**ФГБОУ ВО**

**Национальный исследовательский университет**

**«МЭИ»**

**Лабораторная работа № 7**

по курсу

«Технология программирования»

**МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ**

**Выполнил:**

еще не отчисленный

студент группы А-08-23

Потапов О. В.

Вариант №16

**Москва, 2024**

Задание

1. Построить иерархию классов в соответствии с вариантом задания.
2. Разработать методы и свойства для каждого из определяемых классов.
3. Реализовать программу на C# в соответствии с вариантом задания.
4. Изменить иерархию классов с использованием абстрактного виртуального класса и интерфейсами
5. Реализовать обработку исключительных ситуаций
6. Рассмотреть и описать, какие проблемы возникли при реализации множественного наследования ("алмаз смерти").

Вариант 16

Устройство, принтер, сканер, МФУ

**1. Постановка задачи**

Передо мной стоит задача: создание ромбовидной иерархии классов из устройства, принтера, сканера и МФУ, добавить чистый абстрактный класс во для изменения данной иерархии и создание иерархии интерфейсов.

Разработать несколько методов для понимания структуры работы множественного наследования. Добавить пользовательский интерфейс с использованием Windows Forms.

Пусть наши классы объединяются следующим интерфейсом:

1. Общие состояния:
   * объект включен в сеть/выключен из сети
   * объект разобран/собран
2. Общие методы:
   * включить/выключить устройство из сети
   * разобрать устройство
   * собрать устройство
   * отнести на починку в мастерскую
3. Общие поля:
   * цена устройства
   * производитель устройства

Устройство не будет обладать какими-то уникальными методами или полями, в отличие от принтера и сканера.

Принтер, в свою очередь, будет обладать способностью печатать и вести учет количества бумаги. Сканер будет сканировать. Да, это важно.

МФУ будет лишь объединять функционал этих двух устройств: печатать, вести учет бумаги и сканировать. Добавим ему еще функцию копирования: объединения сканирования и печати.

1. **Разработка приложения**
2. **Разработка структуры приложения**

На Рис. 2.1. изображена иерархия интерфейсов проекта, где видна реализация ромбовидного наследования.

На Рис. 2.2. изображена иерархия классов проекта: диаграмма абстрактного класса Device\_interface, класса «устройств» Device, класса «принтер» Printer, класса «сканер» Scanner, класса «МФУ» MFP, а также используемые делегаты для обработки событий класса.

На Рис. 2.3. изображены вспомогательные классы, такие как обработчик событий, FileReader, класс аргументов для передачи событий.

В Табл. 2.1 описаны методы и свойства интерфейсов IDevice и его наследников.

В Табл. 2.2 описаны методы абстрактного класса Device\_interface, который является базовым для класса Device.

В Табл. 2.3 описаны методы класса Device.

В Табл. 2.4 описаны методы класса Printer.

В Табл. 2.5 описаны методы класса Scanner.

В Табл. 2.6 описаны методы класса MFP.

