Ejercicio 1

```
abstact class Figura
{ public abstract double Calcular(); }

class Rectangulo : Figura
{ public double Ancho{get; set;}
  public double Largo; {get; set;}

  public Rectangulo(double Ancho, double largo)
  { this.Ancho = ancho;
    This. Largo = largo; }

  public override double Calcular()
  { return Ancho.LArgo;}}

class Circulo : Figura
{ public double Radio{get; set;}

  public Circulo(double radio)
  { this.Radio = radio; }

Public override double Calcular()
  { return MAth.Pi * MAth.Pow(radio, 2);}}

class Form1: Form

button VerArea:
{ List<Figura> Figuras = new List<Figura>();
  Figura figuraA1 = new Rectangulo (2;3);
  Figura f2 = new Rectalo (3; 5);
  figura F3 = new Circulo (2);
  figuras. Add (Figura1);
  figuras. Add (F2);
  figuras. Add (f3);
```

```
foreach (figura A in figuras)
{ listBox.Item.Add(A.Calcular()); }}
```

Ejercicio 2

```
interface IFigura
{ ~~calcular~~ double Calcular(); }

class Rectangulo : IFigura
{ public double Ancho {get; set;}
  public double Largo { get; set; };

  public Rectangulo (dobble ancho, double largo)
  { this.Ancho = ancho;
    Inis.largo = largo; }

{ public ~~area~~ double Calcular ()
  {   return this.Largo * this.Ancho; }}

pibass Circulo : IFigura
{ public double Radio {get; set;}
  public Circulo (double radio)
  { this.Radio = radio }
  public ~~area~~ Calcular ()
  { return Math.Pi * Math.Power(this.Radio, 2); }}

class Form1:
{ List <IFigura> lista = new List <IFigura>();
    Lista.Add( new Rectangulo (2,3);
    lista.Add( new Circulo (3);
Button CalcularArea
{  foreach (IFigura A in lista)
  { listBox.Items.Add ( A.Calcular()); } }
```

Ejercicio 3

```
class Persona
{ public string Nombre { get; private set; }
  public Persona (string nombre)
  { this.Nombre = nombre; }

  public virtual string Double ()
    { return this.Nombre; }

class PersonaJuridica : Persona
{ public string CUIT { get; private set; }
  public PersonaJuridico (string nombre, string cuit) : base(nombre)
  { this.CUIT = cuit; }

  public override string Describir ()
  { retur base.Describir() + " - " + this.CUIT; } }

class Form1
{ List <Persona> lista = new List <Persona> () {
      new Persona ("MARIA"), new PersonaJuridico ("Ana", "3002866") };
  button ver_descripcion
  { foreach (Persona A in lista)
    { listbox.Items.Add(A.Describir()); } }
```

Ejercicio 4

```
class Persona : IComparable
{ public string Nombre {get; set;}

Public Persona (string Nombre)
 { this.Nombre = nombre; }
Public virtual string Describir ()
 { return this.Nombre; }
public int CompareTo (Object p)
    { Persona nueva = p as Persona;
    if (nueva != null)
   { return (this.Nombre.CompareTo (nueva.Nombre)); }
   else
    { return 0; }

class PersonaJuridica : Persona
{ public string Cuit {get; set;}
 public PersonaJuridica (string nombre, string cuit)
    : base (nombre) { this.cuit = cuit };
public override string Describir ()
    { return base.Describir + "-" + this.cuit ; }

class Form1 {
List <Persona> lista = new List<Persona> { new Persona ("ANA"),
 new Persona ("Alan"), new PersonaJuridica ("Cel:", "353350") }
button Buscar:
{ string buscar = textbox.Text;
   lista.SORT();
   Persona p = new Persona (buscar, "...");
 int pos = lista.BinarySeach(p);
  if (pos > -1) { Message.Show ($"la persona es {Lista[pos].Describir()}
        en la posición {pos}");
```