TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN UTN-FRC

TUP

UTN

Facultad Regional Córdoba

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

PROGRAMACIÓN I

Unidad Temática 4: Herencia y Polimorfismo

Material de Estudio

1er Año - 1er Cuatrimestre

2019

V.0.1

Índice

Herencia y Polimorfismo

Herencia

En la programación orientada a objetos, la herencia permite que una clase herede las características y el comportamiento de otra clase (sus métodos y atributos) y a continuación, modificar este comportamiento según sean las necesidades.

Esto permite tener varias clases con un comportamiento similar en ciertas situaciones (por ejemplo, al llamar a ciertos métodos) y un comportamiento específico en otras.

Las distintas clases de un programa se organizan mediante la jerarquía de clases.

Para heredar de una clase se ha de incluir la palabra reservada extends seguida de la clase de la que se quiere heredar.

En Java solamente se puede heredar desde una clase, al contrario que en otros lenguajes de programación, como C++, donde existe la herencia múltiple, en la que se puede heredar de varias clases.

La forma que suele presentar una clase que hereda de otra es la siguiente:

Modo Acceso class NombreClase extends ClasePadre...

En el esquema que se muestra a continuación, se plantean dos entidades una Alumno y otra Profesor. Si aplicamos el concepto de Herencia, surgen otras tres entidades:

La implementación en C# queda de la siguiente forma:

```
//Clase Base o Madre
public class Persona
{
        //Atributos
        //Métodos
}
//Declaración de las clases Hijas o Derivadas de Persona.
public class Alumno : Persona
{
        //Atributos
        //Métodos
}
public class Profesor : Persona
{
        //Atributos
        //Métodos
}
```

Cuando una clase hereda de otra, la clase hija hereda los atributos y los métodos de la clase padre, pero existen ciertas restricciones en esta herencia:

Los atributos y los métodos con modo de acceso private no se heredan. Se heredan los atributos y los métodos con modo de acceso public y protected. No se hereda un atributo de una clase padre si en la clase hija se define un atributo con el mismo nombre que el atributo de la clase padre. No se hereda un método si éste es sobrecargado. No se heredan los constructores de la clase base o madre. Por lo tanto en el constructor de la clase derivada, en la primera línea del mismo, hay que invocar al constructor de la clase base con la palabra super y la cantidad de parámetros que requiera el constructor, esto es así porque primero se debe crear la clase base y por último la clase derivada.

Para referenciar a los miembros de la clase base se utiliza la palabra reservada super, generalmente ésta se usa si los miembros de la clase base se llaman de la misma forma que los de la clase derivada, si no, no hace falta usar la palabra super.

Sobrecarga y ocultación

Al crear una clase que hereda de otra se pueden definir nuevos atributos y métodos o se puede modificar el comportamiento de los objetos de la clase sobrecargando los métodos de la clase padre. La sobrecarga se refiere a la posibilidad de proporcionar nuevos métodos con el mismo nombre que tienen los métodos de la clase padre, pero con diferente comportamiento. Esto permite que el entorno y comportamiento de una clase sea heredado por otra y solamente se modifiquen aquel los métodos que sean necesarios para su comportamiento específico.

El operador instanceof

Se puede consultar si un objeto pertenece a una clase mediante:

```
Object obj;
...
if (obj instanceof A)
// obj pertenece a la clase A
A a=(A)obj;    // Ok, nunca hay error
```

Los objetos de la clase B también son instancias de la clase A:

```
Object obj= new B();
if (obj instanceof A) // true
A a= (A)obj;   // Ok
```

Polimorfismo

El polimorfismo indica que una variable puede adoptar múltiples formas. Cuando se habla de polimorfismo en programación orientada a objetos se suelen entender dos cosas:

La primera suele referirse a que se puede trabajar con un objeto de una clase sin conocer ni importar de qué clase se trata. Es decir, se trabajará igual sea cual sea la clase a la que pertenece el objeto. Esto normalmente se consigue mediante jerarquías de clases o interfaces.

La segunda se suele referir a la posibilidad de declarar métodos con el mismo nombre que pueden tener diferentes argumentos dentro de una misma clase. Un ejemplo de la primera podría ser la utilización de un método como toStringO, que tienen todos los objetos al heredar de la clase Object.

Resumiendo: La facultad de llamar a una variedad de métodos utilizando el mismo medio de acceso, proporcionados por los métodos redefinidos en las clases hijas, es denomida polimorfismo. La palabra “polimorfismo” significa “la facultad de asumir muchas formas” refiriéndose a la facultad de llamar a muchos métodos diferentes utilizando una única sentencia.

