TRABAJO PRÁCTICO N° 3

PROYECTO MI DOMO

MACARENA BETSABE FERRERO

Proyecto MI DOMO es un software que registra y procesa los pedidos de solicitud de Domos Geodésicos. Se preparan Kits de armado que son distribuidos en todo el país. Los mismos pueden ser fabricados tanto en Madera de Pino con listones de 2”x 4” o con caños rígidos de PVC de 50mm.

Todo Domo Geodesico de cualquier material sólo puede ser fabricado con un diámetro mínimo de 2 m y máximo de 10 m. Además se fabrica ½ esfera, generando que la altura máxima sea el radio declarado en el sistema.

El kit de Domo en PVC sólo tiene una única conexión entre las aristas de los triángulos, a diferencia de los kit en Madera, los cuales cuentan con tres tipos diferentes de conexión: Good Karma, Cono, o Piped.

La frecuencia de los domos, corresponde a la distancia entre los triángulos interiores, en domos de mayor radio se recomienda utilizar frecuencias mayores a 1, para lograr mayor estabilidad y firmeza en el cuerpo.

Todas estas características deben ser declaradas para ingresar la solicitud de fabricación del domo en cuestión.

El software permite añadir, listar, borrar e imprimir los pedidos generados.

Recuperando al iniciar el sistema, aquellos pedidos que no fueron finalizados del día previo, reponiendo el stock diario.

TEMAS IMPLEMENTADOS

Excepciones

El software cuenta con su propia clase DomoException, la cual crea instancias, lanza y catchea los diferentes errores que pueden ocurrir durante el proceso de solicitud. Entre los posibles errores podemos encontrar el posible intento de solicitud de un domo de radio mayor o menor a lo declarado anteriormente; el tipeo incorrecto del nombre del solicitante; dejar de contar con stock suficiente para el ingreso de nuevos pedidos y cualquier fallo que pueda presentarse para la correcta utilización del sistema.

```
amespace MiDomo

    9 referencias
    public class DomoException : Exception
    {
        private string nombreClase;
        private string nombreMetodo;

        /// <summary>
        /// Constructor que recibe 3 parametros
        /// </summary>
        /// <param name="mensaje">String que indica el mensaje del problema</param>
        /// <param name="clase">String a asignar en el atributo nombreClase</param>
        /// <param name="metodo">String a asignar en el atributo nombreMetodo</param>
        2 referencias
        public DomoException(string mensaje, string clase, string metodo) : this(mensaje, clase, metodo, null)
        {
        }

        /// <summary>
        /// Constructor que recibe 4 parametros
        /// </summary>
        /// <param name="mensaje">String que indica el mensaje del problema</param>
        /// <param name="clase">String a asignar en el atributo nombreClase</param>
        /// <param name="metodo">String a asignar en el atributo nombreMetodo</param>
        /// <param name="innerException">Tipo Excepcion que indica la raiz del problema</param>
        2 referencias
        public DomoException(string mensaje, string clase, string metodo, Exception innerException)
            : base(mensaje, innerException)
        {
            this.nombreClase = clase;
            this.nombreMetodo = metodo;
        }
```

DomoException.cs

Test Unitarios

Se cuenta con un proyecto de UnitTest, el cual valida las funcionalidades de la clase DomoGeodesico.cs. En el mismo se declaran test para validar la correcta instancia de objetos que heredan del tipo Domo Geodésico; el lanzamiento de excepciones en el caso de que no se indiquen valores adecuados o fuera del rango a los atributos; el correcto funcionamiento del método que imprime la información del objeto.

```
[TestClass]
0 referencias
public class TestUnitario
{
    /// <summary>
    /// Test unitario que valida que los constructores creen correctamente el objeto
    /// </summary>
    [TestMethod]
    0 referencias
    public void Test01_Validar_Que_Se_Construyan_Los_Objetos()
    {
        //Arrange
        KitMadera kitMadera = new KitMadera(2, EFrecuencia.F1, "Federico", ETipoConexion.GoodKarma);
        KitPVC kitPVC = new KitPVC(3, EFrecuencia.F2, "Lautaro");
        //Act
        //Assert
        Assert.IsNotNull(kitMadera);
        Assert.IsNotNull(kitPVC);

    }

    /// <summary>
    /// Test Unitario que valida que se lanze una excepcion si el Radio esta fuera del rango que se permite construir
    /// </summary>
    [TestMethod]
    [ExpectedException(typeof(DomoException))]
    0 referencias
    public void Test02_Lanzar_Excepcion_Si_Se_Quiere_Asignar_Valor_Del_Radio_Fuera_Del_Rango()
    {
        //Arange
        KitMadera kitMadera = new KitMadera(6, EFrecuencia.F1, "Federico", ETipoConexion.GoodKarma);
        KitMadera kitMadera2 = new KitMadera(1, EFrecuencia.F1, "Federico", ETipoConexion.GoodKarma);

        //Act
        //Assert
    }
```

TestUnitario.cs

Tipos Genéricos

Se implementa el uso de una clase genérica, la cual contiene una lista que sólo permite que se añadan objetos del tipo Domo Geodésico. Dado que tenemos dos variantes de kits a fabricar. En la misma se inicializa la lista en el constructor y se realiza una sobrecarga del operador +, donde siempre y cuando haya stock del material necesario, se pueda sumar un domo pendiente más a construir.

```
6
7   namespace MiDomo
8   {
9       [Serializable]
        10 referencias
10      public class PendienteAConstruir<T>
11          where T : DomoGeodesico
12      {
13          protected List<T> listadoPendientes;
14
15          /// <summary>
16          /// Constructor que inicializa la lista generica
17          /// </summary>
            3 referencias
18          public PendienteAConstruir()
19          {
20              listadoPendientes = new List<T>();
21          }
22
23          /// <summary>
24          /// Propiedad publica de lectura y escritura,
25          /// que devuelve la lista
26          /// </summary>
            9 referencias
27          public List<T> ListaPendientes
28          {
29              get
30              {
31                  return listadoPendientes;
32              }
33              set
34              {
35                  listadoPendientes = value;
36              }
37
38          }
39
```

