Universidad ORT Uruguay
Instituto de Educación

Primera parte del Obligatorio de Programación 2

2ª entrega correspondiente a la carrera Analista en Tecnologías de la Información / Analista Programador

Facundo Piegas - 217074
Marcos Russo – 217911
Grupo N2A
Docente: Andrés de Sosa

2017

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Dominio
8  {
9      public class Administrador : Usuario
10     {
11         #region Contructor
12         public Administrador(string unNombreUsuario, string unaClave) : base
             (unNombreUsuario, unaClave)
13         {
14         }
15         public override int devolverTipo()
16         {
17             return 1;
18         }
19         #endregion
20
21         #region Metodos Sobreescrios
22         public override bool Equals(object obj)
23         {
24             Usuario otro = obj as Usuario;
25             return this.NombreUsuario.Equals(otro.NombreUsuario);
26         }
27
28         public int CompareTo(Usuario other)
29         {
30             int res;
31             if (this.NombreUsuario.CompareTo(other.NombreUsuario) < 0) // alias
                  de this esta antes que el alias de other
32             {
33                 res = -1;
34             }
35             else
36             {
37                 if (this.NombreUsuario.CompareTo(other.NombreUsuario) > 0)
38                 {
39                     res = 1;
40                 }
41                 else
42                 {
43                     res = 0;
44                 }
45             }
46
47             return res;
48         }
49         #endregion
50     }
51
52
53 }
54
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Dominio
8  {
9      public class Comprador : Usuario
10     {
11         #region Atributos y Properties
12         public string Direccion { set; get; }
13         public string Telefono { set; get; }
14         public string SenaAPagar { set; get; }
15         public string NomLista { set; get; }
16
17         public List<LoteComprado> listaLotesComprados { set; get; }
18
19         #endregion
20
21         #region Constructores
22         public Comprador(string NombreUsuario, string unaClave, string
             unaDireccion, string unTelefono) : base(NombreUsuario, unaClave)
23         {
24             this.Direccion = unaDireccion;
25             this.Telefono = unTelefono;
26             this.listaLotesComprados = new List<LoteComprado>();
27
28         }
29         public override int devolverTipo()
30         {
31             return 3;
32         }
33
34         public double devolverComisionAPagar(int unIdRemate)
35         {
36             double comision=0;
37             foreach(LoteComprado R in listaLotesComprados)
38             {
39                 if(R.Remate.IdRemate == unIdRemate)
40                 {
41                     comision = R.calcularComisionAPagar();
42                 }
43             }
44             this.SenaAPagar = "$ " + comision.ToString();
45             return comision;
46         }
47
48         public bool participoDeRemate(int unIdRemate)
49         {
50             bool participoDeRemate = false;
51             foreach(LoteComprado L in listaLotesComprados)
52             {
53                 if(L.Remate.IdRemate == unIdRemate)
54                 {
55                     participoDeRemate = true;
```

```
56                  }
57              }
58              return participoDeRemate;
59          }
60          #endregion
61      }
62  }
63
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Dominio
8  {
9      public class Empresa : Comprador
10     {
11         #region Atributos y Properties
12         public string PersonaContacto { set; get; }
13         public string RazonSocial { set; get; }
14         public string Rut { set; get; }
15         #endregion
16
17         #region Constructores
18         public Empresa(string unNombreUsuario, string unaClave, string
             unaDireccion, string unTelefono, string unaPersonaContacto, string
             unaRazonSocial, string unRut) : base(unNombreUsuario, unaClave,
             unaDireccion, unTelefono)
19         {
20             this.NomLista = unaRazonSocial;
21             this.PersonaContacto = unaPersonaContacto;
22             this.RazonSocial = unaRazonSocial;
23             this.Rut = unRut;
24         }
25         #endregion
26         #region Metodos Sobreescrios
27         public override bool Equals(object obj)
28         {
29             Usuario otro = obj as Usuario;
30             return this.NombreUsuario.Equals(otro.NombreUsuario);
31         }
32
33         public int CompareTo(Usuario other)
34         {
35             int res;
36             if (this.NombreUsuario.CompareTo(other.NombreUsuario) < 0) // alias
                  de this esta antes qu el alias de other
37             {
38                 res = -1;
39             }
40             else
41             {
42                 if (this.NombreUsuario.CompareTo(other.NombreUsuario) > 0)
43                 {
44                     res = 1;
45                 }
46                 else
47                 {
48                     res = 0;
49                 }
50             }
51
52             return res;
```

```
53          }
54          #endregion
55      }
56  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Dominio
8  {
9      public class Ganado : Remate
10     {
11         #region Atributos y Properties
12         private static double _porcentajeAdicionalRematador = 3;
13         public static double PorcentajeAdicionalRematador
14         {
15             get { return _porcentajeAdicionalRematador; }
16             set { _porcentajeAdicionalRematador = value; }
17         }
18         #endregion
19
20         #region Constructores
21         public Ganado(string unaDescripcion, double unPorcentajeBaseRematador,
             DateTime unaFecha, Lugar unLugar, Rematador unRematador) : base
             (unaDescripcion, unPorcentajeBaseRematador, unaFecha, unLugar,
             unRematador)
22         {
23             // Nada para hacer
24             // Solo se invoca al contructor de la clase base
25         }
26         #endregion
27
28         #region Metodos sobreescritos / heredados
29         public override double calcularComisionAPagarRematador()
30         {
31             double gananciaTotalLotesVendidos = calcularMontoVentasRemate();
32             double comisionRematador = this.PorcentajeBaseRematador;
33             int totalLotes = devolverListaLotesRemate().Count;
34             int lotesVendidos = devolverCantidadLotesVendidos();
35             if(lotesVendidos > (totalLotes / 2)){
36                 comisionRematador += _porcentajeAdicionalRematador;
37                 comisionRematador = (comisionRematador / 100) *