Cuando se invoca a una método que está definido en la clase madre o base y redefinido en las clases derivadas o hijas, la versión que se ejecuta del método depende de la clase del objeto referenciado, no de la variable que lo referencia.

Permite mejorar la implementación de los programas. Por ejemplo, la implementación del ejemplo anterior sin usar polimorfismo, implicaría crear la cantidad de objetos que podría necesitar el usuario, si bien el mismo solo va a usar uno por vez, tiene que tenerlos a todos en la memoria para que cuando requiera el listado correspondiente, tenga acceso a cualquier salida. En cambio si se utiliza polimorfismo, en memoria sólo se requerirá en forma permanente una referencia al dispositivo (genérico) y cuando el usuario decida imprimir el listado, elegirá en que dispositivo lo hará (monitor, impresora o terminal remota), por lo tanto en tiempo de ejecución se decide cual es el objeto que estará activo en el momento de la impresión, dejando en memoria solo el objeto que se requiera y no los n posibles.

Polimorfismo en C#

En C# hay ciertos conceptos que se van a usar para aplicar polimorfismo, ellos son:

Clases abstractas: Son clases genéricas, sirven para agrupar clases del mismo género, no se pueden instanciar, es decir no se pueden crear objetos a partir de ellas, ya que representan conceptos tan generales que no se puede definir como será la implementación de la misma. Es por eso que dichas clases llevan métodos que se llaman abstractos, que no tienen implementación.

Sintaxis:

```
Acceso abstract class NombreClase
{...}
```

Por ejemplo:

```
public abstract class Dispositivo
{ ..... }
```

Una clase abstracta debe tener al menos un método abstracto.

Métodos abstractos: Métodos que no tienen implementación, se utilizan para recibir los mensajes en común. No tiene implementación porque en el nivel donde se definen no hay información suficiente para implementarlos. En los próximos niveles de la jerarquía se pueden implementar, para ello deben ser redefinidos en niveles posteriores.

Sintaxis:

```
Acceso abstract tipo NombreMetodo (parametros);
```

Por ejemplo:

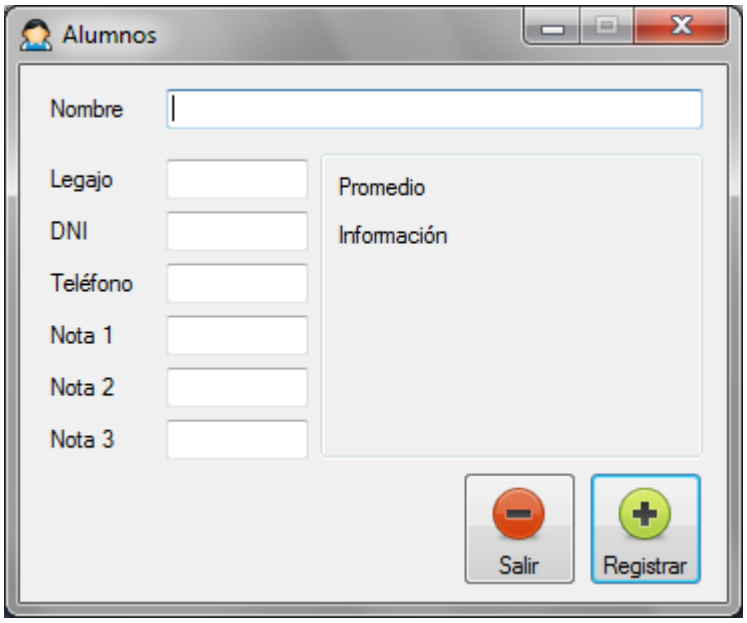
```
public abstract void Imprimir();
```

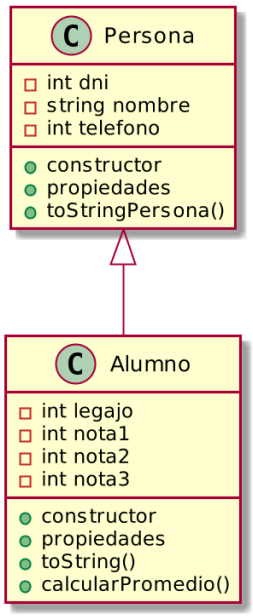
Problemas modelo

Problema 1.1: Alumno:

Dados el nombre, legajo, DNI, teléfono y tres notas de alumnos, calcular su promedio y mostrar sus datos.

