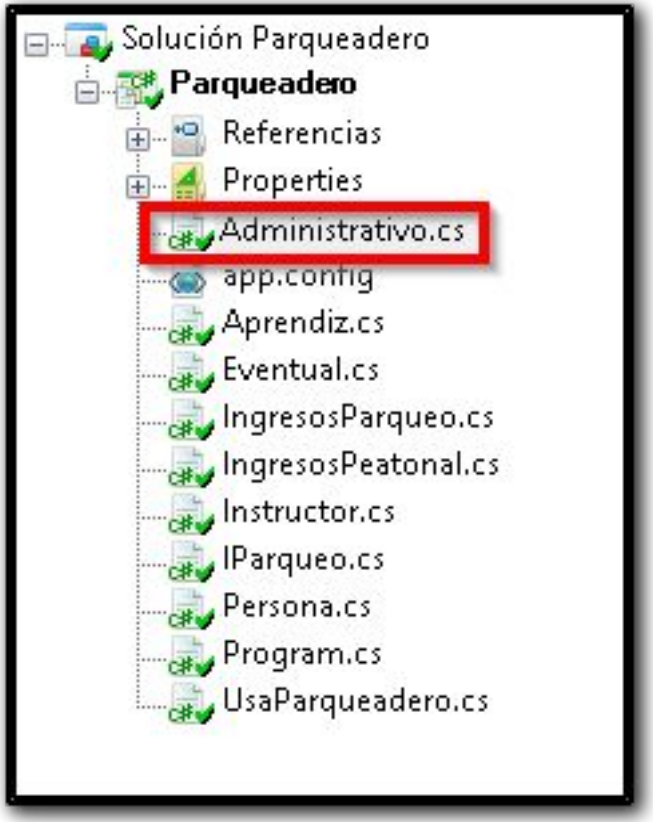


INTERFACES



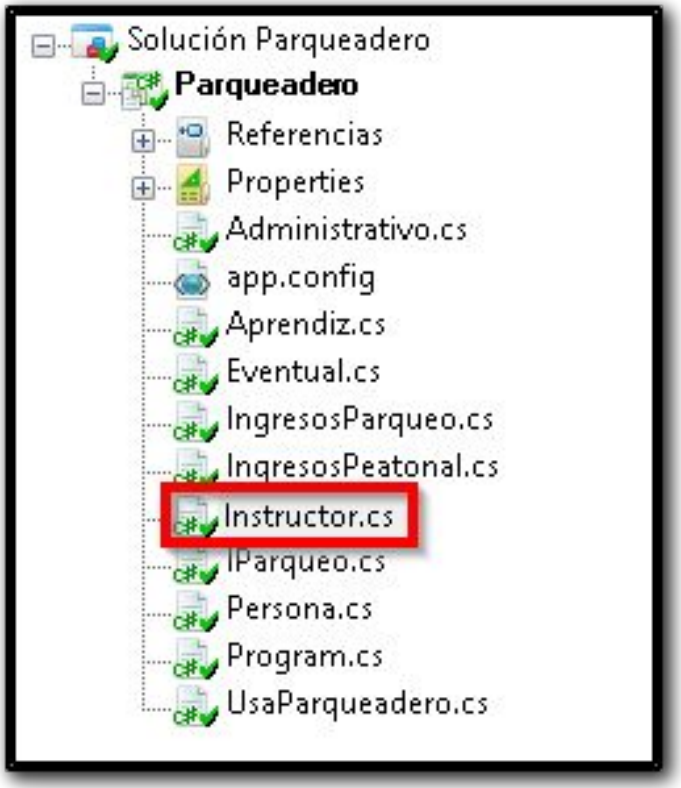

```
using System;

namespace Parqueadero
{
    /// <summary>
    /// Description of Persona.
    /// </summary>
    public class Persona
    {
        public enum Etipo{ 1
            InstructorPlanta,
            InstructorContrato,
            AdministrativoPlanta,
            AdministrativoContratista,
            Aprendiz,
            Egresado
        }
        public struct DatosPer{ 2
        public int dni;
        public string nombre;
        public string apellido;
        public DateTime fnacio ;
        public Etipo tipo;
        }
        public DatosPer DPesonales; 3
        public void AdicionPersona(DatosPer datos) 4
        {

        }
    }
}
```

INTERFACES



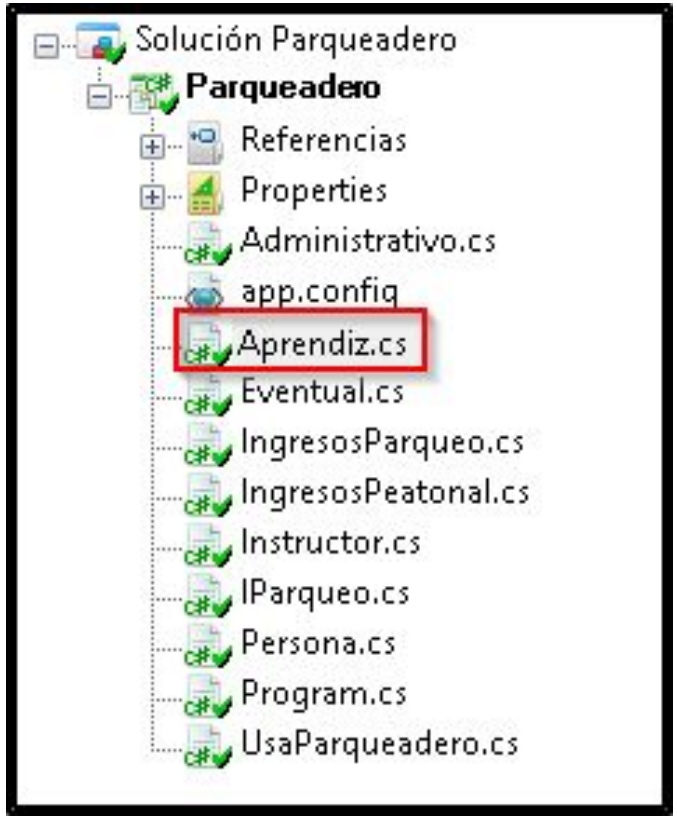

```
using System;

namespace Parqueadero
{
    /// <summary>
    /// Description of Administrativo.
    /// </summary>
    public class Administrativo:Persona
    {
        enum Etipo{ 1
            Planta,
            Contrato
        }
        int idarea; 2
        Etipo tipo;
        public Administrativo() 3
        {
        }
    }
}
```

INTERFACES




```
using System;

namespace Parqueadero
{
    /// <summary>
    /// Description of Instructor.
    /// </summary>
    public class Instructor:Persona
    {
        enum Etipo{ 1
            Planta,
            Contrato
        }
        int idarea; 2
        Etipo tipo;
        public Instructor() 3
        {
        }
    }
}
```

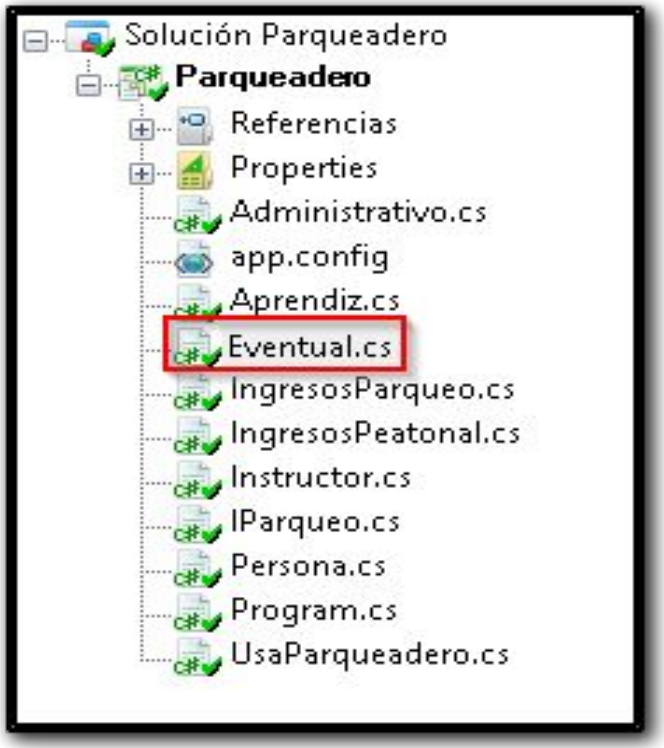
INTERFACES

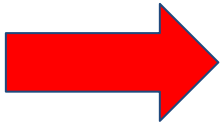



```
using System;

namespace Parqueadero
{
    /// <summary>
    /// Description of Aprendiz.
    /// </summary>
    public class Aprendiz:Persona
    {enum EEstado{
            Inactivo,
            Activo
        }
        int ficha;
        EEstado estado{get;set;}
        public Aprendiz()
        {
        }
    }
}
```

