Министерство науки и высшего образования Российской Федерации

Новосибирский Государственный технический университет

Кафедра автоматизированных систем управления



**Отчет по лабораторной работе № 2**

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

**«**реализация программы, управляемой метаданными**»**

Выполнили

студенты группы АП-026

Давыдов М.С.

Косточкин М.

г. Новосибирск

2021 г

**1. ЗАДАНИЕ**

1) Реализовать класс, конструктор которого создаёт меню, внешний вид которого определяется данными, находящимися во внешнем текстовом файле. Аргументом конструктора будет имя файла. Меню программы рассматривается как множество деревьев (иерархий), корнями которых являются пункты главного меню (уровень иерархии 0). Следующие уровни иерархии представляют подпункты, при этом количество уровней произвольное для каждого пункта (подпункта).

Каждая запись файла имеет следующую структуру:

Нмер\_уровня\_в\_иерархииПробелНазвание\_пунктаПробелСтатус\_пунктаПробелИмяМетода

Последовательность записей соответствует последовательности пунктов. Статус пункта определяется:

0 – пункт виден и доступен;

1 – пункт виден, но не доступен;

2 – пункт не виден.

ИмяМетода – имя обработчика клика мыши на пункте меню.

Если имя обработчика не указано, это означает, что пункт имеет подменю, которое и должно появиться, после выбора этого пункта.

2) Любым текстовым редактором создать файл, описывающий структуру меню.

3) Написать тестовую программу для проверки работы класса. Программа должна построить окно, в котором поместит меню по описанию.

4) При выборе мышью пункта срабатывает обработчик пункта или появляется подменю, если имя метода не указано. Обработчик пункта должен вызвать метод с указанным именем, который просто обозначит себя.