                     gananciaTotalLotesVendidos;
38             }
39             else{
40                 comisionRematador = (comisionRematador / 100) *
                     gananciaTotalLotesVendidos;
41             }
42
43             return comisionRematador;
44         }
45         #endregion
46
47         #region Otros metodos
48
49
50         #endregion
51
```

```
52      }
53  }
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace Dominio
 8  {
 9      public class Lote
10      {
11          #region Atributos y Properties
12
13          private static Random random = new Random();
14          public static string RandomString(int length)
15          {
16              const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
17              return new string(Enumerable.Repeat(chars, length)
18                .Select(s => s[random.Next(s.Length)]).ToArray());
19          }
20
21          private string _idLote;
22          public string IdLote
23          {
24              get { return _idLote; }
25              set { _idLote = value; }
26          }
27
28
29          private string _descripcionLote;
30          public string DescripcionLote
31          {
32              get { return _descripcionLote; }
33              set { _descripcionLote = value; }
34          }
35
36
37          private double _precioBaseLote;
38          public double PrecioBaseLote
39          {
40              get { return _precioBaseLote; }
41              set { _precioBaseLote = value; }
42          }
43
44
45          private int _cantidadUnidadesLote;
46          public int CantidadUnidadesLote
47          {
48              get { return _cantidadUnidadesLote; }
49              set { _cantidadUnidadesLote = value; }
50          }
51
52          private bool _fueAsignado;
53          public bool FueAsignado
54          {
55              get { return _fueAsignado; }
56              set { _fueAsignado = value; }
```

```
57          }
58
59          private bool _fueComprado;
60          public bool FueComprado
61          {
62              get { return _fueComprado; }
63              set { _fueComprado = value; }
64          }
65
66          private double _precioCompraLote;
67          public double PrecioCompraLote
68          {
69              get { return _precioCompraLote; }
70              set { _precioCompraLote = value; }
71          }
72
73          public string Foto { set; get; }
74
75          public string DescripcionListaLote
76          {
77              get { return "Descripción:" + "  " + this.DescripcionLote + " - "
                    + "  " + "Precio base:" + "  " + this.PrecioBaseLote; }
78          }
79          #endregion
80
81          #region Constructores
82
83          public Lote(string unaDescripcionLote, double unPrecioBaseLote, int
              unaCantidadUnidadesLote, string unaFoto)
84          {
85              this.IdLote = Lote.RandomString(5);
86              this.DescripcionLote = unaDescripcionLote;
87              this.PrecioBaseLote = unPrecioBaseLote;
88              this.PrecioCompraLote = 0;
89              this.CantidadUnidadesLote = unaCantidadUnidadesLote;
90              this.FueAsignado = false;
91              this.FueComprado = false;
92              this.Foto = unaFoto;
93          }
94          #endregion
95
96          #region Metodos sobreescritos / heredados
97          public override string ToString()
98          {
99              return "ID: " + this.IdLote + "\n" + "Descripción: " +
                      this.DescripcionLote + "\n" + "Cantidad de Unidades: " +
                      this.CantidadUnidadesLote + "\n" + "\n";
100         }
101
102         #endregion
103
104         #region Otros metodos
105
106
107         #endregion
108     }
```

```
109  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Dominio
8  {
9      public class LoteComprado
10     {
11         //ATRIBUTOS
12         public Remate Remate { set; get; }
13         public List<Lote> listaLotesCompradosEnRemate { set; get; }
14
15         public LoteComprado(Remate unRemate)
16         {
17             this.Remate = unRemate;
18             this.listaLotesCompradosEnRemate = new List<Lote>();
19         }
20
21         public double calcularComisionAPagar()
22         {
23             double comisionAPagar = 0;
24             foreach(Lote L in listaLotesCompradosEnRemate)
25             {
26                 comisionAPagar += L.PrecioCompraLote;
27             }
28             comisionAPagar = (comisionAPagar * 0.1) * 1.22; //0.10 (seña) +
                 1.22 (agrego IVA)
29             return comisionAPagar;
30         }
31     }
32 }
33
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Dominio
8  {
9      public class Lugar
10     {
11         #region Atributos y Properties
12         // Autonumerado
13         private static int _UltimoNro = 0;
14
15         public static int UltimoNumero
16         {
17             get { return _UltimoNro; }
18
19         }
20
21         // Atributos de Instancia
22         private string _nombreLugar;
23         public string NombreLugar
24         {
25             get { return _nombreLugar; }
26             set { _nombreLugar = value; }
27         }
28
29         private int _idLugar;
30         public int IdLugar
31         {
32             get { return _idLugar; }
33             set { _idLugar = value; }
34         }
35
36         private string _calleLugar;
37         public string CalleLugar
38         {
39             get { return _calleLugar; }
40             set { _calleLugar = value; }
41         }
42
43         private string _numeroPuertaLugar;
44         public string NumeroPuertaLugar
45         {
46             get { return _numeroPuertaLugar; }
47             set { _numeroPuertaLugar = value; }
48         }
49
50         private string _ciudadLugar;
51         public string CiudadLugar
52         {
53             get { return _ciudadLugar; }
54             set { _ciudadLugar = value; }
55         }
56         #endregion
```

```
57
58          #region Constructores
59          public Lugar(string unNombreLugar, string unaCalleLugar, string
              unNumeroPuertaLugar, string unaCiudadLugar)
60          {
61              _UltimoNro++;
62              this.IdLugar = Lugar._UltimoNro;
63              this.NombreLugar = unNombreLugar;
64              this.CalleLugar = unaCalleLugar;
65              this.NumeroPuertaLugar = unNumeroPuertaLugar;
66              this.CiudadLugar = unaCiudadLugar;
67          }
68          #endregion
69
70          #region Metodos sobreescritos / heredados
71          public override string ToString()
72          {
73              return "ID: " + this.IdLugar + "\n" + "Nombre: " + this.NombreLugar
                   + "\n" + "Calle: " + this.CalleLugar + "\n" + "Ciudad: " +
