El programa creado se usa para la gestión de un estacionamiento.
(Tiene cocheras designadas para motos y cocheras designadas para auto o camioneta)
En las funcionalidades que podemos encontrar tenemos el alta del vehículo y el cobrado / baja.

El estacionamiento tiene una capacidad de 75 autos o camionetas y 35 motos.

Temas de clases implementados:

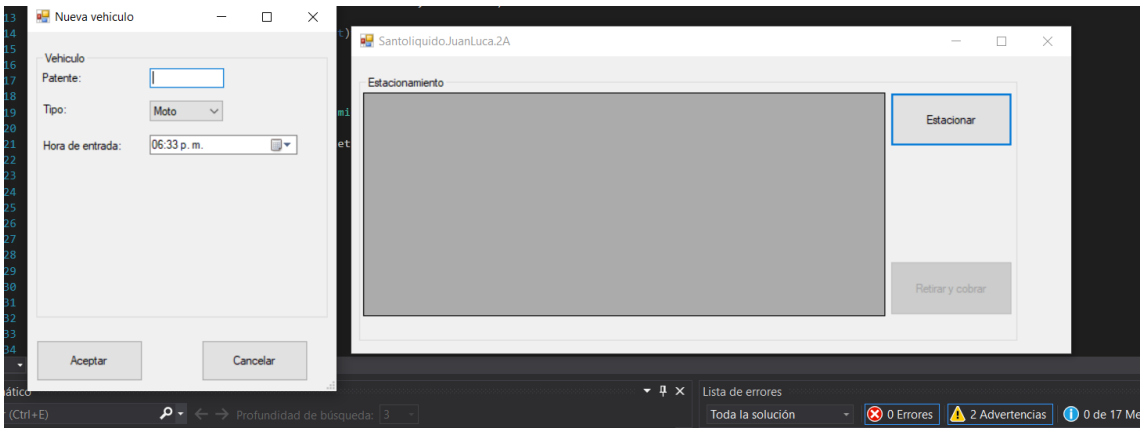
Clase 4 – Sobrecarga: Se sobrecargaron operadores como el + y – para la adicion o sustracción del estacionamiento.

```
/// <returns> Devuelve true or false si puede agregarlo</returns>
3 referencias
public static bool operator +(Estacionamiento e, Vehiculo v)
{
    if ((object)e != null && (object)v != null)
    {
    if (e != v)
    {

            if(v is Auto || v is Camioneta)
            {
                e.EspacioAutoCamioneta--;

            }else
            {
                e.EspacioMoto--;
            }
            e.Vehiculos.Add(v);
            return true;
```

Clase 6 – Windows forms: Se implementaron 2 formularios, 1 para la vista general y otro para el alta o baja del vehiculo.



Clase 7 – Colecciones: Se uso una colección del tipo List<Vehiculo> para el registro de los vehículos.

```
2 referencias
public Estacionamiento()
{
    this.Vehiculos = new List<Vehiculo>();
    this._espacioAutoCamioneta = 75;
```

Clase 8 – Encapsulamiento y propiedades: Se usaron propiedades y encapsulamiento para la clase Estacionamiento.

```
private int _espacioAutoCamioneta;
private int _espacioMoto;
private List<Vehiculo> _vehiculos;


/// <summary>
/// propiedad: carga los atributos
/// </summary>

9 referencias
public List<Vehiculo> Vehiculos
{
    get { return _vehiculos; }
    set { _vehiculos = value; }
}

/// <summary>
/// propiedad: carga los atributos
/// </summary>
4 referencias
public int EspacioAutoCamioneta
{
    get { return _espacioAutoCamioneta; }
    set { _espacioAutoCamioneta = value; }
}

3 referencias
public int EspacioMoto
{
    get { return _espacioMoto; }
    set { _espacioMoto = value; }
}
```

Clase 9 – Herencia: Se implemento herencia entre la clase Vehiculo y sus diferentes clases hijas.

```
5 referencias
public class Moto : Vehiculo
{



    /// <summary>
    /// Constructor por defecto
    /// </summary>
    0 referencias
    public Moto() : base()
    {

    }

    /// <summary>
    /// Constructor con parametros
    /// </summary>
    1 referencia
    public Moto(string patente, EPrecios tipo, DateTime horaEntrada) : base(patente, tipo, horaEntrada)
    {


    }

    0 referencias
    public Moto(string patente, EPrecios tipo, DateTime horaEntrada, DateTime horaSalida) : base(patente, tipo, horaEntrada, horaSalida)
    {
```

Clase 10 – Abstracción: Se implemento la abstracción en la clase Vehiculo

```
public abstract class Vehiculo
{
```

Clase 11 – Polimorfismo: Se uso polimorfismo en la lista Vehiculos

```
private void ConfigurarGrilla()
{
    this.dtGdViewEstacionamiento.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    this.dtGdViewEstacionamiento.MultiSelect = false;
    DataGridViewCellStyle style = new DataGridViewCellStyle();

    style.Font = new Font(this.dtGdViewEstacionamiento.Font, FontStyle.Bold);

    this.dtGdViewEstacionamiento.DefaultCellStyle = style;
    this.dtGdViewEstacionamiento.ColumnHeadersDefaultCellStyle = style;

    foreach (DataGridViewRow item in this.dtGdViewEstacionamiento.Rows)
    {
        if ((this.ES.Vehiculos[item.Index]) is Auto || (this.ES.Vehiculos[item.Index]) is Camioneta)
        {
            item.DefaultCellStyle.BackColor = Color.Green;
        }
        else
        {
            item.DefaultCellStyle.BackColor = Color.LightBlue;
        }
    }
}
```