Solución






```
/*****************************************************************/
/* Persona.cs */
 /*****************************************************************/
namespace ProgrI_Ej012
{
    abstract class Persona
    {
        int dni;
        string nombre;
        int telefono;
        public Persona()
        {
            dni = 0;
            nombre = "";
            telefono = 0;
        }
        public Persona(int doc, string nomb, int tel)
        {
            dni = doc;
            nombre = nomb;
            telefono = tel;
```

```
            }
            public int pDni
            {
                set { dni = value; }
                get { return dni; }
            }
            public string pNombre
            {
                set { nombre = value; }
                get { return nombre; }
            }
            public int pTelefono
            {
                set { telefono = value; }
                get { return telefono; }
            }
            public string toStringPersona()
            {
                return"Nombre: " + nombre + "\nDNI: " + dni.ToString()
                    + "\nTeléfono: " + telefono.ToString();
            }
        }
    }

    /*********************************************************************/
    /* Alumno.cs */
    /*********************************************************************/
    namespace ProgrI_Ej012
{
    class Alumno: Persona
    {
        int legajo;
        double nota1, nota2, nota3;
        public Alumno(): base()
        {
            legajo = 0;
            nota1 = nota2 = nota3 = 0;
        }
        public Alumno(int leg, double n1, double n2, double n3, int d, string n, int t): base(d, n, t)
        {
            legajo = leg;
            nota1 = n1;
            nota2 = n2;
            nota3 = n3;
        }
        public int pLegajo
        {
            set { legajo = value; }
            get { return legajo; }
        }
        public double pNota1
        {
            set { nota1 = value; }
            get { return nota1; }
        }
        public double pNota2
        {
            set { nota2 = value; }
            get { return nota2; }
        }
        public double pNota3
        {
            set { nota3 = value; }
            get { return nota3; }
        }
        public string toString()
        {
            return base.toStringPersona() + "\nLegajo: " + legajo.ToString()
                + "\nNota 1: "+ nota1.ToString() + "\nNota 2: " + nota2.ToString()
```

```
                + "\nNota 3: "+ nota3.ToString();
        }
        public double calcularPromedio()
        {
            return (nota1 + nota2 + nota3) / 3;
        }
    }
}
```

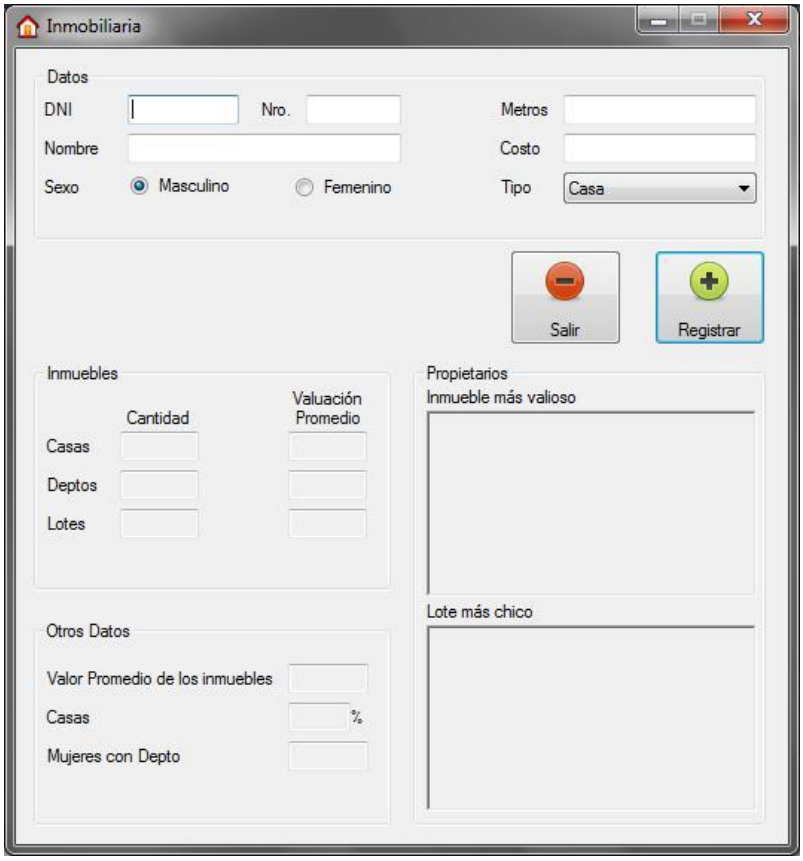
```
/*************************************************************/
/* Form1.cs */
/*************************************************************/
namespace ProgrI_Ej012
{
    public partial class frmAlumnos : Form
    {
        public frmAlumnos()
        {
            InitializeComponent();
        }
        private void btnRegistrar_Click(object sender, EventArgs e)
        {
            if (validarCampos() && validarNotas())
            {
                Alumno alumn = new Alumno();
                alumn.pNombre = txtNombre.Text;
                alumn.pLegajo = Convert.ToInt32(txtLegajo.Text);
                alumn.pTelefono = Convert.ToInt32(txtTelefono.Text);
                alumn.pDni = Convert.ToInt32(txtDni.Text);
                alumn.pNota1 = Convert.ToInt32(txtNota1.Text);
                alumn.pNota2 = Convert.ToInt32(txtNota2.Text);
                alumn.pNota3 = Convert.ToInt32(txtNota3.Text);
                lblPromedio.Text = "Promedio: "
                    + Math.Round(alumn.calcularPromedio(), 2).ToString();
                    + lblTexto.Text = alumn.toString();
                limpiarCampos();
            }
        }
        private void btnSalir_Click(object sender, EventArgs e)
        {
            DialogResult respuesta = MessageBox.Show("¿Está seguro que desea salir?","Salir",
                MessageBoxButtons.YesNo,                    MessageBoxIcon.Question);
            if(respuesta==DialogResult.Yes)
                Close();
        }
        private bool validarCampos()
        {
            foreach (Control c in this.Controls)
            {
                if(c isTextBox&&string.IsNullOrEmpty(c.Text))
                {
                    MessageBox.Show("Por favor, complete con los datos faltantes","Faltan
                     datos",
                        MessageBoxButtons.OK, MessageBoxIcon.Error); c.Focus();
                    return false;
               }
               return true;
        }
        private bool validarNotas()
        {
            if (Convert.ToInt32(txtNota1.Text) < 0 ||
                Convert.ToInt32(txtNota1.Text) > 10)
            {
                MessageBox.Show("La      nota      ingresada      no      es      válida",
                    "Error",MessageBoxButtons.OK, MessageBoxIcon.Error);
                txtNota1.Text = "";
```