                  this.CiudadLugar + "\n" + "\n";
74          }
75          #endregion
76
77          #region Otros metodos
78
79
80          #endregion
81      }
82  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Dominio
8  {
9      public class Mercaderia : Remate
10     {
11         #region Atributos y Properties
12         private double _porcentajeAdicionalPorLote;
13         public double PorcentajeAdicionalPorLote
14         {
15             get { return _porcentajeAdicionalPorLote; }
16             set { _porcentajeAdicionalPorLote = value; }
17         }
18         #endregion
19
20         #region Constructores
21         public Mercaderia(string unaDescripcion, double
             unPorcentajeBaseRematador, DateTime unaFecha, Lugar unLugar,
             Rematador unRematador, double unPorcentajeAdicionalPorLote) : base
             (unaDescripcion, unPorcentajeBaseRematador, unaFecha, unLugar,
             unRematador)
22         {
23             this.PorcentajeAdicionalPorLote = unPorcentajeAdicionalPorLote;
24         }
25         #endregion
26
27         #region Metodos sobreescritos / heredados
28         public override double calcularComisionAPagarRematador()
29         {
30             double gananciaTotalLotesVendidos = calcularMontoVentasRemate();
31             double comisionRematador = this.PorcentajeBaseRematador;
32             int lotesVendidos = devolverCantidadLotesVendidos();
33             comisionRematador += (PorcentajeAdicionalPorLote * lotesVendidos);
34             comisionRematador = (comisionRematador / 100) *
                 gananciaTotalLotesVendidos;
35             return comisionRematador;
36         }
37         #endregion
38
39         #region Otros metodos
40
41
42         #endregion
43     }
44 }
45
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Dominio
8  {
9      public class Particular : Comprador
10     {
11         #region Atributos y Properties
12         public string NombreParticular { set; get; }
13         public string ApellidoParticular { set; get; }
14         public string DocumentoParticular { set; get; }
15         #endregion
16
17         #region Constructores
18         public Particular(string NombreUsuario, string unaClave, string
             unaDireccion, string unTelefono, string unNombreParticular, string
             unApellidoParticular, string unDocumentoParticular) : base
             (NombreUsuario, unaClave, unaDireccion, unTelefono)
19         {
20             this.NomLista = unNombreParticular + " " + unApellidoParticular;
21             this.NombreParticular = unNombreParticular;
22             this.ApellidoParticular = unApellidoParticular;
23             this.DocumentoParticular = unDocumentoParticular;
24
25
26         }
27         #endregion
28
29         #region Metodos Sobreescrios
30         public override bool Equals(object obj)
31         {
32             Usuario otro = obj as Usuario;
33             return this.NombreUsuario.Equals(otro.NombreUsuario);
34         }
35
36         public int CompareTo(Usuario other)
37         {
38             int res;
39             if (this.NombreUsuario.CompareTo(other.NombreUsuario) < 0) // alias
                  de this esta antes qu el alias de other
40             {
41                 res = -1;
42             }
43             else
44             {
45                 if (this.NombreUsuario.CompareTo(other.NombreUsuario) > 0)
46                 {
47                     res = 1;
48                 }
49                 else
50                 {
51                     res = 0;
52                 }
```

```
53              }
54
55              return res;
56          }
57          #endregion
58      }
59  }
60
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace Dominio
 8  {
 9      public class Rematador : Usuario
10      {
11          #region Atributos y Properties
12          // Autonumerado
13          private static int _UltimoNro = 0;
14
15          public static int UltimoNumero
16          {
17              get { return _UltimoNro; }
18          }
19
20          // Atributos de Instancia
21          private int _idRematador;
22          public int IdRematador
23          {
24              get { return _idRematador; }
25              set { _idRematador = value; }
26          }
27
28          private string _nombreRematador;
29          public string NombreRematador
30          {
31              get { return _nombreRematador; }
32              set { _nombreRematador = value; }
33          }
34
35          private string _apellidoRematador;
36          public string ApellidoRematador
37          {
38              get { return _apellidoRematador; }
39              set { _apellidoRematador = value; }
40          }
41
42          private string _telefonoRematador;
43          public string TelefonoRematador
44          {
45              get { return _telefonoRematador; }
46              set { _telefonoRematador = value; }
47          }
48
49          public string NombreApellido{
50              get
51              {
52                  return this.NombreRematador + "  " + this.ApellidoRematador;
53              }
54          }
55          public string ComisionRematador { set; get; }
56
```

```
57          #endregion
58
59          #region Constructores
60          public Rematador(string unNombreUsuario, string unaClave, string
              unNombreRematador, string unApellidoRematador, string
              unTelefonoRematador) : base(unNombreUsuario, unaClave)
61          {
62              _UltimoNro++;
63              this.IdRematador = Rematador._UltimoNro;
64              this.Clave = unaClave;
65              this.NombreRematador = unNombreRematador;
66              this.ApellidoRematador = unApellidoRematador;
67              this.TelefonoRematador = unTelefonoRematador;
68          }
69          #endregion
70
71          #region Metodos sobreescritos / heredados
72          public override string ToString()
73          {
74              return "ID: " + this.IdRematador + "\n" + "Nombre: " +
                  this.NombreRematador + "\n" + "Apellido: " +
                  this.ApellidoRematador + "\n" + "\n";
75          }
76
77          public override int devolverTipo()
78          {
79              return 2;
80          }
81
82          public override bool Equals(object obj)
83          {
84              Usuario otro = obj as Usuario;
85              return this.NombreUsuario.Equals(otro.NombreUsuario);
86          }
87
88          public int CompareTo(Usuario other)
89          {
90              int res;
91              if (this.NombreUsuario.CompareTo(other.NombreUsuario) < 0) //
                  alias de this esta antes qu el alias de other
92              {
93                  res = -1;
94              }
95              else
96              {