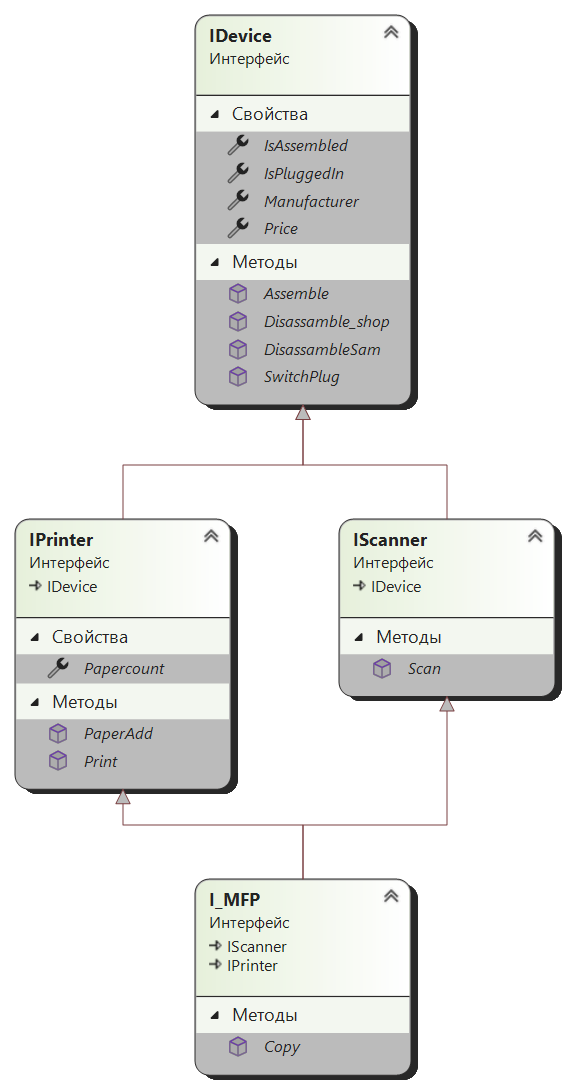


Рис 2.1. Диаграмма интерфейсов

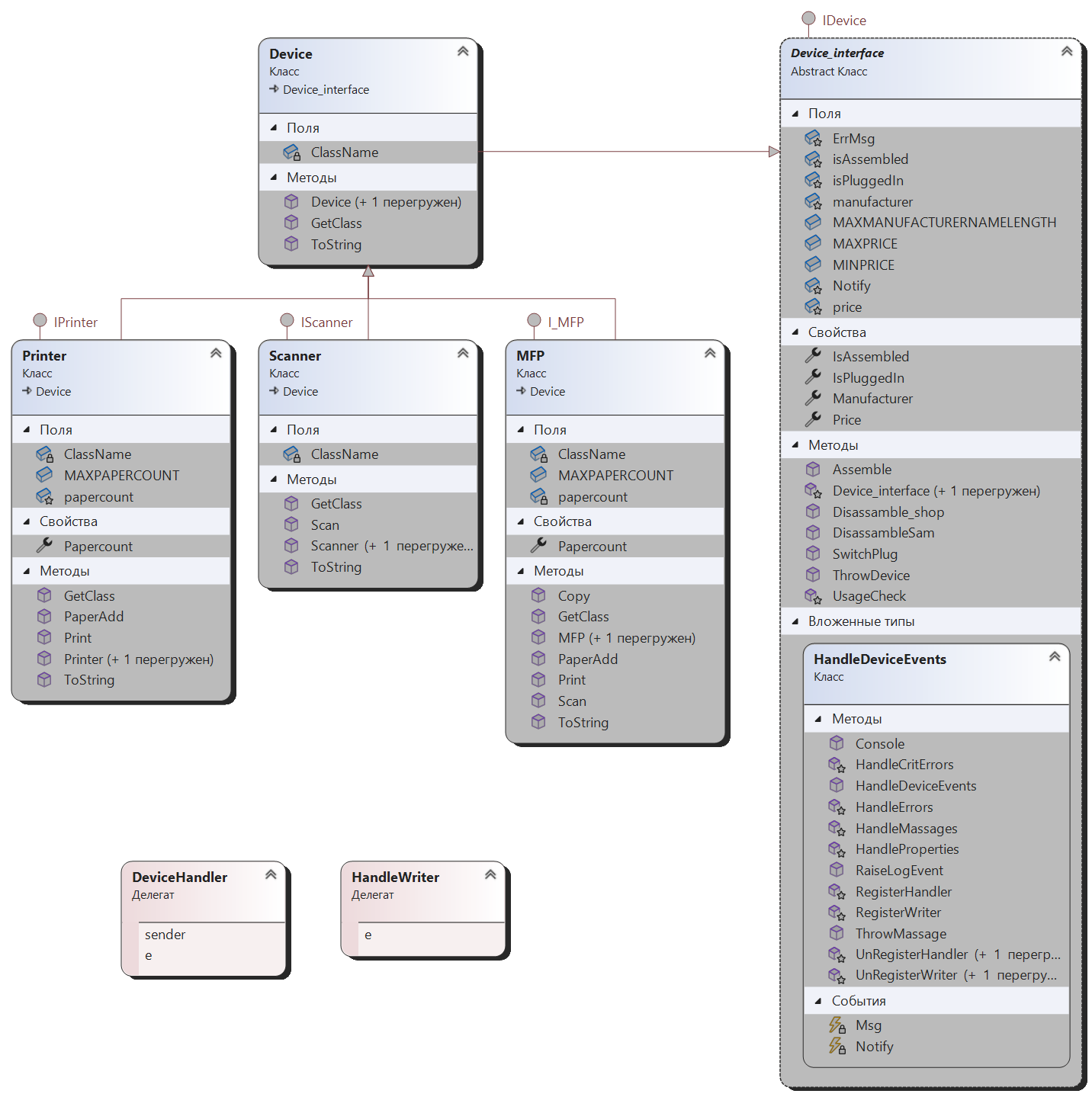


Рис 2.2. Диаграмма классов

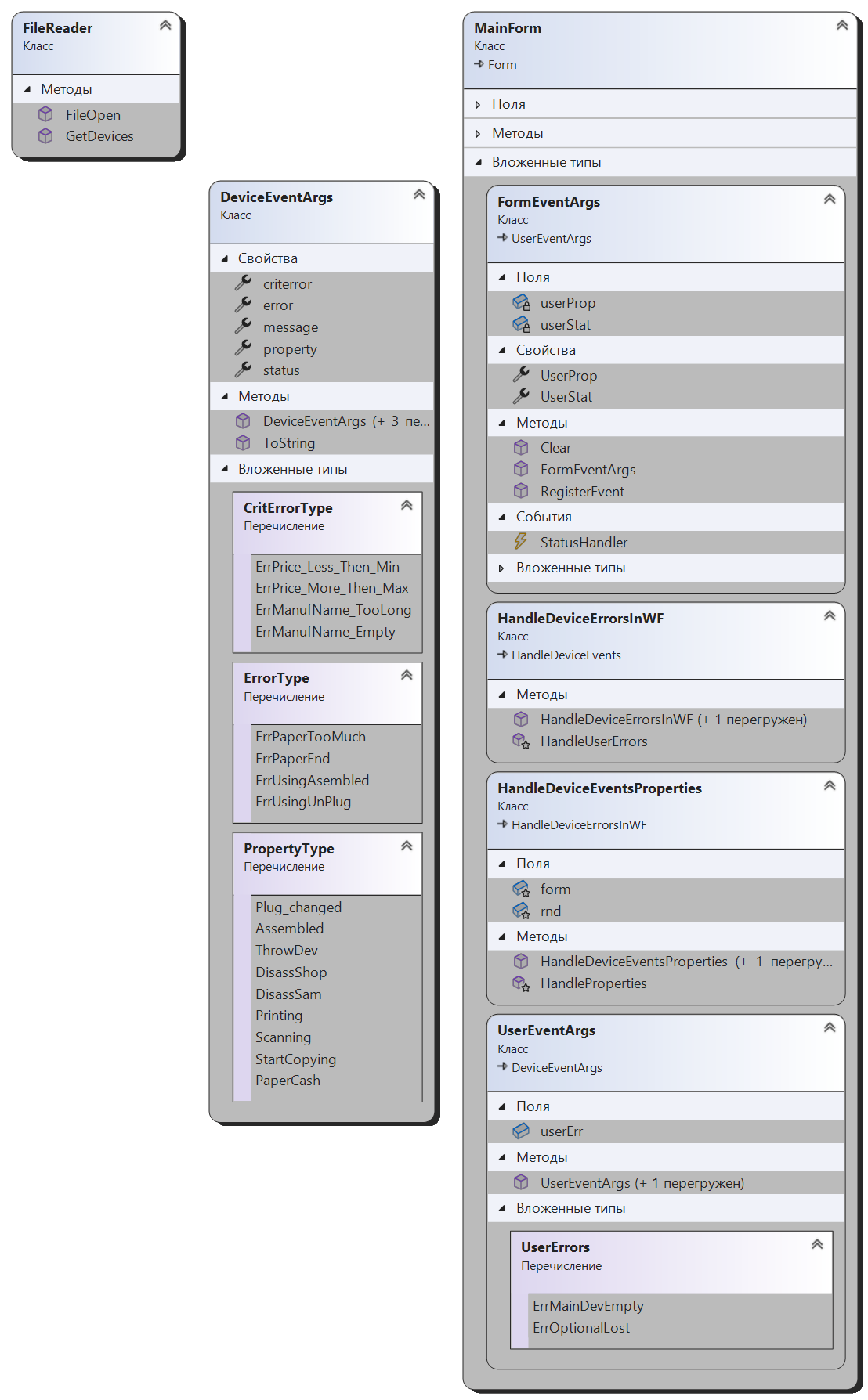


Рис 2.3. Диаграмма вспомогательных классов

Табл. 2.1. Интерфейсы для класса Device

| **Название элемента** | **Назначение, тип элемента** | **Входные данные** | **Выходные данные** |
| --- | --- | --- | --- |
| IDevice | | | |
| Price | int, свойство  цена устройства | - | int |
| Manufacturer | string, свойство  название производителя | - | string |
| IsPluggedIn | bool, свойство  состояние устройства включения в сеть | - | bool |
| IsAssembled | bool, свойство  состояние разобранности устройства | - | bool |
| SwitchPlug | void, функция  изменение сосояния включения в сеть | - | - |
| Assemble | void, функция  разборка усройства | - | - |
| DisassebleSam | void, функция  самостоятельная сборка устройства | - | - |
| Disassemble\_shop | void, функция  отнести устройство в мастерскую | - | - |
| IPrinter: IDevice | | | |
| Papercount | int, свойство  количество бумаги | - | int |
| Print | void, функция  печать | int paperused = 1  количество используемой для печати бумаги | - |
| PaperAdd | void, функция  добивить бумагу | int paper\_add = 1  количество бумаги для добавления | - |
| IScanner: IDevice | | | |
| Scan | void, функция  сканировать | - | - |
| I\_MFP: IPinter, IScanner | | | |
| Copy | void, функция  копировать | int paperused = 1  количество используемой бумаги | - |

Табл. 2.2. Методы класса Device\_interface

| **Название элемента** | **Назначение, тип элемента** | **Входные данные** | **Выходные данные** |
| --- | --- | --- | --- |
| Device\_interface | конструктор | int price: цена устройства  string manufacturer: производитль устройства | - |
| Device\_interface | конструктор | int price: цена устройства  string manufacturer: производитель устройства  HandleDeviceEvents notify: обработчик событий объекта устройства  HandleDeviceEvents errMsg = null: обработчик ошибок объекта устройства | - |
| SwitchPlug | void  изменение сосояния включения в сеть | - | - |
| Assemble | void  разборка усройства | - | - |
| DisassebleSam | void  самостоятельная сборка устройства | - | - |
| Disassemble\_shop | void  отнести устройство в мастерскую | - | - |
| ThrowDevice | void  выбросить устройство | - | - |
| UsageCheck | protected void  проверка возможности воспользоваться устройством | - | bool, можно ли пользоваться устройством |

Продолжение табл. 2.2.

Табл. 2.3. Методы класса Device

| **Название элемента** | **Назначение, тип элемента** | **Входные данные** | **Выходные данные** |
| --- | --- | --- | --- |
| Device | конструктор | int price: цена устройства  string manufacturer: производитль устройства | - |
| Device | конструктор | int price: цена устройства  string manufacturer: производитель устройства  HandleDeviceEvents notify: обработчик событий объекта устройства  HandleDeviceEvents errMsg = null: обработчик ошибок объекта устройства | - |
| ToString | override string  название класса | - | string  название класса |
| GetClass | static string  называние класса | - | string  название класса |

Табл. 2.4. Методы класса Printer

| **Название элемента** | **Назначение, тип элемента** | **Входные данные** | **Выходные данные** |
| --- | --- | --- | --- |
| Printer | конструктор | int price: цена устройства  string manufacturer: производитль устройства | - |
| Printer | конструктор | int price: цена устройства  string manufacturer: производитель устройства  HandleDeviceEvents notify: обработчик событий объекта устройства  HandleDeviceEvents errMsg = null: обработчик ошибок объекта устройства | - |
| Print | void, функция  печать | int paperused = 1  количество используемой для печати бумаги | - |
| PaperAdd | void, функция  добивить бумагу | int paper\_add = 1  количество бумаги для добавления | - |
| ToString | override string  название класса | - | string  название класса |
| GetClass | static string  называние класса | - | string  название класса |

Табл. 2.5. Методы класса Scanner

| **Название элемента** | **Назначение, тип элемента** | **Входные данные** | **Выходные данные** |
| --- | --- | --- | --- |
| Scanner | конструктор | int price: цена устройства  string manufacturer: производитль устройства | - |
| Scanner | конструктор | int price: цена устройства  string manufacturer: производитель устройства  HandleDeviceEvents notify: обработчик событий объекта устройства  HandleDeviceEvents errMsg = null: обработчик ошибок объекта устройства | - |
| Scan | void, функция  сканировать | - | - |
| ToString | override string  название класса | - | string  название класса |
| GetClass | static string  называние класса | - | string  название класса |

Продолжение табл. 2.5.

Табл. 2.6. Методы класса MFP

| **Название элемента** | **Назначение, тип элемента** | **Входные данные** | **Выходные данные** |
| --- | --- | --- | --- |
| MFP | конструктор | int price: цена устройства  string manufacturer: производитль устройства | - |
| MFP | конструктор | int price: цена устройства  string manufacturer: производитель устройства  HandleDeviceEvents notify: обработчик событий объекта устройства  HandleDeviceEvents errMsg = null: обработчик ошибок объекта устройства | - |
| Print | void, функция  печать | int paperused = 1  количество используемой для печати бумаги | - |
| PaperAdd | void, функция  добивить бумагу | int paper\_add = 1  количество бумаги для добавления | - |
| Scan | void, функция  сканировать | - |  |
| Copy | void, функция  копировать | - |  |
| ToString | override string  название класса | - | string  название класса |
| GetClass | static string  называние класса | - | string  название класса |

1. **Разработка схемы алгоритма**

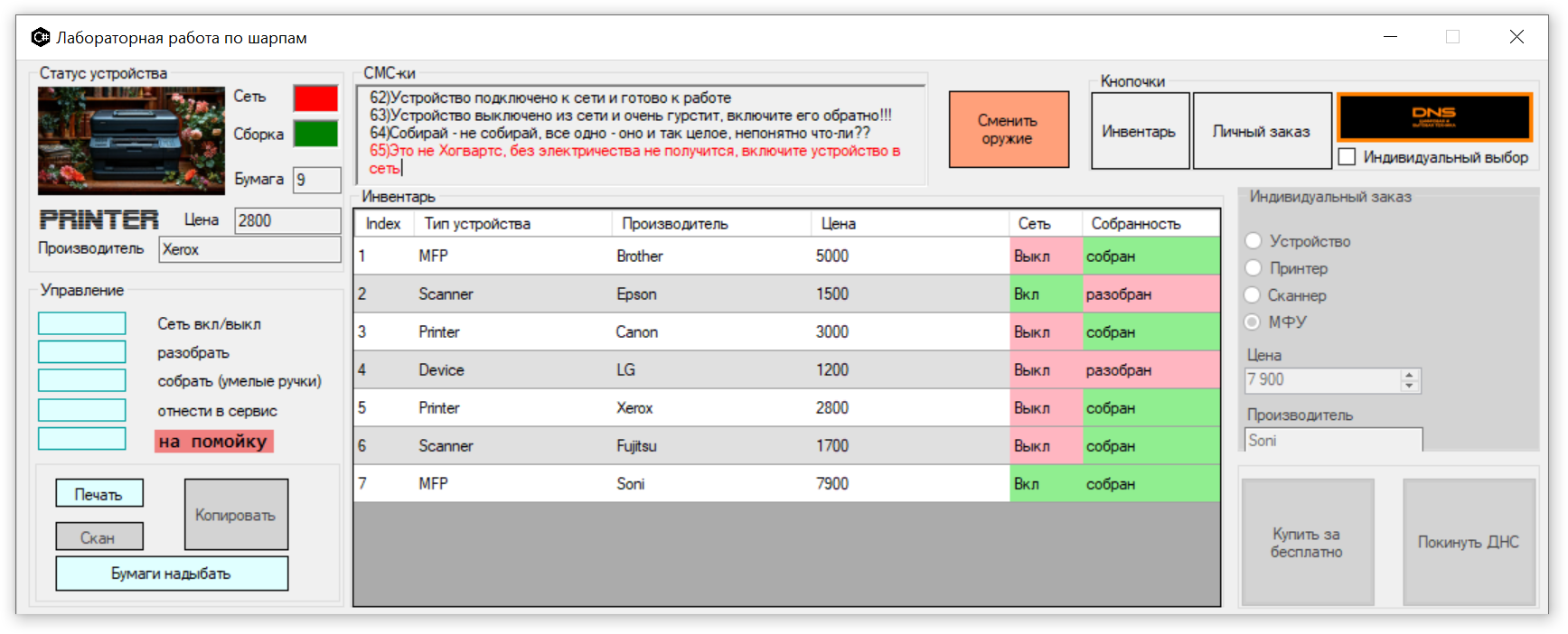
Программа представляет из себя иерархию классов с абстрактным классом в вершине, регулируемая привязанным к классам интерфейсам. Необходима разработка иерархии в соответствии с основными правилами ООП, и привязка выполнения методов к пользовательскому интерфейсу.