PendienteAConstruir.cs

Interfaces

Para el correcto funcionamiento del software, se utilizan dos interfaces diferentes, una de ellas es implementada por las herederas de Domo Geodésico, ya que cuenta con una propiedad de sólo lectura que indica la cantidad de días que demora la confección del pedido. Esta demora del trabajo depende del tipo de material y frecuencia de domo seleccionado.
La segunda interfaz se implementa para guardar y leer el archivo al serializar.

```
namespace MiDomo
{
    2 referencias
    public interface IAcciones
    {
        /// <summary>
        /// Propiedad de solo lectura, indica cantidad de dias de Fabricacion
        /// </summary>
        4 referencias
        int CantidadDiasDeFabricacion { get; }

    }
}
```

IAcciones.cs

```
namespace MiDomo
{
    1 referencia
    interface IArchivo<T>
    {
        /// <summary>
        /// Guarda un objeto del tipo T en un archivo en una ruta declarada
        /// </summary>
        /// <param name="rutaArchivo">String donde se creara el archivo</param>
        /// <param name="info">Objeto T con la informacion a guardar en el archivo</param>
        /// <returns>Retorna true si pudo hacerlo, False sino</returns>
        2 referencias
        bool Guardar(string rutaArchivo, T info);

        /// <summary>
        /// Lee un archivo en la ruta declarada
        /// </summary>
        /// <param name="rutaArchivo">String de donde se leera el archivo</param>
        /// <param name="info">Objeto T con la informacion a leer en el archivo</param>
        /// <returns>Retorna true si pudo hacerlo, False sino</returns>
        2 referencias
        bool Leer(string rutaArchivo, out T info);

    }
}
```

IArchivo.cs

Archivos

Se implementa la confección de archivos al posibilitar imprimir el informe detallado en el Form Informe. Al confirmar que se desea imprimir el listado de los pedidos pendientes, si no hay un archivo previo lo crea, pero en el caso de que sí existiera el archivo, el software permite sobreescribir el mismo o concatenar este último al listado existente.
También como se mencionó al principio, se permite la eliminación del listado pendiente en el caso que hiciera falta, fuese confusión o pedidos que pasaron a la etapa de distribución.

```
/// <summary>
/// Evento del Boton imprimir que genera un archivo con el listado pendiente a construir
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 referencia
private void btnImprimir_Click(object sender, EventArgs e)
{
    string file_name = AppDomain.CurrentDomain.BaseDirectory + "ArchivoDeInforme01.txt";

    try
    {
        if (File.Exists(file_name) && (MessageBox.Show($"El archivo {file_name} existe. ¿Desea sobreescribirlo (SI) " +
            $"o agregar los datos al existente (NO)?", "Aviso", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes))
        {
            using (StreamWriter sw = new StreamWriter(file_name))
            {
                if (pendienteAConstruir2.ListaPendientes.Count >= 1)
                {
                    sw.Write("Fecha del Informe: ");
                    sw.WriteLine(DateTime.Now);
                    sw.WriteLine("----------------------------");

                    sw.WriteLine(Mostrar(pendienteAConstruir2));
                }

                MessageBox.Show("El archivo fue modificado correctamente", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
        else
        {
            using (StreamWriter sw = new StreamWriter(file_name, true))
            {
                sw.Write("Fecha del Informe: ");
                sw.WriteLine(DateTime.Now);
                sw.WriteLine("----------------------------");

                sw.WriteLine(Mostrar(pendienteAConstruir2));
                MessageBox.Show("El archivo fue creado correctamente", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
```

FormInforme.cs

Serialización

Proyecto MI DOMO, al iniciar el programa recopila toda la información de domos pendientes a construir si la hubiera. Esta funcionalidad se implementa con la serialización del listado pendiente a construir. Recuperando el archivo al inicio del programa si existiese y guardando el mismo al finalizar el día de trabajo.

```
4 referencias
public partial class FormMiDomo : Form
{
    KitMadera kitMadera;
    KitPVC kitPVC;
    PendienteAConstruir<DomoGeodesico> pendienteAConstruir;
    List<DomoGeodesico> listaDomos;
    XML<List<DomoGeodesico>> xmlSerializador;
    string rutaArchivo = $"{Environment.CurrentDirectory}\\pendienteAContruir.xml";

    /// <summary>
    /// Constructor del formulario que inicializa la lista
    /// </summary>
    1 referencia
    public FormMiDomo()
    {
        InitializeComponent();
        pendienteAConstruir = new PendienteAConstruir<DomoGeodesico>();
        xmlSerializador = new XML<List<DomoGeodesico>>();
        listaDomos = pendienteAConstruir.ListaPendientes;
    }

    /// <summary>
    /// Al cargar el formulario se seleccionan propiedades por defecto
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    1 referencia
    private void FormMiDomo_Load(object sender, EventArgs e)
    {
        cmbConexion.SelectedIndex = 0;
        cmbFrecuencia.SelectedIndex = 0;
        rbtMadera.Checked = true;
        try
        {
            if (File.Exists(rutaArchivo))
            {
                if (xmlSerializador.Leer(rutaArchivo, out listaDomos))
                {
                    pendienteAConstruir.ListaPendientes = listaDomos;
                    MessageBox.Show("Archivo cargaado correctamente", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
```

FormMiDomo.cs