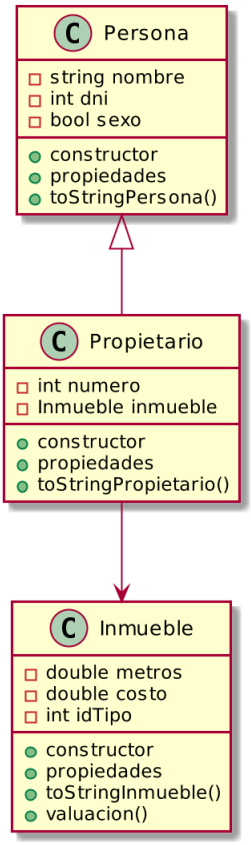
```
                txtNota1.Focus();
                return false;
            }
            if (Convert.ToInt32(txtNota2.Text) < 0 ||
                Convert.ToInt32(txtNota2.Text) > 10)
            {
                MessageBox.Show("La nota ingresada no es válida", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                txtNota2.Text = "";
                txtNota2.Focus();
                return false;
            }
            if (Convert.ToInt32(txtNota3.Text) < 0 ||
                Convert.ToInt32(txtNota3.Text) > 10)
            {
                MessageBox.Show("La nota ingresada no es válida", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                txtNota3.Text = "";
                txtNota3.Focus();
                return false;
            }
            return true;
        }
        private void limpiarCampos()
        {
            foreach (Control c in this.Controls)
            {
                if (c isTextBox)
                {
                    c.Text = "";
                }
            }
            txtNombre.Focus();
        }
    }
}
```

Problema 1.2: Inmobiliaria:

Dados los datos de un propietario (tipo y número de DNI, nombre y sexo) y de un inmueble (metros, costo por metro y tipo de inmueble), determinar la cantidad de casas, departamentos y lotes, su valuación promedio, el valor promedio de los inmuebles, el porcentaje de casas, la cantidad de mujeres con departamento, el propietario con inmueble más valioso y el propietario con lote más chico.