Взаимодействие с объектами класса происходит в модели «магазина товаров» с возможностью преобретения и дальнейшего использования:

1. В магазине пользователю предлагается на выбор некоторое количество устройств, из которых он выбирает себе подходящие. Для товаров существуют ограничения: по цене (товаря не могут быть дешевле 1000 рублей и дороже 10000р, а также их наименование не может быть длиннее 100 символов, иначе у покупателя голова закружится. Выйдя из магазина, пользователь может просмотреть свой инвентарь и техническое состояние всех устройств.
2. Из «купленных» товаров пользователь выбирает «текущее устройство», с которым может производить различные манипуляции:
   1. просмотр текущих параметров устройства (например включен он в розетку или нет).
3. Общие для всех устройств действия:
   1. включение/выключение устройства.
   2. разборка устройства.
   3. сборка устройства: здесь вводится элемент случайности: с разной вероятностью пользователь либо может, либо не может собрать устройство самостоятельно.
   4. починка устройства в сервисе.
   5. избавление от девайса: можно удалить устройство своего инвентаря.
4. Другие узкоспециализированные действия, при условии того, что устройство подключено к сети и находится в собранном состоянии:
   1. печать для принтера и МФУ: пользователь должен быть внимателен, потому что при каждой печати в устройстве используется бумага, и если воспользоваться печатью с пустым лотком для бумаги, прибор может сломаться.
   2. сканирование для сканера и МФУ.
   3. сканирование и копирование для МФУ.
5. **Разработка пользовательского интерфейса**

Взаимодействие программы с пользователем происходит в окне программы. Да, мы апнулись настолько, что перестали работать в консоли и узнали о существовании gui.

Общее меню устройства:



Кнопочки:

1. ДНС – вывод данных файла в таблицу для предоставления пользователю выбор для покупки. При включенной галочке индивидуального выбора появляется возможность выбрать файл для загрузки данных.
2. Личный заказ позволяет пользователю сотворить из ничего новое устройство с вымышленным названием и ценой. Просто немыслимо.
3. Покупка осуществляется двойным кликом по таблице или нажатием кнопки Купить за бесплатно.
4. Чтобы выйти в меню инвентаря пользователя требуется нажать кнопку Инвентарь, будучи в магазине можно воспользоватсься кнопкой покинуть ДНС.
5. Слева расположены кнопочки для манипуляций выбранным объектом для мучений.
6. **Реализация и тестирование приложения**
7. **Описание разработанной программы**

Программа реализована с использованием языка программирования С#. Код программы содержится в следующих файлах:

* Interfaces.cs, Device\_interface.cs, Sourse.cs – файлы хранения данных класса и его вспомогательных методов и классов.
* MainForm.cs, MainForm\_E.cs, MainForm\_F.cs – файлы для реализации интерфейса класса и обработки событий в пространстве WindowsForms

1. **Тестирование программы**

Табл. 3.1 Функциональные тесты

| **№ теста** | **Элементы формы** | **Входные данные** | **Ожидаемый результат** | **Смысл теста** |
| --- | --- | --- | --- | --- |
| Пользовательский интерфейс (ввод данных с клавиатуры) | | | | |
| 1 | Текстовое поле производитель | ввод пустой строки | сообщение об ошибке с просьбой исправить данные | логика работы поля ввода |
| ввод слишком длинной строки (больше 15 символов) | сообщение об ошибке |
| 2 | Поле цена | ввод букв | ничего, поле не должно принимать ничего кроме целых чисел | логика работы поля ввода |
| ввод числа меньше 1000 | автоисправление поля на 1000 |
| ввод числа больше 10000 | автоисправление поля на 10000 |
| Файловые взаимодействия | | | | |
| 3 | Кнопка «ДНС» | выбор пустого файла | сообщение об ошибке | логика работы файлового считывания |
| выбор нетекстового файла |
| выбор файла размером больше 2Мб |
| отмена выбора файла |
| ошибка данных в файле |
| Имитация логики работы устройства | | | | |
| 4 | Кнопки «печать», «сканировать», «копировать» | попытка воспользоваться устройством, выключенном из сети | сообщение об ошибке | проверка правильности работы логики работы объекта «устройство» |
| попытка воспользоваться устройством, выключенном из сети |
| попытка воспользоваться разобранным устройством |
| попытка воспользоваться устройством, выключенном из сети, к тому же разобранным |

**Вывод:** самое сложное в написании программ – это оформление отчета и специализации.

Но помимо этого было непросто понять и реализовать правильный интерфейс для полиморфных превращений устройств в их наследников: в данном случае явные приведения типов происходят не только через проверку совместимости с классом, но и с интерфейсом, так как через класс происходит наследование переменных и конструкторов, а функции наследуются через интерфейсы.

Так же было непросто понять принцип организации работы событий и делегатов, особенно сложно было перенести реализацию обработки событий в другой класс: MainForm, предоставив также доступ к переменным формы, не нарушая принципов инкапсуляции. Ну или мне так показалось, что у меня получилось их не нарушить.

Пора на стажировку, за новыми познаниями в области навыков программирования.

