Министерство науки и высшего образования Российской Федерации

Новосибирский Государственный технический университет

Кафедра автоматизированных систем управления



**Отчет по лабораторной работе №3**

**по дисциплине «Технологии и методы программирования»**

**«РЕАЛИЗАЦИЯ ПРОГРАММНЫХ МОДУЛЕЙ МНОГОКРАТНОГО ИСПОЛЬЗОВАНИЯ»**

Выполнил

студент группы АП-027

Савченко С.В.

г. Новосибирск

2022 г

1.ЦЕЛЬ РАБОТЫ

Ознакомиться с приёмами создания программных модулей многократного использования, научиться применять их при проектировании и разработке программного обеспечения.

2. ПОСТАНОВКА ЗАДАЧИ И РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ

Описание класса и все модули, созданные при выполнении работы 2, оформить как DLL. Тестирующая программа должна повторить задание работы 2, используя созданную DLL и разные варианты связывания.

Код:

DLL:

|  |
| --- |
| MenuStructure |
| namespace \_3Laba  {  public class MenuStructure  {  public int NumberHierarchy { get; set; }  public string Name { get; set; }  public int Status { get; set; }  public string MethodName { get; set; }  public List<MenuStructure> SubMenu { get; set; }  public MenuStructure()  {  SubMenu = new List<MenuStructure>();  }  }  } |
| Program |
| namespace \_3Laba  {  internal static class Program  {  [STAThread]  static void Main()  {  string filePath = @"D:\ТиМП\2Laba\2Laba\TextFile1.txt";  ApplicationConfiguration.Initialize();  Application.Run(new Form1(filePath));  }  }  } |
| Form1.Designer.cs |
| namespace MyLib  {  partial class UserControl1  {  /// <summary>  /// Обязательная переменная конструктора.  /// </summary>  private System.ComponentModel.IContainer components = null;  /// <summary>  /// Освободить все используемые ресурсы.  /// </summary>  /// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>  protected override void Dispose(bool disposing)  {  if (disposing && (components != null))  {  components.Dispose();  }  base.Dispose(disposing);  }  #region Код, автоматически созданный конструктором компонентов  /// <summary>  /// Требуемый метод для поддержки конструктора — не изменяйте  /// содержимое этого метода с помощью редактора кода.  /// </summary>  private void InitializeComponent()  {  this.menuStrip1 = new System.Windows.Forms.MenuStrip();  this.SuspendLayout();  //  // menuStrip1  //  this.menuStrip1.Location = new System.Drawing.Point(0, 0);  this.menuStrip1.Name = "menuStrip1";  this.menuStrip1.Size = new System.Drawing.Size(800, 24);  this.menuStrip1.TabIndex = 0;  this.menuStrip1.Text = "menuStrip1";  //  // UserControl1  //  this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);  this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  this.ClientSize = new System.Drawing.Size(800, 450);  this.Controls.Add(this.menuStrip1);  this.MainMenuStrip = this.menuStrip1;  this.Name = "UserControl1";  this.Text = "3Laba";  this.ResumeLayout(false);  this.PerformLayout();  }  #endregion  private System.Windows.Forms.MenuStrip menuStrip1;  }  } |
| Form1.cs |
| using System.Diagnostics;  namespace \_3Laba  {  public partial class Form1 : Form  {  private Dictionary<string, string> fileNames = new Dictionary<string, string>();  private List<MenuStructure> menus = new List<MenuStructure>();  private MenuStructure menu = new MenuStructure();  private MenuStructure menuHead = new MenuStructure();  private List<MenuStructure> previos = new List<MenuStructure>();  private int level = 0;  private ToolStripMenuItem tool = new ToolStripMenuItem();  private ToolStripMenuItem toolHead;  private List<ToolStripMenuItem> toolsHead = new List<ToolStripMenuItem>();  private List<ToolStripMenuItem> result = new List<ToolStripMenuItem>();  private Dictionary<ToolStripMenuItem, int> previosToolsWithLevel = new Dictionary<ToolStripMenuItem, int>();  public Form1(string path)  {  InitializeComponent();  using var reader = new StreamReader(path);  