Solución






```
/*********************************************************************/
/* Persona.cs */
/*********************************************************************/
namespace ProgrI_Ej016
{
    abstract class Persona
    {
        //Atributos:
        int dni;
        string nombre;
        bool sexo;
        //Constructores:
        public Persona()
        {
            dni = 0;
            nombre = "";
            sexo = false;
        }
        public Persona(int doc, string nomb, bool sex)
        {
            dni = doc;
            nombre = nomb;
            sexo = sex;
        }
        //Propiedades:
        public int pDni
        {
            set { dni = value; }
            get { return dni; }
        }
        public string pNombre
         {
             set { nombre = value; }
             get { return nombre; }
         }
         public bool pSexo
```

```
            {
                set { sexo = value; }
                get { return sexo; }
            }
            //Métodos:
            public string toStringPersona()
            {
                string sex;
                if (sexo)
                    sex = "Masculino";
                else
                    sex = "Femenino";
                return"DNI: " + dni.ToString() + "\nNombre: " + nombre + "\nSexo: " + ,! sex;
            }
        }
    }

    /*********************************************************************/
    /* Propietario.cs */
    /*********************************************************************/
    namespace ProgrI_Ej016
    {
        class Propietario:Persona
        {
            //Atributos:
            int numero;
            Inmueble inmueble;
            //Constructores:
            public Propietario()
                : base()
            {
                numero = 0;
                inmueble = null;
            }
            public Propietario(int nro, Inmueble i, int d, string n, bool s)
                : base(d, n, s)
            {
                numero = nro;
                inmueble = i;
            }
            //Propiedades:
            public int pNumero
            {
                set { numero = value; }
                get { return numero; }
             }
           public Inmueble pInmueble
           {
                set { inmueble = value; }
                get { return inmueble; }
           }
           //Métodos:
           public string toStringPropietario()
           {
                return base.toStringPersona() + "\nNúmero: " + numero + "\n"
                    + inmueble.toStringInmueble();
           }
       }
    }

    /*********************************************************************/
    /* Inmueble.cs */
    /*********************************************************************/
    namespace ProgrI_Ej016
    {
        class Inmueble
        {
            //Atributos:
            double metros;
```

```
        double costo;
        int idTipo;
        //Constructores:
        public Inmueble()
        {
            metros = 0;
            costo = 0;
            idTipo = 0;
        }
        public Inmueble(double m, double c, int t)
        {
            metros = m;
            costo = c;
            idTipo = t;
        }
        //Propiedades:
        public double pMetros
        {
            set { metros = value; }
            get { return metros; }
        }
        public double pCosto
        {
            set { costo = value; }
            get { return costo; }
        }
        public int pIdTipo
          {
              set { idTipo = value; }
              get { return idTipo; }
          }
          //Métodos:
          public string toStringInmueble()
          {
              string tipo;
              switch (idTipo)
              {
                  case 1: tipo = "Casa"; break;
                  case 2: tipo = "Depto"; break;
                  case 3: tipo = "Lote"; break;
                  default: tipo = "Otro"; break;
              }
                return"\nMetros: " + metros.ToString() + "\nCosto: " + costo.ToString()
                  + "\nTipo: " + tipo;
          }
          public double valuacion()
          {
              return costo * metros;
          }
    }
}

/*********************************************************************/
/* Form1.cs */
/*********************************************************************/
namespace ProgrI_Ej016
{
    public partial class frmInmobiliaria : Form
    {
        int contCasas, contDeptos, contLotes;
        int contMujDepto;
        double acCasas, acDeptos, acLotes;
        double valorPromedio;
        double porcentajeCasas;
        Propietario propMayor, propChico;
        bool banderaMayorValor, banderaMasChico;
        public frmInmobiliaria()
        {
            InitializeComponent();
```