**Приложение**

Листинг программы Interfaces.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LR7\_LastOne

{

interface IDevice

{

int Price { get; }

string Manufacturer { get; }

bool IsPluggedIn { get; }

bool IsAssembled { get; }

void SwitchPlug();

void Assemble();

void DisassambleSam();

void Disassamble\_shop();

}

interface IPrinter: IDevice

{

int Papercount { get; }

void Print(int paper\_used = 1);

void PaperAdd(int paper = 1);

}

interface IScanner: IDevice

{

void Scan();

}

interface I\_MFP: IScanner, IPrinter

{

void Copy(int paper\_used = 1);

}

}

Листинг программы Device\_interface.cs

using System;

using System.Drawing;

using System.IO;

namespace LR7\_LastOne

{

delegate void HandleWriter(DeviceEventArgs e);

delegate void DeviceHandler(Device\_interface sender, DeviceEventArgs e);//??несогласованность по доступности

//назначение класса абстрактным чтобы не было соблазна создавать его экземпляров

abstract class Device\_interface : IDevice

{

//переменные

protected HandleDeviceEvents Notify;

protected HandleDeviceEvents ErrMsg;

protected int price;

protected string manufacturer;

protected bool isPluggedIn;

protected bool isAssembled;

static public readonly int MINPRICE = 1000;

static public readonly int MAXPRICE = 10000;

static public readonly byte MAXMANUFACTURERNAMELENGTH = 15;

//свойства

virtual public int Price

{

get => price;

private set

{

price = value;//criterror не позволит создаться объекту. по идее. нужно передать в обработчик событий значение не принятой цены

if (value < MINPRICE)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.CritErrorType.ErrPrice\_Less\_Then\_Min));

else if (value > MAXPRICE)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.CritErrorType.ErrPrice\_More\_Then\_Max));

}

}

virtual public string Manufacturer

{

get => manufacturer;

private set

{

if(value.Length == 0)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.CritErrorType.ErrManufName\_Empty));

manufacturer = value;

if (value.Length > MAXMANUFACTURERNAMELENGTH)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.CritErrorType.ErrManufName\_TooLong));

}

}

virtual public bool IsPluggedIn { get => isPluggedIn; set => isPluggedIn = value; }

virtual public bool IsAssembled { get => isAssembled; set => isAssembled = value; }

//конструкторы

protected Device\_interface(int price, string manufacturer)

{

ErrMsg = Notify = new HandleDeviceEvents();

Manufacturer = manufacturer;

isPluggedIn = isAssembled = false;

Price = price;

}

protected Device\_interface(int price, string manufacturer, HandleDeviceEvents notify, HandleDeviceEvents errMsg = null)

{

if (errMsg == null)

ErrMsg = notify;

else

ErrMsg = errMsg;

Notify = notify;

Manufacturer = manufacturer;

isPluggedIn = isAssembled = false;

Price = price;

}

//методы работают исключительно через события или нет?

virtual public void SwitchPlug()

{

IsPluggedIn = !IsPluggedIn;

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.Plug\_changed));

}

virtual public void Assemble()

{

if (IsAssembled)

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.Assembled,null, IsAssembled));

else

{

IsAssembled = true;

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.Assembled));

}

}

virtual public void DisassambleSam()

{

bool prev = IsAssembled;

IsAssembled = false;

if (!prev)

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.DisassSam, null, prev));

else

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.DisassSam, null, prev));

}

virtual public void Disassamble\_shop()

{

bool prev = IsAssembled;

IsAssembled = false;

if (!prev)

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.DisassShop, null, prev));

else

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.DisassShop, null, prev));

}

virtual public void ThrowDevice()

{

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.ThrowDev));

}

protected bool UsageCheck()

{

if (IsAssembled)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.ErrorType.ErrUsingAsembled));

else if (!IsPluggedIn)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.ErrorType.ErrUsingUnPlug));

return !IsAssembled && IsPluggedIn;

}

//класс обработчик событий

public class HandleDeviceEvents

{

private event DeviceHandler Notify;

private event HandleWriter Msg;

protected void RegisterWriter(HandleWriter msg) { Msg += msg; }

protected void UnRegisterWriter(HandleWriter msg) { Msg -= msg; }

protected void UnRegisterWriter() => Msg = null;

protected void RegisterHandler(DeviceHandler notify) { Notify += notify; }

protected void UnRegisterHandler(DeviceHandler notify) { Notify -= notify; }

protected void UnRegisterHandler() => Notify = null;

public void RaiseLogEvent(Device\_interface sender, DeviceEventArgs e)

{

if (Notify == null)

throw new ArgumentException("Error: Oбработчик событий отсутствует");

Notify?.Invoke(sender, e);

}

public void ThrowMassage(DeviceEventArgs msg)

{

if (Msg == null)

throw new ArgumentException("Error: Оператор вывода ошибок отсутствует");

Msg?.Invoke(msg);

}

public HandleDeviceEvents()

{

RegisterHandler(HandleMassages);

RegisterHandler(HandleProperties);

RegisterHandler(HandleCritErrors);

RegisterHandler(HandleErrors);

RegisterWriter(Console);

}

public static void Console(DeviceEventArgs msg)

{

System.Console.WriteLine(msg);

}

protected void HandleMassages(Device\_interface device, DeviceEventArgs e)

{

if (e.message != null)

ThrowMassage(e);

}

protected void HandleProperties(Device\_interface device, DeviceEventArgs e)

{

switch (e.property)

{

case null:

break;

case DeviceEventArgs.PropertyType.Assembled:

{

ThrowMassage(new DeviceEventArgs( $"Assemble changed of {device.ToString()}"));

}

break;

case DeviceEventArgs.PropertyType.Plug\_changed:

{

ThrowMassage(new DeviceEventArgs( "Plug changed of {device.ToString()}"));

}

break;

case DeviceEventArgs.PropertyType.StartCopying:

{

ThrowMassage(new DeviceEventArgs( "{device.ToString()} is copying"));

}

break;

case DeviceEventArgs.PropertyType.Printing:

{

ThrowMassage(new DeviceEventArgs( "{device.ToString()} is printing"));

}

break;

case DeviceEventArgs.PropertyType.Scanning:

{

ThrowMassage(new DeviceEventArgs( "{device.ToString()} is scanning"));

}

break;

default:

ThrowMassage(new DeviceEventArgs( "Мы не знаем, что произошло в отделе исполнений, но помощь уже в пути"));

break;

}

}

protected void HandleCritErrors(Device\_interface device, DeviceEventArgs e)

{

switch (e.criterror)

{

case null:

break;

case DeviceEventArgs.CritErrorType.ErrPrice\_Less\_Then\_Min:

{

throw new ArgumentException(string.Format("{0} {1} can't have so little price ({2}); Minimum: {3}", device.ToString(), device.Manufacturer, device.Price, Device\_interface.MINPRICE));

}

case DeviceEventArgs.CritErrorType.ErrPrice\_More\_Then\_Max:

{

throw new ArgumentException(string.Format("{0} {1} can't have so high price ({2}); Maximum: {3}", device.ToString(), device.Manufacturer, device.Price, Device\_interface.MAXPRICE));

}

case DeviceEventArgs.CritErrorType.ErrManufName\_TooLong:

{

throw new ArgumentException(string.Format("{0} can't have so long manufacturer's name: {1}; Max length: {2}", device.ToString(), device.Manufacturer, Device\_interface.MAXMANUFACTURERNAMELENGTH));

}

case DeviceEventArgs.CritErrorType.ErrManufName\_Empty:

{

throw new ArgumentException(string.Format("Can't accept empty manufacturers name for {0}", device.ToString()));

}

default:

ThrowMassage(new DeviceEventArgs( "Мы не знаем, что произошло в отделе критических ошибок, но помощь уже в пути"));

break;

}

}

protected void HandleErrors(Device\_interface device, DeviceEventArgs e)

{

switch (e.error)

{

case null:

break;

case DeviceEventArgs.ErrorType.ErrPaperEnd:

{

ThrowMassage(new DeviceEventArgs(string.Format("{0} потерял бумагу и печатать не будет", device.ToString())));

}

break;

case DeviceEventArgs.ErrorType.ErrPaperTooMuch:

{

ThrowMassage(new DeviceEventArgs(string.Format("{0} Больше не может скушать в себя так много бумаги", device.ToString())));

}

break;

case DeviceEventArgs.ErrorType.ErrUsingAsembled:

{

ThrowMassage(new DeviceEventArgs(string.Format("Пользоваться разобранным устройством можно, но лучше сначала собрать")));

}

break;

case DeviceEventArgs.ErrorType.ErrUsingUnPlug:

{

ThrowMassage(new DeviceEventArgs(string.Format("Это не Хогвартс, без электричества не получится, включите устройство в сеть")));

}

break;

default:

ThrowMassage(new DeviceEventArgs("Мы не знаем, что произошло в отделе ошибок, но помощь уже в пути"));

break;

}

}

}

}

class Device : Device\_interface

{

private static string ClassName = "Device";

public Device(int price, string manufacturer) : base(price, manufacturer) { }

public Device(int price, string manufacturer, HandleDeviceEvents notify, HandleDeviceEvents errMsg = null) : base(price, manufacturer, notify, errMsg) { }

public override string ToString() { return ClassName; }

public static string GetClass() { return ClassName; }

}

class Printer : Device, IPrinter

{

public readonly int MAXPAPERCOUNT = 10;

private static string ClassName = "Printer";

public Printer(int price, string manufacturer, int papercount = 10) : base(price, manufacturer)

{

Papercount = papercount;

}

public Printer(int price, string manufacturer, HandleDeviceEvents notify, HandleDeviceEvents errMsg = null, int papercount = 10) : base(price, manufacturer, notify, errMsg)

{

Papercount = papercount;

}

protected int papercount;

public int Papercount

{

get => papercount;

private set

{

if (value < 0)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.ErrorType.ErrPaperEnd));

else if (value > MAXPAPERCOUNT)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.ErrorType.ErrPaperTooMuch));

else

papercount = value;

}

}

public void Print(int paper\_used = 1)

{

if (UsageCheck())

{

int value = papercount - paper\_used;

Papercount = value;

if (Papercount == value)

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.Printing));

}

}

public void PaperAdd(int paperadd = 1)

{

int value = papercount + paperadd;

Papercount = value;

if (Papercount == value)

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.PaperCash));

}

public override string ToString() { return ClassName; }

public static string GetClass() { return ClassName; }

}

class Scanner : Device, IScanner

{

private static string ClassName = "Scanner";

public Scanner(int price, string manufacturer) : base(price, manufacturer) { }

public Scanner(int price, string manufacturer, HandleDeviceEvents notify, HandleDeviceEvents errMsg = null) : base(price, manufacturer, notify, errMsg) { }

virtual public void Scan()

{

if (UsageCheck())

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.Scanning));

}

public override string ToString() { return ClassName; }

public static string GetClass() { return ClassName; }

}

class MFP : Device, I\_MFP

{

private static string ClassName = "MFP";

public readonly int MAXPAPERCOUNT = 11;

public MFP(int price, string manufacturer, int papercount = 10) : base(price, manufacturer)

{

Papercount = papercount;

}

public MFP(int price, string manufacturer, HandleDeviceEvents notify, HandleDeviceEvents errMsg = null, int papercount = 10) : base(price, manufacturer, notify, errMsg)

{

Papercount = papercount;

}

int papercount;

public int Papercount

{

get => papercount;

private set

{

if (value < 0)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.ErrorType.ErrPaperEnd));

else if (value > MAXPAPERCOUNT)

ErrMsg.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.ErrorType.ErrPaperTooMuch));

else

papercount = value;

}

}

public void Copy(int paper\_used = 1)

{

if (UsageCheck())

{

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.StartCopying));

Scan();

Print(paper\_used);

}

}

public void Scan()

{

if (UsageCheck())

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.Scanning));

}

//не понимаю где реализовать логику изменения количества - в событии или в основном коде

public void Print(int paper\_used = 1)

{

if (UsageCheck())

{

int value = papercount - paper\_used;

Papercount = value;

if (Papercount == value)

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.Printing));

}

}

public void PaperAdd(int paperadd = 1)

{

int value = papercount + paperadd;

Papercount = value;

if (Papercount == value)

Notify.RaiseLogEvent(this, new DeviceEventArgs(DeviceEventArgs.PropertyType.PaperCash));

}

public override string ToString() { return ClassName; }

public static string GetClass() { return ClassName; }

}

}

Листинг программы Sourse.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static LR7\_LastOne.Device\_interface;

namespace LR7\_LastOne

{

class FileReader

{

public static bool FileOpen(string filepath, long maxSizeBytes)

{

FileInfo fileInfo = new FileInfo(filepath);

if (!fileInfo.Exists)

{

throw new FileNotFoundException($"File {filepath} doesn't exists");

}

else if (fileInfo.Length >= maxSizeBytes)

{

throw new FileNotFoundException($"File {filepath} is too big for this program. File size: {fileInfo.Length}; Max: {maxSizeBytes}");

}

return true;

}

public static Device[] GetDevices(Device\_interface.HandleDeviceEvents notify, Device\_interface.HandleDeviceEvents errMsg = null, string path = "data.txt")

{

if (!FileOpen(path, 2 \* 1024 \* 1024))

return null;

if (errMsg == null)

errMsg = notify;

string[] lines = File.ReadAllLines(path); // Чтение всех строк файла

Device[] result\_file = new Device[lines.Length];

int price;

string manufacturer;

int j = 0;

for (int i = 0; i < lines.Length; ++i, ++j)

{

string[] line = lines[i].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (int.TryParse(line[1], out price) && line[2].Length != 0)

{

manufacturer = line[2];

try

{

if (line[0] == Device.GetClass())

{

result\_file[j] = new Device(price, manufacturer, notify, errMsg);

}

else if (line[0] == Printer.GetClass())

{

result\_file[j] = new Printer(price, manufacturer, notify, errMsg);

}

else if (line[0] == Scanner.GetClass())

{

result\_file[j] = new Scanner(price, manufacturer, notify, errMsg);

}

else if (line[0] == MFP.GetClass())

{

result\_file[j] = new MFP(price, manufacturer, notify, errMsg);

}

else

{

notify.ThrowMassage(new DeviceEventArgs($"FileError: Неверный формат строки: {lines[i]}"));

}

}

catch (ArgumentException ex)

{

result\_file[j] = null;

errMsg.ThrowMassage(new DeviceEventArgs("FileError: " + ex.Message));

}

}

if (result\_file[j] == null) --j;

}

int elem\_count = j;

Device[] result = new Device[elem\_count];

for (int i = 0; i < elem\_count; ++i)

{

result[i] = result\_file[i];

}

return result;

}

}

public class DeviceEventArgs//несогласованность по доступности

{

public ErrorType? error { get; private set; }

public PropertyType? property { get; private set; }

public CritErrorType? criterror { get; private set; }

public string message { get; private set; }

public bool? status { get; private set; }

public DeviceEventArgs(string message = null, bool? status = null)

{

property = null;

error = null;

criterror = null;

this.message = message;

this.status = status;

}

public DeviceEventArgs(PropertyType notify, string message = null, bool? status = null) : this(message, status) { property = notify; }

public DeviceEventArgs(ErrorType notify, string message = null, bool? status = null) : this(message, status) { error = notify; }

public DeviceEventArgs(CritErrorType notify, string message = null, bool? status = null) : this(message, status) { criterror = notify; }

public override string ToString()

{

return message != null ? message : "no messages";

}

public enum CritErrorType

{

ErrPrice\_Less\_Then\_Min,

ErrPrice\_More\_Then\_Max,

ErrManufName\_TooLong,

ErrManufName\_Empty

}

public enum ErrorType

{

ErrPaperTooMuch,

ErrPaperEnd,

ErrUsingAsembled,

ErrUsingUnPlug

}

public enum PropertyType

{

Plug\_changed,

Assembled,

ThrowDev,

DisassShop,

DisassSam,

Printing,

Scanning,

StartCopying,

PaperCash

}

}

}

Листинг программы MainForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace LR7\_LastOne

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

this.Icon = new Icon("c.ico");

this.Text = "Лабораторная работа по шарпам";

}

private void MainForm\_Load(object sender, EventArgs e)

{

FormLoad();

this.Width = 1150;

this.Height = 452;

nudPrice\_OtBaldy.Minimum = Device\_interface.MINPRICE;

nudPrice\_OtBaldy.Maximum = Device\_interface.MAXPRICE;

tbManufact\_OtBaldy.MaxLength = Device\_interface.MAXMANUFACTURERNAMELENGTH;

FormStatusArg.UserStat = FormEventArgs.Status.Inventary;

}

//батоны все здесь

private void bEnterDNS\_Click(object sender, EventArgs e)

{

FormStatusArg.UserStat = FormEventArgs.Status.ShoppingDNS;

}

private void bBuy\_OtBaldy\_Click(object sender, EventArgs e)

{

FormStatusArg.UserStat = FormEventArgs.Status.Shopping\_OtBaldy;

}

private void bShowInventory\_Click(object sender, EventArgs e)

{

FormStatusArg.UserStat = FormEventArgs.Status.Inventary;

}

private void bSwitchMainDevice\_Click(object sender, EventArgs e)

{

FormStatusArg.UserProp = FormEventArgs.UserProperties.MainSwitch;

}

private void bBuyForFree\_Click(object sender, EventArgs e)

{

FormStatusArg.UserProp = FormEventArgs.UserProperties.BuyDevice;

}

private void bLeaveDNS\_Click(object sender, EventArgs e)

{

FormStatusArg.UserStat = FormEventArgs.Status.Inventary;

}

private void bPlugSwitch\_Click(object sender, EventArgs e)

{

try

{

if (MainDev == null)

{

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

return;

}

else

MainDev.SwitchPlug();

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs(string.Format(ex.Message)));

}

}

private void bAssem\_Click(object sender, EventArgs e)

{

try

{

if (MainDev == null)

{

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

return;

}

else

MainDev.Assemble();

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs(string.Format(ex.Message)));

}

}

private void bDisassSam\_Click(object sender, EventArgs e)

{

try

{

if (MainDev == null)

{

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

return;

}

else

MainDev.DisassambleSam();

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs(string.Format(ex.Message)));

}

}

private void bDisassShop\_Click(object sender, EventArgs e)

{

try

{

if (MainDev == null)

{

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

return;

}

else

MainDev.Disassamble\_shop();

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs(string.Format(ex.Message)));

}

}

private void bThrow\_Click(object sender, EventArgs e)

{

try

{

if (MainDev == null)

{

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

return;

}

else

MainDev.ThrowDevice();

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs(string.Format(ex.Message)));

}

}

private void bScan\_Click(object sender, EventArgs e)

{

try

{

if (MainDev == null)

{

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

return;

}

else

if (MainDev is IScanner)

((IScanner)MainDev).Scan();//допускаем ведь есть try и логика кнопок

else

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrOptionalLost));

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs(string.Format(ex.Message)));

}

}

private void bPrint\_Click(object sender, EventArgs e)

{

try

{

if (MainDev == null)

{

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

return;

}

else

if (MainDev is IPrinter)

((IPrinter)MainDev).Print();//допускаем ведь есть try и логика кнопок

else

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrOptionalLost));

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs(string.Format(ex.Message)));

}

}

private void bCopy\_Click(object sender, EventArgs e)

{

try

{

if (MainDev == null)

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

else

{

if (MainDev is I\_MFP mfp)

((I\_MFP)MainDev).Copy();//допускаем ведь есть try и логика кнопок

else

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrOptionalLost));

}

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs(string.Format(ex.Message)));

}

}

private void bPaperCash\_Click(object sender, EventArgs e)

{

try

{

if (MainDev == null)

{

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

return;

}

else

{

if (MainDev is IPrinter printer)