string? line;  while ((line = reader.ReadLine()) != null)  {  var text = line.Split(' ');  if (text.Count() == 3)  {  MenuStructure menu = new MenuStructure()  {  NumberHierarchy = Convert.ToInt32(text[0]),  Name = text[1],  Status = Convert.ToInt32(text[2]),  };  if (menu.NumberHierarchy > this.menu.NumberHierarchy)  {  this.menu.SubMenu.Add(menu);  }  else if (menu.NumberHierarchy > menuHead.NumberHierarchy)  {  var item = previos.LastOrDefault(x => x.NumberHierarchy == menu.NumberHierarchy - 1);  item.SubMenu.Add(menu);  }  else  {  menuHead = menu;  menus.Add(menuHead);  }  previos.Add(menu);  this.menu = menu;  }  else  {  MenuStructure menu = new MenuStructure()  {  NumberHierarchy = Convert.ToInt32(text[0]),  Name = text[1],  Status = Convert.ToInt32(text[2]),  MethodName = text[3]  };  if (menu.NumberHierarchy > menuHead.NumberHierarchy)  {  if (menu.NumberHierarchy > this.menu.NumberHierarchy)  {  this.menu.SubMenu.Add(menu);  }  else  {  var item = previos.LastOrDefault(x => x.NumberHierarchy == menu.NumberHierarchy - 1);  item.SubMenu.Add(menu);  }  }  else  {  menus.Add(menu);  }  }  }  Show(menus);  }    private void Show(List<MenuStructure> menus)  {  for (int i = 0; i < menus.Count; i++)  {  ToolStripMenuItem tool = new ToolStripMenuItem(menus[i].Name);  if (menus[i].Status == 2)  {  continue;  }  if (menus[i].MethodName != null && menus[i].Status != 1)  {  fileNames.Add(menus[i].MethodName, menus[i].Name);  tool.Click += Click;  }  if (menus[i].SubMenu.Count > 0)  {  if(toolHead == null)  {  toolHead = tool;  previosToolsWithLevel.Add(tool, level);  level++;  Show(menus[i].SubMenu);  level--;  result.Add(toolHead);  toolHead = null;  continue;  }  else  {  previosToolsWithLevel.Add(tool, level);  this.tool = previosToolsWithLevel.LastOrDefault(x => x.Value == level - 1).Key;  this.tool.DropDownItems.Add(tool);  if (toolHead != this.tool)  {  if (menus[i].SubMenu.Count == 3)  {  toolsHead.Add(this.tool);  toolHead.DropDownItems.AddRange(toolsHead.ToArray());  }  }  level++;  Show(menus[i].SubMenu);  level--;  continue;  }  }  if (toolHead != null)  {  if (level == 1)  {  toolHead.DropDownItems.Add(tool);  }  else  {  var t = previosToolsWithLevel.LastOrDefault(x => x.Value == level - 1);  this.tool = t.Key;  this.tool.DropDownItems.Add(tool);  }  }  else  {  if (result.Count > 0)  {  foreach (var item in result)  {  menuStrip1.Items.Add(item);  }  toolsHead = new List<ToolStripMenuItem>();  result = new List<ToolStripMenuItem>();  }  menuStrip1.Items.Add(tool);  }  }  }  private void Click(object sender, EventArgs e)  {  foreach (var item in fileNames)  {  if (item.Value == ((ToolStripMenuItem)sender).Text)  {  MessageBox.Show($"Hello from {item.Key}");  }  }  }  }  } |
| TextFile1.txt |
| 0 Разное 0 method  0 Сотрудники 0  1 Приказы 0 method1  0 Разное 0  1 Сотрудники 0 K  0 Приказы 0 method2  0 Разное 0 method3  0 Сотрудники 0  1 Приказы 0 method4  0 Документы 0  1 Справочники 0  2 Отделы 0 method5  1 Города 0  2 Должности 0  3 Окно 0 Window  0 Справка 0 Yelp |

Старт программы

|  |
| --- |
| Program.cs |
| using System.Reflection;  namespace \_3Laba  {  internal static class Program  {  static void Main()  {  string fileName = @"D:\ТиМП\2Laba\2Laba\TextFile1.txt";  string libraryName = "MyLib.dll";  var typeName = "MyLib.UserControl1";  string methodName = "UserControl1";  ApplicationConfiguration.Initialize();  #region неявное связывание  //Application.Run(new UserControl1(@"D:\ТиМП\2Laba\2Laba\TextFile1.txt"));  #endregion  #region явное связывание  Assembly assembly = Assembly.LoadFrom(libraryName);  var type = assembly.GetType(typeName);  if(type is not null)  {  object? obj = Activator.CreateInstance(type, fileName);  MethodInfo? init = type.GetMethods().FirstOrDefault(x => x.Name == methodName);  if(obj is Form form)  {  Application.Run(form);  }  }  #endregion  }  }  } |

Результат:

