Errores TP2

```
Rocha, Francisco
(object)v1 == null  -> Utilizar algo como object.ReferenceEquals o similar, en vez del casteo
                    -> La lógica está mal planteada con IF y luego ese otro IF en vez de
                    ELSE, vuelve a castear y comparar
                    -> Mismos errores de Arispe Ledesma, solo agrega un ELSE mal planteado
sb.AppendFormat("TAMAÑO : {0}\n", this.Tamanio); puede ir en la base, para no repetir código
(simil polimorfismo)
public Sedan seria mejor que se reutilicen al reves sus constructores
operator + count int? Si toma 2 valores, es boolean
string Listar lógica mal
```

Error 1 en el operador == de vehículo:

(object)v1 == null -> Utilizar algo como object. ReferenceEquals o similar, en vez del casteo

-> La lógica está mal planteada con IF y luego ese otro IF en vez de

ELSE, vuelve a castear y comparar

-> Mismos errores de Arispe Ledesma, solo agrega un ELSE mal planteado

Antes:

```
public static bool operator ==(Vehiculo v1, Vehiculo v2)
{
    bool retorno = false;

    if (((object)v1) == null && ((object)v2) == null)
    {
        retorno = true;
    }
    else
    {
        if (((object)v1) != null && ((object)v2) != null)
        {
            if (v1.chasis == v2.chasis)
            {
                retorno = true;
            }
        }
    }

    return retorno;
}
```

Corrección:

Clase vehiculo:

```
public static bool operator ==(Vehiculo v1, Vehiculo v2)
{
    return (v1.chasis == v2.chasis);
}
```

Error 2 en la clase base Vehículo y en las clases heredadas Sedan, Suv y Ciclomotor

sb.AppendFormat("TAMAÑO : {0}\n", this.Tamanio); puede ir en la base, para no repetir código

(simil polimorfismo)

Antes:

```
public static explicit operator string(Vehiculo p)
{
    StringBuilder sb = new StringBuilder();

    sb.AppendFormat("CHASIS: {0}\r\n", p.chasis);
    sb.AppendFormat("MARCA : {0}\r\n", p.marca.ToString());
    sb.AppendFormat("COLOR : {0}\r\n", p.color.ToString());
    sb.Append("---------------------\n");

    return sb.ToString();
}
```

```
public override string Mostrar()
{
    StringBuilder sb = new StringBuilder();

    sb.AppendLine("SUV");
    sb.AppendLine(base.Mostrar());
    sb.AppendFormat("TAMAÑO : {0}\n", this.Tamanio);
    sb.AppendLine("---------------------");

    return sb.ToString();
}
```

Corrección:

Clase base:

```
/// <summary>
/// Sobrecarca explicita del operador string retornando los datos del vehiculo
/// </summary>
/// <param name="p">Vehiculo del cual se toman los datos</param>
public static explicit operator string(Vehiculo p)
{
    StringBuilder sb = new StringBuilder();

    sb.AppendFormat("CHASIS: {0}\r\n", p.chasis);
    sb.AppendFormat("MARCA : {0}\r\n", p.marca.ToString());
    sb.AppendFormat("COLOR : {0}\r\n", p.color.ToString());
    sb.AppendLine("---------------------\n");
    sb.AppendFormat("TAMAÑO : {0}\r", p.Tamanio.ToString());

    return sb.ToString();
}
```

Clases heredadas (Sedan, Suv y Ciclomotor):

```
/// <summary>
/// Metodo que sobrescribe al de la clase base mostrando la informacion de Vehiculo y Suv
/// </summary>
/// <returns>Retorna la informacion de Suv y Vehiculo en formato string</returns>
8 references
public override string Mostrar()
{
    StringBuilder sb = new StringBuilder();

    sb.AppendLine("SUV");
    sb.AppendLine(base.Mostrar());
    sb.Append("--------------------\n");

    return sb.ToString();
}
```

Error 3 en la clase Sedan:

public Sedan sería mejor que se reutilicen al revés sus constructores

Antes:

```
2 references
public Sedan(EMarca marca, string chasis, ConsoleColor color) : base(chasis, marca, color)
{
    this.tipo = ETipo.CuatroPuertas;
}

/// <summary>
/// Constructor de sedan que le asigna un tipo pasado por parametro al atributo "tipo"
/// </summary>
/// <param name="marca">Le asigna una marca al atributo marca de vehiculo</param>
/// <param name="chasis">Le asigna un chasis al atributo chasis de vehiculo></param>
/// <param name="color">Le asigna un color al atributo color de vehiculo</param>
/// <param name="tipo">Le asigna un tipo al atributo tipo</param>
1 reference
public Sedan(EMarca marca, string chasis, ConsoleColor color, ETipo tipo) : this(marca, chasis, color)
{
    this.tipo = tipo;
}
```

Corrección:

Clase sedan:

```
2 references
public Sedan(EMarca marca, string chasis, ConsoleColor color, ETipo tipo) : base(chasis, marca, color)
{
    this.tipo = tipo;
}

/// <summary>
/// Constructor de sedan que asigna por defecto el valor "CuatroPuertas" al atributo "tipo"
/// </summary>
/// <param name="marca">Le asigna una marca al atributo marca de vehiculo</param>
/// <param name="chasis">Le asigna un chasis al atributo chasis de vehiculo</param>
/// <param name="color">Le asigna un color al atributo color de vehiculo</param>
1 reference
public Sedan(EMarca marca, string chasis, ConsoleColor color) : this(marca, chasis, color, ETipo.CuatroPuertas)
{

}
```

Error 4 en la clase Taller:

operator + count int? Si toma 2 valores, es boolean

Antes:

```
public static Taller operator +(Taller taller, Vehiculo vehiculo)
{
    int bandera = 0;

    foreach (Vehiculo v in taller.vehiculos)
    {
        if (v == vehiculo)
        {
            bandera = 1;
            break;
        }
    }

    if(bandera == 0 && taller.vehiculos.Count < taller.espacioDisponible)
    {
        taller.vehiculos.Add(vehiculo);
    }

    return taller;
}
```

Corrección:

Clase taller:

```
10 references
public static Taller operator +(Taller taller, Vehiculo vehiculo)
{
    bool bandera = false;

    foreach (Vehiculo v in taller.vehiculos)
    {
        if (v == vehiculo)
        {
            bandera = true;
            break;
        }
    }

    if(!bandera && taller.vehiculos.Count < taller.espacioDisponible)
    {
        taller.vehiculos.Add(vehiculo);
    }

    return taller;
}
```

Error 5 en la clase Taller:

string Listar lógica mal

Antes:

```
public string Listar(Taller taller, ETipo tipo)
{
    StringBuilder sb = new StringBuilder();

    sb.AppendFormat("Tenemos {0} lugares ocupados de un total de {1} disponibles\n", taller.vehiculos.Count, taller.espacioDisponible);

    foreach (Vehiculo v in taller.vehiculos)
    {
        switch (tipo)
        {
            case ETipo.Ciclomotor:
                if(v is Ciclomotor)
                {
                    sb.AppendLine(v.Mostrar());
                }
                break;
            case ETipo.Sedan:
                if (v is Sedan)
                {
                    sb.AppendLine(v.Mostrar());
                }
                break;
            case ETipo.SUV:
                if (v is Suv)
                {
                    sb.AppendLine(v.Mostrar());
                }
                break;
            case ETipo.Todos:
                sb.AppendLine(v.Mostrar());
                break;
        }
    }

    return sb.ToString();
}
```

Corrección:

Clase taller:

```
4 references
public string Listar(Taller taller, ETipo tipo)
{
    StringBuilder sb = new StringBuilder();

    sb.AppendFormat("Tenemos {0} lugares ocupados de un total de {1} disponibles\n", taller.vehiculos.Count, taller.espacioDisponible);

    foreach (Vehiculo v in taller.vehiculos)
    {
        switch (tipo)
        {
            case ETipo.Ciclomotor:
                if(v is Ciclomotor)
                {
                    sb.AppendLine(v.Mostrar());
                }
                break;
            case ETipo.Sedan:
                if (v is Sedan)
                {
                    sb.AppendLine(v.Mostrar());
                }
                break;
            case ETipo.SUV:
                if (v is Suv)
                {
                    sb.AppendLine(v.Mostrar());
                }
                break;
            default:
                sb.AppendLine(v.Mostrar());
                break;
        }
    }

    return sb.ToString();
```