**2. ИСХОДНЫЙ ТЕКСТ КЛАССА И ТЕСТИРУЮЩЕЙ ПРОГРАММЫ**

|  |
| --- |
| Класс MenuItemTree |
| using System.Collections.Generic;  using System.Windows.Forms;  namespace ProgrammingMethodsLab\_2  {  public class MenuItemsTree  {  public List<MenuItemsTree> nextLevelNodes = new List<MenuItemsTree>();  public int ItemStatus { get; private set; }  public string ItemName { get; private set; }  public string ItemMethod { get; private set; }  public ToolStripMenuItem MenuItem { get; private set; }  public MenuItemsTree(string value, int status, string method)  {  ItemStatus = status;  ItemName = value;  ItemMethod = method;  MenuItem = new ToolStripMenuItem(ItemName);  }  public MenuItemsTree(string value, int status)  {  ItemStatus = status;  ItemName = value;  MenuItem = new ToolStripMenuItem(ItemName);  }  public MenuItemsTree FindNode(string nodeData\_)  {  if (ItemName == nodeData\_)  {  return this;  }  foreach (MenuItemsTree item in nextLevelNodes)  {  if (item.ItemName == nodeData\_) return item;  MenuItemsTree ret = item.FindNode(nodeData\_);  if (ret != null) return ret;  }  return null;  }  public bool JoinToNode(string val, string toJoin, int status, string method)  {  MenuItemsTree joined = FindNode(val);  if (joined == null)  {  return false;  }  else  {  MenuItemsTree newTree = new MenuItemsTree(toJoin, status, method);  if (status == 2)  {  newTree.MenuItem.Visible = false;  }  joined.nextLevelNodes.Add(newTree);  return true;  }  }  public bool JoinToNode(string val, string toJoin, int status)  {  MenuItemsTree joined = FindNode(val);  if (joined == null)  {  return false;  }  else  {  MenuItemsTree newTree = new MenuItemsTree(toJoin, status);  if (status == 2)  {  newTree.MenuItem.Visible = false;  }  joined.nextLevelNodes.Add(newTree);  return true;  }  }  }  } |
| Класс MenuCreator |
| using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Windows.Forms;  namespace ProgrammingMethodsLab\_2  {  public class MethodArgs : EventArgs  {  public string value { get; private set; }  public MethodArgs(string value)  {  this.value = value;  }  }  class MenuCreator  {  private List<MenuItemsTree> menuItems;  public MenuCreator(string filename, MenuStrip menuStrip)  {  menuStrip.Items.Clear();  List<string[]> elements = new List<string[]>();  StreamReader f = new StreamReader(filename);  while (!f.EndOfStream)  {  string s = f.ReadLine();  string[] words = s.Split(' ');  if (words.Count() != 3 && words.Count() != 4)  {  f.Close();  throw new Exception("Неверное количество слов в строке!");  }  if (!Int32.TryParse(words[0], out \_) || !Int32.TryParse(words[2], out \_))  {  f.Close();  throw new Exception("В первом или третьем слове строк(и) находятся не числа!");  }  elements.Add(words);  }  f.Close();  menuItems = new List<MenuItemsTree>();  int i = 0;  Stack<int> parentsID = new Stack<int>();  Stack<string> parentsName = new Stack<string>();  int CurrentLevel = 0;  while (i < elements.Count())  {  if (elements[i][0] != "0" && parentsID.Count == 0)  {  throw new Exception("Нерректная иерархия!");  }  if (Int32.Parse(elements[i][0]) < 0)  {  throw new Exception("Номер иерархии предмета(ов) меньше нуля!");  }  if (elements[i][0] == "0" && elements[i].Length == 4)  {  menuItems.Add(new MenuItemsTree(elements[i][1], Int32.Parse(elements[i][2]), elements[i][3]));  }  if (elements[i][0] == "0" && (elements[i].Length == 3))  {  menuItems.Add(new MenuItemsTree(elements[i][1], Int32.Parse(elements[i][2])));  parentsID.Push(menuItems.Count - 1);  parentsName.Push(elements[i][1]);  CurrentLevel++;  }  if (Int32.Parse(elements[i][0]) == CurrentLevel && CurrentLevel != 0)  {  if (elements[i].Length == 3)  {  menuItems[parentsID.Peek()].JoinToNode(parentsName.Peek(), elements[i][1], Int32.Parse(elements[i][2]));  parentsID.Push(menuItems.Count - 1);  parentsName.Push(elements[i][1]);  }  else  {  menuItems[parentsID.Peek()].JoinToNode(parentsName.Peek(), elements[i][1], Int32.Parse(elements[i][2]), elements[i][3]);  }  }  else if (Int32.Parse(elements[i][0]) > CurrentLevel)  {  CurrentLevel++;  if(Int32.Parse(elements[i][0]) != CurrentLevel || parentsName.Count == 0)  {  throw new Exception("Неверная иерархия меню!");  }  if (elements[i].Length == 3)  {  menuItems[parentsID.Peek()].JoinToNode(parentsName.Peek(), elements[i][1], Int32.Parse(elements[i][2]));  parentsID.Push(menuItems.Count - 1); parentsName.Push(elements[i][1]);  }  else  {  menuItems[parentsID.Peek()].JoinToNode(parentsName.Peek(), elements[i][1], Int32.Parse(elements[i][2]), elements[i][3]);  }  }  else if (Int32.Parse(elements[i][0]) < CurrentLevel)  {  int difference = CurrentLevel - Int32.Parse(elements[i][0]);  CurrentLevel = Int32.Parse(elements[i][0]);    if (parentsID.Count() != 0 && parentsName.Count() != 0)  {  for (int j = 0; j < difference; j++)  {  parentsID.Pop();  parentsName.Pop();  }  }  if (parentsID.Count() >= 1 && parentsName.Count() >= 1)  {  if (elements[i].Length == 3)  {  menuItems[parentsID.Peek()].JoinToNode(parentsName.Peek(), elements[i][1], Int32.Parse(elements[i][2]));  }  else  {  menuItems[parentsID.Peek()].JoinToNode(parentsName.Peek(), elements[i][1], Int32.Parse(elements[i][2]), elements[i][3]);  }  }  if (elements[i].Length == 3 )  {  parentsID.Push(menuItems.Count - 1);  parentsName.Push(elements[i][1]);  }  }  i++;  }  foreach (MenuItemsTree treeParent in menuItems)  {  SetProperties(treeParent);  AddChildren(treeParent);  menuStrip.Items.Add(treeParent.MenuItem);  }  }  private void SetProperties(MenuItemsTree item)  {  if (item.nextLevelNodes.Count == 0 && item.ItemStatus == 0)  {  MethodArgs ee = new MethodArgs(item.ItemMethod);  item.MenuItem.Click += new EventHandler((sender, e) => AddedItemClickEvent(sender, ee));  }  else if (item.ItemStatus == 1)  item.MenuItem.Enabled = false;  else if (item.ItemStatus == 2)  item.MenuItem.Visible = false;  }  private void AddChildren(MenuItemsTree nonBinTree)  {  foreach (MenuItemsTree treeChild in nonBinTree.nextLevelNodes)  {  SetProperties(treeChild);  AddChildren(treeChild);  nonBinTree.MenuItem.DropDownItems.Add(treeChild.MenuItem);  }  }  private void AddedItemClickEvent(object sender, MethodArgs e)  {  MessageBox.Show($"Вы нажали на {e.value} ");  }  }  } |
| Класс Form1 |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Windows.Forms;  using System.IO;  namespace ProgrammingMethodsLab\_2  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  openFileDialog1.Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";  openFileDialog1.Title = "Открыть файл с структурой меню";  openFileDialog1.FileName = "";  }  private void Form1\_Load(object sender, EventArgs e)  {  }  private void Button1Click(object sender, EventArgs e)  {  if (openFileDialog1.ShowDialog() == DialogResult.Cancel)  return;  string filename = openFileDialog1.FileName;  try  {  MenuCreator menu = new MenuCreator(filename, menuStrip1);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  }  } |

**3. ТЕКСТ УПРАВЛЯЮЩЕГО ФАЙЛА**

Файл, содержащий описание этого меню, содержит следующие записи:

0 Разное 0 Others

0 Сотрудники 0 Stuff

0 Приказы 0 Orders

0 Документы 0 Docs

0 Справочники 0

1 Отделы 0 Departs

1 Должности 0

2 Свободные 0

3 Смена 1 FullTime

3 Полсмены 0 PartTime

2 Занятые 0 Occupied

1 Города 0 Towns

0 Окно 1 Window

0 Справка 0 Help

**4. СКРИНШОТ ЭКРАНА**