97                  if (this.NombreUsuario.CompareTo(other.NombreUsuario) > 0)
98                  {
99                      res = 1;
100                 }
101                 else
102                 {
103                     res = 0;
104                 }
105             }
106
107             return res;
```

```
108         }
109         #endregion
110
111         #region Otros metodos
112
113
114         #endregion
115     }
116 }
117
```

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

namespace Dominio
{
    public class Rematadora
    {
        #region Atributos y Properties
        private static Rematadora instancia;

        public static Rematadora Instancia
        {
            get
            {
                if (instancia == null)
                {
                    instancia = new Rematadora();
                    instancia.cargaObjetosPrueba();
                }
                return instancia;
            }
        }
        #endregion

        #region Constructores
        private List<Remate> listaRemates;
        private List<Rematador> listaRematadores;
        private List<Lugar> listaLugares;
        private List<Lote> listaLotes;
        private List<Usuario> listaUsuarios;
        private List<Comprador> listaCompradores;
        private Rematadora()
        {
            listaRemates = new List<Remate>();
            listaRematadores = new List<Rematador>();
            listaLugares = new List<Lugar>();
            listaLotes = new List<Lote>();
            listaUsuarios = new List<Usuario>();
            listaCompradores = new List<Comprador>();
        }
        #endregion

        #region Metodos
        public void cargaObjetosPrueba()
        {
            //precargo datos del lote de pueba
            crearLote("Lote de Prueba 1", 15000, 1, "imagenes\\ad.jpg");
            crearLote("Lote de Prueba 2", 16000, 2, "imagenes\\ad.jpg");
            crearLote("Lote de Prueba 3", 17000, 3, "imagenes\\ad.jpg");
            crearLote("Lote de Prueba 4", 18000, 4, "imagenes\\ad.jpg");
            crearLote("Lote de Prueba 5", 19000, 5, "imagenes\\ad.jpg");
            //precargo usuarios
            crearUsuarioAdministrador("1", "1");
```

```
57              crearRematador("rem1", "rem1", "Carlos1", "Bueno", "099868407");
58              crearRematador("rem2", "rem2", "Carlos2", "Bueno", "099868407");
59              crearRematador("rem3", "rem3", "Carlos3", "Bueno", "099868407");
60              crearRematador("rem4", "rem4", "Carlos4", "Bueno", "099868407");
61              crearUsuarioCompradorParticular("part1", "part1", "Guayaqui 3342",
                    "099868877", "Susaana", "Horia", "45867946");
62              crearUsuarioCompradorEmpresa("emp1", "emp1", "Cuareim 3322",
                  "47327844", "Ese COntacto", "Razon de Ser SRL", "111111");
63              //precargo datos del lugar de prueba
64              crearLugar("Casona del Lago", "Calle de Lugar", "564654",
                  "Montevideo");
65              crearLugar("Galpon", "Calle de Galpon", "78744", "Montevideo");
66              crearLugar("Hotel", "Calle del Hotel", "98745", "Montevideo");
67              crearLugar("Campo", "Calle del Campo", "25773", "Montevideo");
68              //precargo un remate
69              crearRemateGanado("remate1", 2, DateTime.Now, listaLugares[0],
                  listaRematadores[0]);
70              crearRemateGanado("remate2", 2, DateTime.Now, listaLugares[1],
                  listaRematadores[1]);
71              crearRemateGanado("remate3", 2, DateTime.Now, listaLugares[2],
                  listaRematadores[2]);
72              //asigno lotes a los remates
73              listaRemates[0].agregarLoteALista(listaLotes[0]);
74              listaRemates[0].agregarLoteALista(listaLotes[1]);
75             listaRemates[0].agregarLoteALista(listaLotes[2]);
76          }  //Objetos de Prueba
77
78          #region ALTAS DE OBJETOS
79          public void crearRematador(string unNombreUsuario, string unaClave,
              string unNombreRematador, string unApellidoRematador, string
              unTelefonoRematador)
80          {
81              Rematador unRematador = new Rematador(unNombreUsuario, unaClave,
                  unNombreRematador, unApellidoRematador, unTelefonoRematador);
82              listaRematadores.Add(unRematador);
83              listaUsuarios.Add(unRematador);
84          }
85
86          public void crearLugar(string unNombreLugar, string unaCalleLugar,
              string unNumeroPuertaLugar, string unaCiudadLugar)
87          {
88              Lugar unLugar = new Lugar(unNombreLugar, unaCalleLugar,
                  unNumeroPuertaLugar, unaCiudadLugar);
89              listaLugares.Add(unLugar);
90          }
91
92          public void crearLote(string unaDescripcionLote, double
              unPrecioBaseLote, int unaCantidadUnidadesLote, string unaFoto)
93          {
94              Lote unLote = new Lote(unaDescripcionLote, unPrecioBaseLote,
                  unaCantidadUnidadesLote, unaFoto);
95              listaLotes.Add(unLote);
96          }
97
98          public LoteComprado crearLoteComprado(Remate unRemate)
99          {
```

```
100             LoteComprado unLoteComprado = new LoteComprado(unRemate);
101             return unLoteComprado;
102         }
103
104         public void crearRemateGanado(string unaDescripcion, double
              unPorcentajeBaseRematador, DateTime unaFecha, Lugar unLugar,
              Rematador unRematador)
105         {
106             //Creo el remate y lo agrego a la lista
107             Ganado unRemateGanado = new Ganado(unaDescripcion,
                  unPorcentajeBaseRematador, unaFecha, unLugar, unRematador);
108             listaRemates.Add(unRemateGanado);
109         }
110
111         public void crearRemateMercaderia(string unaDescripcion, double
              unPorcentajeBaseRematador, DateTime unaFecha, Lugar unLugar,
              Rematador unRematador, double unPorcentajeAdicionalPorLote)
112         {
113             //Creo el remate y lo agrego a la lista
114             Mercaderia unRemateMercaderia = new Mercaderia(unaDescripcion,
                  unPorcentajeBaseRematador, unaFecha, unLugar, unRematador,
                  unPorcentajeAdicionalPorLote);
115             listaRemates.Add(unRemateMercaderia);
116         }
117
118         public void crearUsuarioAdministrador(string unNombreUsuario, string
              unaClave)
119         {
120             Administrador unAdministrador = new Administrador(unNombreUsuario,
                    unaClave);
121             listaUsuarios.Add(unAdministrador);
122         }
123
124         public void crearUsuarioCompradorParticular(string unNombreUsuario,
              string unaClave, string unaDireccion, string unTelefono, string
              unNombreParticular, string unApellidoParticular, string
              unDocumentoParticular)
125         {
126             Particular unParticular = new Particular(unNombreUsuario,
                  unaClave, unaDireccion, unTelefono, unNombreParticular,
                  unApellidoParticular, unDocumentoParticular);
127             listaUsuarios.Add(unParticular);
128             listaCompradores.Add(unParticular);
129         }
130
131         public void crearUsuarioCompradorEmpresa(string unNombreUsuario,
              string unaClave, string unaDireccion, string unTelefono, string
              unaPersonaContacto, string unaRazonSocial, string unRut)
132         {
133             Empresa unaEmpresa = new Empresa(unNombreUsuario, unaClave,
                  unaDireccion, unTelefono, unaPersonaContacto, unaRazonSocial,
                  unRut);
134             listaUsuarios.Add(unaEmpresa);
135             listaCompradores.Add(unaEmpresa);
136         }
137         #endregion
```

```
138
139         #region METODOS CON LISTAS
140         public List<Lugar> devolverListaLugares()
141         { //metodo que devuleve la lista de lugares
142             return listaLugares;
143         }
144
145         public List<Rematador> devolverListaRematadores()
146         { //metodo que devuelve la lista de rematadores
147             return listaRematadores;
148         }
149
150         public List<Rematador> devolverListaRematadoresActivosEnElAño()
151         { //metodo que devuelve la lista de rematadores que participaron de
              uno o mas remates en el año
152             List<Rematador> listaRematadoresActivosEnElAño = new
                  List<Rematador>();
153             foreach (Remate R in listaRemates)
154             {
155                 double comTmp;
156                 if (R.Cerrado ==true && R.Fecha.Year == DateTime.Today.Year)
157                 {
158                     if (listaRematadoresActivosEnElAño.Count != 0)
159                     {
160                         foreach (Rematador tmp in
                        listaRematadoresActivosEnElAño)
161                         {
162                             foreach (Rematador tmp2 in
                            listaRematadoresActivosEnElAño)
163                             {
164                                 if (tmp.IdRematador == tmp2.IdRematador)//si
                                ya existe solo lke agrego comision
165                                 {
166                                     comTmp = Convert.ToDouble
                                    (R.Rematador.ComisionRematador);//obtengo el valor que ya
                                    tenia convirtiendolo
167                                     R.Rematador.ComisionRematador = ((comTmp +
                                     devolverComisionRematador(R.IdRemate)).ToString());//le
                                    agrego el nuevo y lo substituyo
168                                 }
169                                 else
170                                 {// si termine de recorrer y no lo encontre,
                                    lo creo como objeto
171                                     R.Rematador.ComisionRematador = ((comTmp =
                                     devolverComisionRematador(R.IdRemate)).ToString());
172                                     listaRematadoresActivosEnElAño.Add
                                    (R.Rematador);
173                                 }
174                             }
175                         }
176                     }
177                     else // lo creo porque la lista esta vacia, solo entra aca
                         una y solo una vez
178                     {
179                         R.Rematador.ComisionRematador = ((comTmp =
                        devolverComisionRematador(R.IdRemate)).ToString());
```

```
180                         listaRematadoresActivosEnElAño.Add(R.Rematador);
181                     }
182                 }
183             }
184             return listaRematadoresActivosEnElAño;
185         }
186
187         public List<Lote> devolverListaLotesDisponibles()
188         {
189             List<Lote> listaLotesDiponibles = new List<Lote>();
190             foreach (Lote tmpLote in listaLotes)
191             {
192                 if (tmpLote.FueAsignado == false)
193                 {
194                     listaLotesDiponibles.Add(tmpLote);
195                 }
196             }
197             return listaLotesDiponibles;
198         } //metodo que devuleve la lista de lotes disponibles para ser
              asignados a un nuevo remate
199
200         public List<Remate> devolverListaRemates()
201         {
202             return listaRemates;
203
204         } //metodo qeu develve la lista de remates
205
206         public List<Remate> devolverListaRematesCerradosPorRematador(int
              unIdRem)
207         {
208             List<Remate> listaRematesCerradosRematador = new List<Remate>();
209             foreach (Remate tmpRemate in listaRemates)
210             {
211                 if (tmpRemate.Cerrado == true &&
                      tmpRemate.Rematador.IdRematador == unIdRem)
212                 {
213                     listaRematesCerradosRematador.Add(tmpRemate);
214                 }
215             }
216             return listaRematesCerradosRematador;
217
218         } //metodo qeu develve la lista de remates cerrados de un rematador
219
220         public List<Remate> devolverRematesPorFecha(DateTime unaFechaDesde,
              DateTime unaFechaHasta)
221         {
222             List<Remate> listaRematesPorFecha = new List<Remate>();
223             foreach (Remate tmpRemate in listaRemates)
224             {
225                 if (tmpRemate.Fecha >= unaFechaDesde && tmpRemate.Fecha <=
                      unaFechaHasta)
226                 {
227                     listaRematesPorFecha.Add(tmpRemate);
228                 }
229             }
230             return listaRematesPorFecha;
```

```
231         } //metodo que devuleve la lista de remates disponibles validando un
              intervalo de fechas
232
233         public List<Remate> devolverRematesDisponibles()
234         {
235             List<Remate> listaRematesDisponibles = new List<Remate>();
236             foreach (Remate tmpRemate in listaRemates)
237             {
238                 if (tmpRemate.Cerrado == false)
239                 {
240                     listaRematesDisponibles.Add(tmpRemate);
241                 }
242             }
243             return listaRematesDisponibles;
244         } //metodo que devuleve la lista de remates disponibles para utilizar
              en asignar lotes a remate.