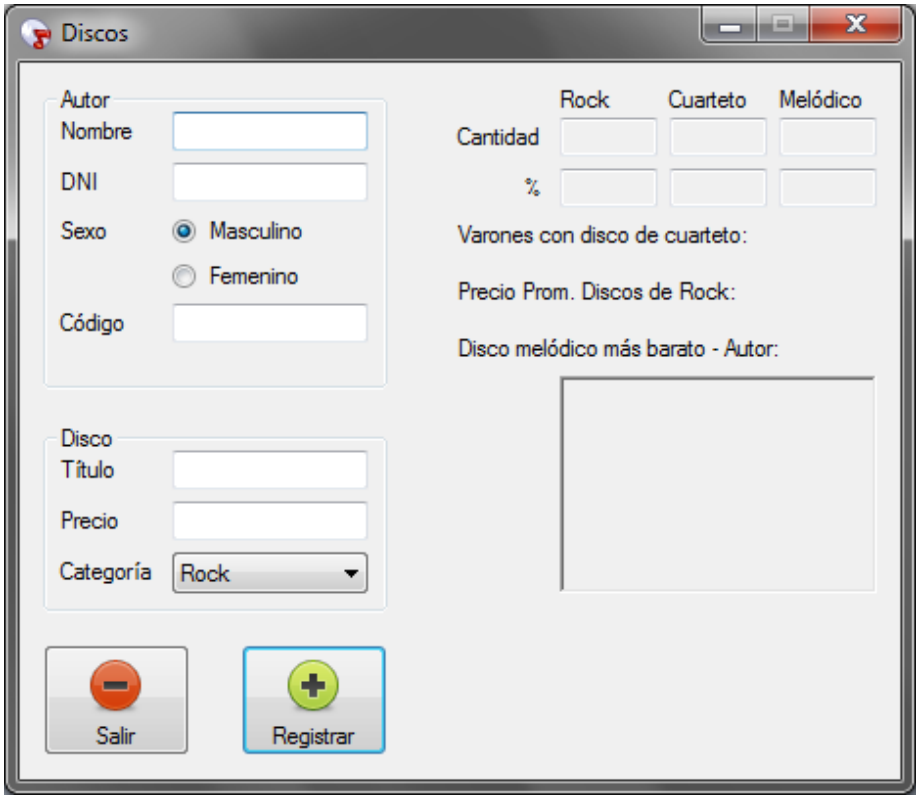
```
        contCasas = 0;
        contDeptos = 0;
        contLotes = 0;
        contMujDepto = 0;
        acCasas = 0;
        acDeptos = 0;
        acLotes = 0;
        valorPromedio = 0;
        porcentajeCasas = 0;
        propMayor = null;
        propChico = null;
        banderaMayorValor = true;
        banderaMasChico = true;
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        cmbTipo.SelectedIndex = 0;
        rbMasculino.Checked = true;
    }
    private void btnRegistrar_Click(object sender, EventArgs e)
    {
        Propietario P = new Propietario();
        Inmueble I = new Inmueble();
        P.pDni = Convert.ToInt32(txtDni.Text);
        P.pNombre = txtNombre.Text;
        P.pSexo = rbMasculino.Checked;
        P.pNumero = Convert.ToInt32(txtNumero.Text);
        I.pMetros  =  Convert.ToDouble(txtMetros.Text);  I.pCosto  =
        Convert.ToDouble(txtCosto.Text);         I.pIdTipo         =
        cmbTipo.SelectedIndex + 1;
        P.pInmueble = I;
        switch (P.pInmueble.pIdTipo)
        {
            case 1:
                contCasas++;
                acCasas += P.pInmueble.valuacion();
                break;
            case 2:
                contDeptos++;
                acDeptos += P.pInmueble.valuacion();
                if (!P.pSexo)
                    contMujDepto++;
                break;
            case 3:
                contLotes++;
                acLotes += P.pInmueble.valuacion();
                //Ejercicio 8:
                if (banderaMasChico && P.pInmueble.pIdTipo == 3)
                {
                    banderaMasChico = !banderaMasChico;
                    propChico = P;
                }
                else
                {
                     if (P.pInmueble.pMetros < propChico.pInmueble.pMetros
                                && P.pInmueble.pIdTipo == 3)
                    {
                        propChico = P;
                    }
                 }
                 richMasChico.Text = propChico.toStringPropietario(); break;
        }

        //Ejercicio 2:
        txtContCasas.Text = contCasas.ToString(); txtContDeptos.Text
        =      contDeptos.ToString();      txtContLotes.Text      =
        contLotes.ToString(); //Ejercicio 3:
        if (contCasas != 0)
            txtValCasas.Text = Math.Round((acCasas / contCasas), 2).ToString();
```

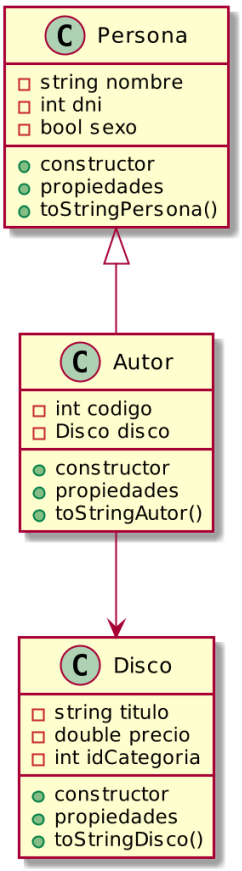
```
            else
                txtValCasas.Text = "0";
            if (contDeptos != 0)
                txtValDeptos.Text = Math.Round((acDeptos / contDeptos), ¡! 2).ToString();
            else
                txtValDeptos.Text = "0";
            if (contLotes != 0)
                txtValLotes.Text = Math.Round((acLotes / contLotes), 2).ToString();
            else
                txtValLotes.Text = "0";

            //Ejercicio 4:
            valorPromedio = (acCasas + acDeptos + acLotes) / (contCasas + contDeptos + contLotes);
            txtValPromedio.Text = Math.Round(valorPromedio, 2).ToString();
            //Ejercicio 5:
            porcentajeCasas = Convert.ToDouble(contCasas) / (contCasas + contDeptos + contLotes) * 100;
            txtPorcentajeCasas.Text = Math.Round(porcentajeCasas, 2).ToString(); //Ejercicio 6:
            txtMujDepto.Text = contMujDepto.ToString(); //Ejercicio 7:
            if (banderaMayorValor)
            {
                banderaMayorValor = !banderaMayorValor;
                propMayor = P;
            }
            else
            {
                if (P.pInmueble.valuacion() > propMayor.pInmueble.valuacion())
                {
                    propMayor = P;
                }
            }
            richMayorValor.Text = propMayor.toStringPropietario(); limpiarCampos();
        }
        private void limpiarCampos()
        {
            foreach (Control c in gbxDatos.Controls)
            {
              if (c isTextBox)
             {
                c.Text = "";
                cmbTipo.SelectedIndex = 0;
                rbMasculino.Checked = true;
              }
                    txtDni.Focus();
            }
        private void btnSalir_Click(object sender, EventArgs e)
        {
            DialogResult opcion = MessageBox.Show("¿Está seguro que desea salir?", ¡! "Salir",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (opcion ==
            DialogResult.Yes)
                Close();

        }
    }

}
```