1 2 3

INTERFACES

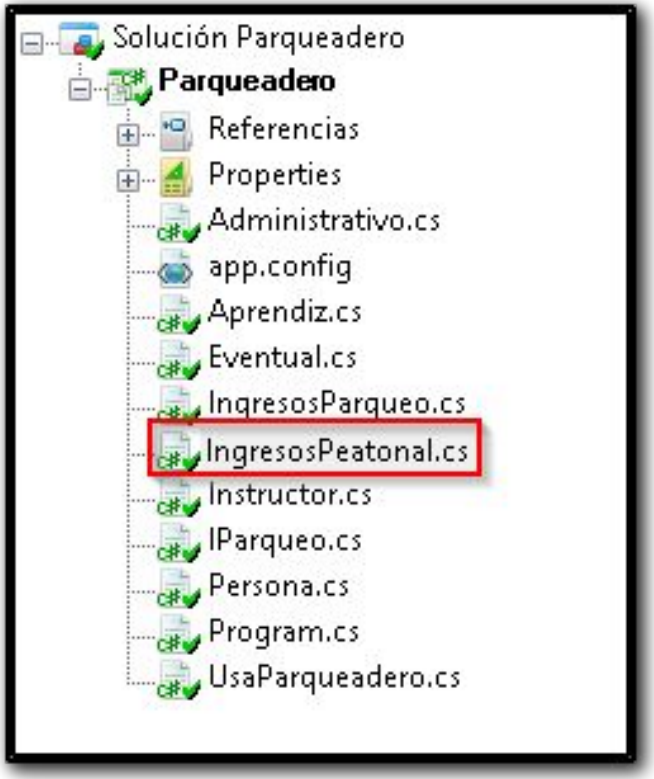




```
using System;

namespace Parqueadero
{
    /// <summary>
    /// Description of Eventual.
    /// </summary>
    public class Eventual:Persona
    {
        public void AutorizaEventual()
        {
        }
    }
}
```

1

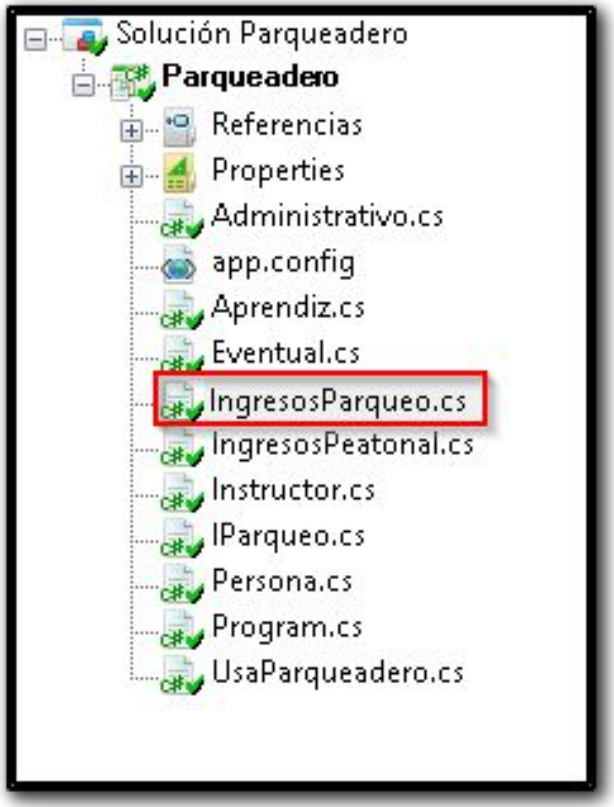
INTERFACES




```
using System;
namespace Parqueadero
{
    public class IngresosPeatonal:Persona
    {
        public enum Emodo{
            entra,
            sale
        }
        public struct Datos{
            public DateTime fechae;
            public DateTime fechas;
            public Emodo modo{get;set;}

        }
        public Datos DatosE;
        public IngresosPeatonal()
        {
            DatosE.fechae=DateTime.Now;
            DatosE.modo=Emodo.entra;
        }
        public void SalidaPeatonal(){
            DatosE.fechas=DateTime.Now;
            DatosE.modo=Emodo.sale;
        }
        public void ReporteIngresosxRango(DateTime fechai,DateTime fechas){

        }

    }
}
```

1 2 3 4 5 6

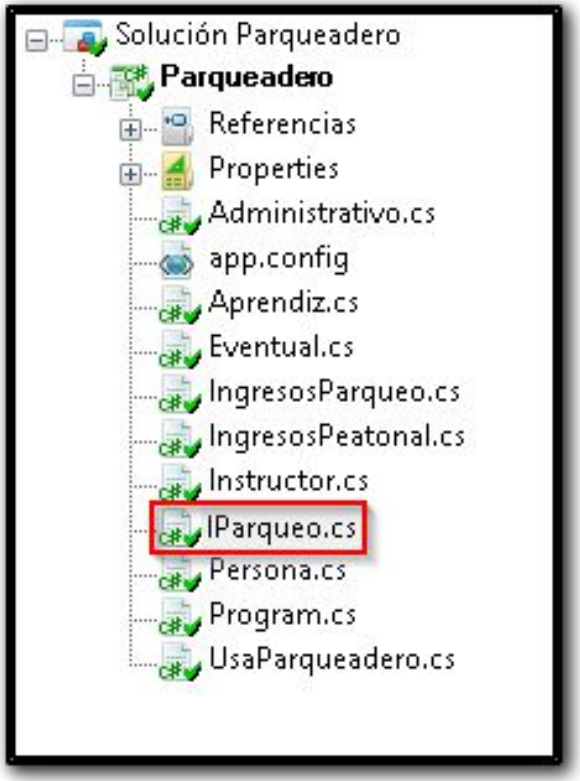
INTERFACES




```
using System;
namespace Parqueadero
{
    public class IngresosParqueo
    {
        public enum Emodo{
            entra,
            sale
        }
        public struct Datos{
            public DateTime fechae;
            public DateTime fechas;
            public Emodo modo{get;set;}

        }
        public Datos DatosE;

        public IngresosParqueo()
        {
            DatosE.fechae=DateTime.Now;
            DatosE.modo=Emodo.entra;
        }
        public void SalidaParqueo(){
            DatosE.fechas=DateTime.Now;
            DatosE.modo=Emodo.sale;
        }
        public void ReporteIngresosxRango(DateTime fechai,DateTime fechas){

        }
    }
}
```

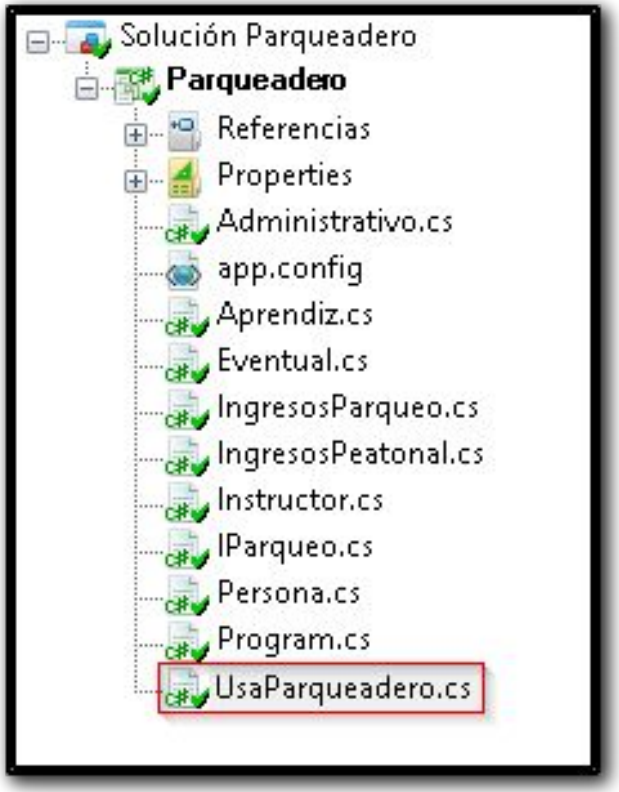
1 2 3 4 5 6

INTERFACES




```
using System;

namespace Parqueadero
{
    public interface IParqueo 1
    {
    2 bool EstaAutorizado(int ID);
      void Ingresos(string Placa,int tipo); 3
    4 void Autoriza(string Placa,int tipo);
    }
}
```