245
246         public void darBajaRematador(Rematador unRematador)
247         {
248             listaRematadores.Remove(unRematador);
249         }//metodo que elimina un objeto rematador de la lista de rematadores
              validando que no tenga remates en el futuro (metodo
              'tieneRematesAsignados')
250
251         public List<Usuario> devolverListaUsuarios()
252         {
253             return listaUsuarios;
254
255         } //metodo que develve la lista de usuarios
256
257         public List<Comprador> devolverListaCompradores()
258         {
259             return listaCompradores;
260
261         } //metodo que develve la lista de compradores
262
263         public List<Comprador> devolverListaCompradoresQueParticiparonDeRemate
              (int unIdRemate)//metodo que recibe un idRemate como parametro y
              retorna una lista de compradores que participaron del mismo
264         {
265             List<Comprador> listaCompradoresQueParticiparonDeRemate = new
                  List<Comprador>();
266
267             foreach(Comprador C in listaCompradores)
268             {
269                 if (C.participoDeRemate(unIdRemate))
270                 {
271                     listaCompradoresQueParticiparonDeRemate.Add(C);
272                 }
273             }
274
275             return listaCompradoresQueParticiparonDeRemate;
276
277         } //metodo qeu develve la lista de compradores
278
279         #endregion
```

```
280
281         #region VALIDACIONES
282         public bool validoLugarFecha(DateTime unaFecha, int unIdLugar)
283         { //metodo que valida la disponibilidad de la combinacion fecha/lugar
284             bool retorno = true;
285             foreach (Remate r in listaRemates)
286             {
287                 if (r.Lugar.IdLugar == unIdLugar && r.Fecha == unaFecha)
288                 {
289                     retorno = false;
290                 }
291             }
292             return retorno;
293         }
294
295         public int validarUsuario(string unNombre, string unaClave)
296         {
297             Usuario u = new Usuario();
298             int retorno = 0;
299             u.NombreUsuario = unNombre;
300             u.Clave = unaClave;
301             int indice = listaUsuarios.IndexOf(u);
302             if (indice != -1)
303             {
304
305                 u = listaUsuarios[indice]; //traigo todo el objeto usuario
306                 if (u.Clave.Equals(unaClave))
307                 {
308                     retorno = u.devolverTipo();
309                 }
310             }
311             return retorno;
312         }//metodo que valida usuario y devuelve su tipo
313
314         public bool tieneRematesAsignados(int unIdRematador)
315         {
316             bool retonro = false;
317             bool loEncontre = false;
318             foreach (Remate r in listaRemates)
319             {
320                 if (r.Cerrado == false && r.Rematador.IdRematador ==
                      unIdRematador && !loEncontre)
321                 {
322                     retonro = true;
323                     loEncontre = true;
324                 }
325             }
326             return retonro;
327         }//valida que el rematador a ser dado de baja no tenga remates a
            relaizar em el furuto
328
329         public bool nombreusuarioYaExiste(string unNombreUsuario){
330             bool loEncontre = false;
331             int pos = 0;
332             List<Usuario> listaUsuarios = devolverListaUsuarios();
333             while (pos < listaUsuarios.Count && !loEncontre)
```

```
334             {
335                 // Recorro hasta que lo encuentre
336                 if (listaUsuarios[pos].NombreUsuario == unNombreUsuario)
337                 {
338                     loEncontre = true;
339                 }
340                 pos++;
341             }
342             return loEncontre;
343         }//metodo que valida que el nombre de usuario ya no haya sido usado
344 
345         public bool stringEsSoloNumeros(string unString) {
346             foreach (char c in unString){
347                 if (c < '0' || c > '9') {
348                     return false;
349                 }
350             }
351             return true;
352         }
353         #endregion
354 
355         #region METODOS DE BUSQUEDA
356         public Lugar buscarLugar(int unIdLugar)
357         {
358             Lugar retorno = null;
359             bool loEncontre = false;
360             int pos = 0;
361             List<Lugar> listaDisponibles = devolverListaLugares();
362             while (pos < listaDisponibles.Count && !loEncontre)
363             {
364                 // Recorro hasta que lo encuentre
365                 if (listaDisponibles[pos].IdLugar == unIdLugar)
366                 {
367                     retorno = listaDisponibles[pos];
368                     loEncontre = true;
369                 }
370                 pos++;
371             }
372             return retorno;
373         }
374 
375         public Rematador buscarRematador(int unIdRematador)
376         {
377             Rematador retorno = null;
378             bool loEncontre = false;
379             int pos = 0;
380             while (pos < listaRematadores.Count && !loEncontre)
381             {
382                 // Recorro hasta que lo encuentre
383                 if (listaRematadores[pos].IdRematador == unIdRematador)
384                 {
385                     retorno = listaRematadores[pos];
386                     loEncontre = true;
387                 }
388                 pos++;
389             }
```

```
390             return retorno;
391         } //método que devuelve un objeto rematador, recibiendo su id como
              parámetro
392
393         public Comprador buscarComprador(string unNomUsuarioComprador){
394             Comprador retorno = null;
395             bool loEncontre = false;
396             int pos = 0;
397             while (pos < listaCompradores.Count && !loEncontre)
398             {
399                 // Recorro hasta que lo encuentre
400                 if (listaCompradores[pos].NombreUsuario ==
                      unNomUsuarioComprador)
401                 {
402                     retorno = listaCompradores[pos];
403                     loEncontre = true;
404                 }
405                 pos++;
406             }
407             return retorno;
408         } //método que devuelve un objeto comprador, recibiendo su nombre de
              usuario como parámetro
409
410         public Rematador buscarRematadorPorNombreUsuario(string
              unNomUsuarioRematador)
411         {
412             Rematador retorno = null;
413             bool loEncontre = false;
414             int pos = 0;
415             while (pos < listaRematadores.Count && !loEncontre)
416             {
417                 // Recorro hasta que lo encuentre
418                 if (listaRematadores[pos].NombreUsuario ==
                      unNomUsuarioRematador)
419                 {
420                     retorno = listaRematadores[pos];
421                     loEncontre = true;
422                 }
423                 pos++;
424             }
425             return retorno;
426         } //método que devuelve un objeto rematador, recibiendo su nombre de
              usuario como parámetro
427
428         public Lote buscarLoteDisponible(string unIdLote)
429         {
430             Lote retorno = null;
431             bool loEncontre = false;
432             int pos = 0;
433             List<Lote> listaDisponibles = devolverListaLotesDisponibles();
434             while (pos < listaDisponibles.Count && !loEncontre)
435             {
436                 // Recorro hasta que lo encuentre
437                 if (listaDisponibles[pos].IdLote == unIdLote)
438                 {
439                     retorno = listaDisponibles[pos];
```

```
440                     loEncontre = true;
441                 }
442                 pos++;
443             }
444             return retorno;
445         } //método que devuelve un objeto lote DISPONIBLE, recibiendo su id
              como parámetro
446
447         public Lote buscarLote (string unIdLote)
448         {
449             Lote retorno = null;
450             bool loEncontre = false;
451             int pos = 0;
452             while (pos < listaLotes.Count && !loEncontre)
453             {
454                 // Recorro hasta que lo encuentre
455                 if (listaLotes[pos].IdLote == unIdLote)
456                 {
457                     retorno = listaLotes[pos];
458                     loEncontre = true;
459                 }
460                 pos++;
461             }
462             return retorno;
463         } //método que devuelve un objeto lote, recibiendo su id como
              parámetro
464
465         public Remate buscarRemate(int unIdRemate)
466         {
467             Remate retorno = null;
468             bool loEncontre = false;
469             int pos = 0;
470             while (pos < listaRemates.Count && !loEncontre)
471             {
472                 // Recorro hasta que lo encuentre
473                 if (listaRemates[pos].IdRemate == unIdRemate)
474                 {
475                     retorno = listaRemates[pos];
476                     loEncontre = true;
477                 }
478                 pos++;
479             }
480             return retorno;
481         } //método que devuelve un objeto remate, recibiendo su id como
482         #endregion
483
484         public void modificarDatosRematador(int unIdRematador, string
              unNombre, string unApellido, string unTelefono){
485             Rematador unRematador = buscarRematador(unIdRematador);
486             unRematador.NombreRematador = unNombre;
487             unRematador.ApellidoRematador = unApellido;
488             unRematador.TelefonoRematador = unTelefono;
489         }
490
491         public void cerrarRemate(int unIdRemate)
492         {
```

```
493             Remate unRemate = buscarRemate(unIdRemate);
494             unRemate.Cerrado = true;
495             unRemate.liberarLotesNoVendidos();
496         }
497
498         public double devolverComisionRematador(int unIdRemate)
499         {
500             Remate unRemate = buscarRemate(unIdRemate);
501             double comisionRematador =
                  unRemate.calcularComisionAPagarRematador();
502             return comisionRematador;
503         }
504         #endregion
505     }
506 }
507
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace Dominio
 8  {
 9      public abstract class Remate
10      {
11          private List<Lote> listaLotesDeRemate;
12
13          #region Atributos y Properties
14          //Atributos de Clase
15          // Autonumerado
16          private static int _UltimoNro = 0;
17
18          public static int UltimoNumero
19          {
20              get { return _UltimoNro; }
21          }
22
23          // Atributos de Instancia
24
25          private int _idRemate;
26          public int IdRemate
27          {
28              get { return _idRemate; }
29              set { _idRemate = value; }
30          }
31
32          private string _descripcion;
33          public string Descripcion
34          {
35              get { return _descripcion; }
36              set { _descripcion = value; }
37          }
38
39          private double _porcentajeBaseRematador;
40          public double PorcentajeBaseRematador
41          {
42              get { return _porcentajeBaseRematador; }
43              set { _porcentajeBaseRematador = value; }
44          }
45
46          private DateTime _fecha;
47          public DateTime Fecha
48          {
49              get { return _fecha; }
50              set { _fecha = value; }
51          }
52
53          private Lugar _lugar;
54          public Lugar Lugar