Problema 1.3: Discos:

Dados los datos del autor (nombre, DNI, sexo y código) y de un disco (título, precio y categoría: rock, cuarteto, melódico), determinar la cantidad y porcentaje de discos para cada categoría, la cantidad de varones con disco de cuarteto, el precio promedio de los discos de rock y el autor del disco melódico más barato.

Solución






```
/*****************************************************************/
/* Persona.cs */
/*****************************************************************/
namespace ProgrI_Ej020
{
    abstract class Persona
    {
        int dni;
        string nombre;
        bool sexo;

        //Constructores:

        public Persona()
        {
            dni = 0;
            nombre = "";
            sexo = true;
        }
        public Persona(int d, string n, bool s)
        {
            dni = d;
            nombre = n;
            sexo = s;
        }
        //Propiedades:
        public int pDni
        {
            set { dni = value; }
            get { return dni; }
```

```
        }
        public string pNombre
        {
            set { nombre = value; }
            get { return nombre; }
        }
        public bool pSexo
        {
            set { sexo = value; }
            get { return sexo; }
        }
        //Métodos:
        public string toStringPersona()
        {
            string sex;
            if (sexo)
                sex = "Masculino";
            else
                sex = "Femenino";
           return"Nombre: " + nombre + "\nDNI: " + dni.ToString() + "\nSexo: "
             + sex;
        }
    }
}
/*************************************************************************/
/* Autor.cs */
/*************************************************************************/
namespace ProgrI_Ej020
{
    class Autor:Persona
    {
        int codigo;
        Disco disco;
        //Constructores:
        public Autor()
:               base()
        {
            codigo = 0;
            disco = null;
        }
        public Autor(int c, Disco disc, int d, string n, bool s)
            : base(d, n, s)
        {
            codigo = c;
            disco = disc;
        }
        //Propiedades:
        public int pCodigo
        {
            set { codigo = value; }
            get { return codigo; }
        }
        public Disco pDisco
        {
            set { disco = value; }
            get { return disco; }
        }
        //Métodos:
        public string toStringAutor()
        {
            return base.toStringPersona() + "\nCódigo Autor: " + codigo.ToString() +
                disco.toStringDisco(); ;
        }
    }
}
```

```
/*********************************************************************/
/* Disco.cs */
/*********************************************************************/
namespace ProgrI_Ej020
{
    class Disco
    {
        string titulo;

     double precio;
     int idCategoria;
     //Constructores:
     public Disco()
     {
           titulo = "";
           precio = 0;
           idCategoria = 0;
     }
     public Disco(string tit, double prec, int idCat)
     {
           titulo = tit;
           precio = prec;
           idCategoria = idCat;
     }
     //Propiedades:
     public string pTitulo
     {
          set { titulo = value; }
          get { return titulo; }
     }
     public double pPrecio
     {
          set { precio = value; }
          get { return precio; }
     }
     public int pIdCategoria
     {
          set { idCategoria = value; }
          get { return idCategoria; }
     }
     //Métodos:
     public string toStringDisco()
     {
          string categ;
          switch(idCategoria)
          {
              case 0: categ="Rock"; break;
              case 1: categ="Cuarteto";break;
                case 2: categ="Melódico";break;
                default: categ="Otra";break;
            }
            return"\nDISCO\nTítulo: " + titulo + "\nPrecio: $" + precio.ToString()
                + "\nCategoría: " + categ ;
       }
   }
```

```
/*************************************************************/
/* Form1.cs */
/*************************************************************/

namespace ProgrI_Ej020

  {
      public partial class frmDiscos : Form
      {
          int contRock, contCuarteto, contMelodico;
          int varonesCuarteto;
          double sumadorRock;
          bool bandera;
          Autor melodicoBarato;
          public frmDiscos()
          {