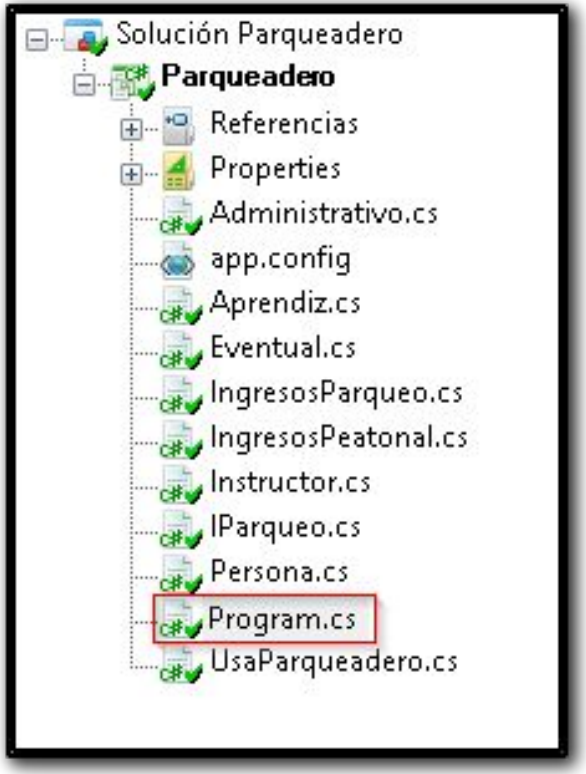
INTERFACES




```
using System;
namespace Parqueadero
{
    public class UsaParqueadero:Persona,IParqueo
    {
        public enum EEstado{Inactivo,Activo}
        public struct SDatos{
            public string placa;
            public DateTime fechai;
            public DateTime fechaf;
            public EEstado estado;
        }
        public SDatos Datos;
        public UsaParqueadero(string Placa)
        {
            Datos.placa=Placa;
            Datos.fechai=DateTime.Now;
        }
        public bool EstaAutorizado(int ID){return true;}
        public void Ingresos(string Placa,int tipo){}
        public void Autoriza(string Placa,int tipo){}
    }
}
```

1 2 3 4 5 6 7 8

INTERFACES




```
using System;
namespace Parqueadero
{
    class Program
    {
        public static void Main(string[] args)
        {
            1 var i=new Instructor();
              i.DPesonales.dni=51693445;
              i.DPesonales.nombre="Zoila";
              i.DPesonales.apellido="Vaca";
              i.DPesonales.tipo=0;
              // TODO: Implement Functionality Here
              Console.WriteLine("DNI={0}\nNombre={1} {2}\n",i.DPesonales.dni,i.DPesonales.nombre,i.DPesonales.apellido);
            3 Console.Write("Press any key to continue . . . ");
              Console.ReadKey(true);
        }
    }
}
```

2

```
DNI=51693445
Nombre=Zoila Vaca

Press any key to continue . . .
```

EVIDENCIA DE APRENDIZAJE DESEMPEÑO

EVIDENCIA DE APRENDIZAJE

Desarrollar los códigos de programación propuestos, ejecutarlos y entregarlos al instructor subiendo cada salida y el código a TERRITORIUM.

<u>Aprendiz</u>, no olvide subir esta evidencia a su portafolio de evidencias.

DESCARGAR ARCHIVOS

COMPILADOR SHARPDEV

VERSIÓN PDF

CRÉDITOS

Realizado por el instructor José Fernando Galindo Suárez
jgalindos@sena.edu.co 2022

G R A C I A S

Línea de atención al ciudadano: 018000 910270
Línea de atención al empresario: 018000 910682

@SENAcomunica

www.sena.edu.co