55          {
56              get { return _lugar; }
```

```
57              set { _lugar = value; }
58          }
59
60          private Rematador _rematador;
61          public Rematador Rematador
62          {
63              get { return _rematador; }
64              set { _rematador = value; }
65          }
66
67          private bool _cerrado;
68          public bool Cerrado
69          {
70              get { return _cerrado; }
71              set { _cerrado = value; }
72          }
73
74          private double _comisionRematador;
75          public double ComisionRematador
76          {
77              get { return _comisionRematador; }
78              set { _comisionRematador = value; }
79          }
80
81          #endregion
82
83          #region Constructores
84          public Remate(string unaDescripcion, double unPorcentajeBaseRematador,
               DateTime unaFecha, Lugar unLugar, Rematador unRematador)
85          {
86              _UltimoNro++;
87              listaLotesDeRemate = new List<Lote>();
88              this.IdRemate = Remate._UltimoNro;
89              this.Descripcion = unaDescripcion;
90              this.PorcentajeBaseRematador = unPorcentajeBaseRematador;
91              this.Fecha = unaFecha;
92              this.Lugar = unLugar;
93              this.Rematador = unRematador;
94              this.Cerrado = false;
95          }
96          #endregion
97
98          #region Metodos sobreescritos / heredados
99          public override string ToString()
100         {
101             return "Fecha: " + this.Fecha.ToString() + "\n" + "Lugar: " +
                  this.Lugar.NombreLugar + "\n" + "Rematador: " +
                  this.Rematador.NombreRematador + "\n" + "\n";
102         }
103
104         // Si ya se que este metodo es polimórfico y tiene codigo debo
              indicarlo agregando la palabra virtual
105         public abstract double calcularComisionAPagarRematador();
106
107         public virtual double calcularMontoVentasRemate()
108         {
```

```
109             double montoVentasRemate = 0;
110             List<Lote> listaLotesRemate = devolverListaLotesRemate();
111             foreach(Lote L in listaLotesDeRemate){
112                 if (L.FueComprado == true){
113                     montoVentasRemate += L.PrecioCompraLote;
114                 }
115             }
116             return montoVentasRemate;
117         }//devuelve el monto total cobrado por los lotes vendidos
118         #endregion
119 
120         #region Otros metodos
121         public void agregarLoteALista(Lote unLote)
122         {
123             unLote.FueAsignado = true;//cambio el estado del lote
124             listaLotesDeRemate.Add(unLote);
125         }//metodo que agrega un objeto lote a la lista de lotes del remate
126 
127         public List<Lote> devolverListaLotesRemate()
128         {
129             return listaLotesDeRemate;
130         }
131 
132         public List<Lote> devolverListaLotesNoVendidosRemate()
133         {
134             List<Lote> listaLotesNoVendidos = new List<Lote>();
135             foreach(Lote L in listaLotesDeRemate){
136                 if(L.FueComprado == false){
137                     listaLotesNoVendidos.Add(L);
138                 }
139             }
140             return listaLotesNoVendidos;
141         }
142 
143         public void liberarLotesNoVendidos()
144         {
145             foreach(Lote L in listaLotesDeRemate)
146             {
147                 if (L.FueComprado == false)
148                 {
149                     L.FueAsignado = false;
150                 }
151             }
152         }
153 
154         public int devolverCantidadLotesVendidos()
155         {
156             int cantidadLotesVendidos = 0;
157             List<Lote> listaLotesRemate = devolverListaLotesRemate();
158             foreach (Lote L in listaLotesRemate)
159             {
160                 if (L.FueComprado == true)
161                 {
162                     cantidadLotesVendidos++;
163                 }
164             }
```

```
165             return cantidadLotesVendidos;
166         }//devuelve la cantidad de lotes que fueron vendidos para calcular la ↵
              comision adicional
167
168         #endregion
169     }
170 }
171
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Dominio
8  {
9      public class Usuario
10     {
11         #region Atributos y Properties
12         // Atributos de Instancia
13         public string NombreUsuario { set; get; }
14         public string Clave { set; get; }
15         #endregion
16
17         #region Constructores
18         public Usuario(string unNombreUsuario, string unaClave)
19         {
20             this.NombreUsuario = unNombreUsuario;
21             this.Clave = unaClave;
22         }
23
24         public Usuario()
25         {
26         }
27
28         #endregion
29
30         #region Otros Metodos
31         public virtual int devolverTipo() {
32            return 0 ;
33                 }
34         #endregion
35
36         #region Metodos Sobreescrios
37         public override bool Equals(object obj)
38         {
39             Usuario otro = obj as Usuario;
40             return this.NombreUsuario.Equals(otro.NombreUsuario);
41         }
42
43         public int CompareTo(Usuario other)
44         {
45             int res;
46             if (this.NombreUsuario.CompareTo(other.NombreUsuario) < 0) // alias ↵
                  de this esta antes qu el alias de other
47             {
48                 res = -1;
49             }
50             else
51             {
52                 if (this.NombreUsuario.CompareTo(other.NombreUsuario) > 0)
53                 {
54                     res = 1;
55                 }
```

```
56                  else
57                  {
58                      res = 0;
59                  }
60              }
61
62              return res;
63          }
64          #endregion
65      }
66  }
67
```