              InitializeComponent();

              contRock = 0;
              contCuarteto = 0;
              contMelodico = 0;
              varonesCuarteto = 0;
              sumadorRock = 0;
              melodicoBarato = null;
              bandera = true;
          }

          private void btnSalir_Click(object sender, EventArgs e)
          {
              DialogResult opcion = MessageBox.Show("¿Está seguro que desea salir?", "Salir",
                  MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (opcion ==
              DialogResult.Yes)
                  Close();
          }
          private void btnRegistrar_Click(object sender, EventArgs e)
          {
              Autor autor = new Autor();
              Disco disco = new Disco();

              autor.pCodigo = Convert.ToInt32(txtCodigo.Text); autor.pDni =
              Convert.ToInt32(txtDni.Text); autor.pNombre = txtNombre.Text;
              autor.pSexo = rbMasculino.Checked;

              disco.pIdCategoria = cmbCategoria.SelectedIndex; disco.pPrecio =
              Convert.ToDouble(txtPrecio.Text); disco.pTitulo = txtTitulo.Text;

              autor.pDisco=disco;

              switch(autor.pDisco.pIdCategoria)
              {
                  case 0: //Rock
                      contRock++;
                      sumadorRock += autor.pDisco.pPrecio;
                      break;
                  case 1: //Cuarteto


                 contCuarteto++;


                 if (autor.pSexo)
                     varonesCuarteto++;
                 break;
             case 2: //Melódico
                 if (bandera)
                 {
```

```
                    melodicoBarato = autor;
                    bandera = !bandera;
                }
                else
                {
                    if (autor.pDisco.pPrecio < melodicoBarato.pDisco.pPrecio)
                    {
                        melodicoBarato = autor;
                    }
                }
                contMelodico++;
                break;
            default:
                break;
        }
        txtCantRock.Text = contRock.ToString(); txtCantCuarteto.Text =
        contCuarteto.ToString(); txtCantMelodico.Text =
        contMelodico.ToString();

        txtPorcentRock.Text = Math.Round((Convert.ToDouble(contRock)
/               (contRock + contMelodico + contCuarteto) * 100), 2).ToString();
txtPorcentCuarteto.Text = Math.Round((Convert.ToDouble(contCuarteto)
/               (contRock + contCuarteto + contMelodico) * 100), 2).ToString();
txtPorcentMelodico.Text = Math.Round((Convert.ToDouble(contMelodico)
/               (contRock + contCuarteto + contMelodico) * 100), 2).ToString();

        lblVaronesCuarteto.Text = "Varones con discos de cuarteto: "
+               varonesCuarteto.ToString(); if(contRock !=
0)
            lblPrecioRock.Text = "Precio prom. discos de rock: "
+                   (sumadorRock / contRock).ToString();
        if(melodicoBarato!=null)
            rtxtAutorMelodicoBarato.Text = melodicoBarato.toStringAutor();

        limpiarCampos();
    }
    public void limpiarCampos()
    {
        foreach(Control c in gbxAutor.Controls)
        {
                        if (c isTextBox)
                            c.Text = "";
        }
        foreach (Control c in gbxDisco.Controls)



            {
                            if (c isTextBox)
                                c.Text = "";
            }
            cmbCategoria.SelectedIndex = 0;
            txtNombre.Focus();
            rbMasculino.Checked = true;
         }
         private void frmDiscos_Load(object sender, EventArgs e)
         {
            rbMasculino.Checked = true;
            cmbCategoria.SelectedIndex = 0;
         }
     }
 }
```

Bibliografía

- P.Bishop – Fundamentos de Informática – Anaya 1992.
- G.Brookshear – Computer Sciense: An Overview – Benjamin/Cummings. 1994.
- L. Joyanes – Problemas de Metodología de la Programación – McGraw Hill. 1990.
- L. Joyanes - Fundamentos de Programación: Algoritmos y Estructura de Datos – McGraw Hill. 1993.
- P. de Miguel – Fundamentos de los Computadores – Paraninfo. 1994.
- P. Norton – Introducción a la Computación – McGraw Hill. 1995.
- A. Prieto, A. Lloris y J.C. Torres – Introducción a la Informática – McGraw Hill. 1989.
- A. Tucker, W. Bradley, R. Cupper y D. Garnick – Fundamentos de Informática (Lógica, resolución de problemas, programas y computadoras). – McGraw Hill. 1994.