((IPrinter)MainDev).PaperAdd();//допускаем ведь есть try и логика кнопок

else

ErrNotify.RaiseLogEvent(MainDev, new UserEventArgs(UserEventArgs.UserErrors.ErrOptionalLost));

}

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs(string.Format(ex.Message)));

}

}

private void dgvListOfDevices\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0 && e.ColumnIndex >= 0)

FormStatusArg.UserProp = FormEventArgs.UserProperties.BuyDevice;

}

private void label12\_Click(object sender, EventArgs e)

{

bThrow\_Click(sender, e);

}

}

}

Листинг программы MainForm\_E.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Resources;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LR7\_LastOne

{

public partial class MainForm : Form

{

class HandleDeviceErrorsInWF : Device\_interface.HandleDeviceEvents

{

public HandleDeviceErrorsInWF() : base()

{

UnRegisterHandler(HandleProperties);

RegisterHandler(HandleUserErrors);

}

public HandleDeviceErrorsInWF(HandleWriter msg) : this()

{

//UnRegisterWriter();

RegisterWriter(msg);

}

protected void HandleUserErrors(Device\_interface device, DeviceEventArgs e)

{

if (e is UserEventArgs)

switch (((UserEventArgs)e).userErr)

{

case null:

break;

case UserEventArgs.UserErrors.ErrMainDevEmpty:

{

ThrowMassage(new DeviceEventArgs( "Главное устройство не выбрано, незачем так тыкать"));

}

break;

case UserEventArgs.UserErrors.ErrOptionalLost:

{

ThrowMassage(new DeviceEventArgs( "Устройство не так многофункционально как кажется. Эту операцию я не могу вам позволить"));

}

break;

default:

ThrowMassage(new DeviceEventArgs( "Что-то наследование не полиморфируется, надо разбаговывать"));

break;

}

}

}

public class UserEventArgs : DeviceEventArgs

{

public UserErrors? userErr;

public UserEventArgs(string message = null, bool? status = null) : base(message, status) { userErr = null; }

public UserEventArgs(UserErrors usererr, string message = null, bool? status = null) : this(message, status) { userErr = usererr; }

public enum UserErrors

{

ErrMainDevEmpty,

ErrOptionalLost,

}

}

public class FormEventArgs : UserEventArgs

{

public delegate void FormStatusHandler(DeviceEventArgs msg);

public event FormStatusHandler StatusHandler;

private Status? userStat;

private UserProperties? userProp;

public UserProperties? UserProp { get => userProp; set { userProp = value; StatusHandler(this); } }

public Status? UserStat { get => userStat; set { Clear(); userStat = value; StatusHandler(this); } }

public FormEventArgs(FormStatusHandler notify) : base()

{

StatusHandler = notify;

UserStat = null;

UserProp = null;

}

public void RegisterEvent(FormStatusHandler notify)

{

StatusHandler = notify;

}

public void Clear()

{

userProp = null;

}

public enum UserProperties

{

BuyDevice,

MainSwitch

}

public enum Status

{

ShoppingDNS,

Shopping\_OtBaldy,

Inventary

}

}

class HandleDeviceEventsProperties : HandleDeviceErrorsInWF//не могу регистрировать обработчик событий вне класса

{

protected MainForm form;

protected Random rnd;

public HandleDeviceEventsProperties(MainForm form) : base()

{

rnd = new Random();

this.form = form;

UnRegisterHandler();

RegisterHandler(HandleProperties);

}

public HandleDeviceEventsProperties(MainForm form, HandleWriter msg) : this(form)

{

UnRegisterWriter();

RegisterWriter(msg);

}

new protected async void HandleProperties(Device\_interface device, DeviceEventArgs e)

{

try

{

if (device == null)

{

HandleUserErrors(device, new UserEventArgs(UserEventArgs.UserErrors.ErrMainDevEmpty));

return;

}

switch (e.property)

{

case null:

break;

case UserEventArgs.PropertyType.Plug\_changed:

{

if (device.IsPluggedIn)

{

if (!device.IsAssembled)

{

if (rnd.Next(0, 100) < 20)

{

form.Notify.ThrowMassage(new DeviceEventArgs("Устройство включено в сеть (электрическую)"));

form.Notify.ThrowMassage(new DeviceEventArgs("Начинается прогрев картриджей. БЖ"));

form.cts = new System.Threading.CancellationTokenSource();

for (int i = 0; i < 3; ++i)

{

await Task.Delay(400, form.cts.Token);

form.Notify.ThrowMassage(new DeviceEventArgs("-Ж", false));

}

form.Notify.ThrowMassage(new DeviceEventArgs("ШЬМЯК! Вы прихлопнули муху и больше ничего не жужжит", false));

}

else

form.Notify.ThrowMassage(new DeviceEventArgs( "Устройство подключено к сети и готово к работе"));

}

else

{

form.Notify.ThrowMassage(new DeviceEventArgs("Прежде чем тыкать вилку в розетку, надо бы собраться с силами и собрать устройство"));

device.IsPluggedIn = false;

}

}

else

{

form.Notify.ThrowMassage(new DeviceEventArgs("Устройство выключено из сети и очень гурстит, включите его обратно!!!"));

form.cts?.Cancel();

}

}

break;

case UserEventArgs.PropertyType.Assembled:

{

if (device.IsPluggedIn)

{

form.Notify.ThrowMassage(new DeviceEventArgs("Допустим, вы такой смелый, что разобрали устройство, пока оно было под напряжением. Может пора вытащить вилку из розетки, чего она одна болтается?"));

}

else

if (e.status == true)

switch (rnd.Next(1, 3))

{

case 1: form.Notify.ThrowMassage(new DeviceEventArgs("Если дальше разбирать, то только на атомы")); break;

case 2: form.Notify.ThrowMassage(new DeviceEventArgs("Внимание! дальнейшая разборка может привести к открытию новых физических законов")); break;

case 3: form.Notify.ThrowMassage(new DeviceEventArgs("Устройство больше не разбирается. Никак и ни в чем, даже в php")); break;

default: form.Notify.ThrowMassage(new DeviceEventArgs("Дальше разбираться нет смысла, это ведь не уроки программирования")); break;

}

else

form.Notify.ThrowMassage(new DeviceEventArgs("Устройство теперь разобрано. Все винтики в коробочке или как обычно?"));

form.panelAssembleStat.BackColor = device.IsAssembled ? Color.Red : Color.Green;

form.cts?.Cancel();

}

break;

case UserEventArgs.PropertyType.DisassSam:

{

if (device.IsPluggedIn)

form.ErrNotify.ThrowMassage(new DeviceEventArgs("Под напряжением собирать устройство явно веселее, но минздрав реккомендует таким не заниматься"));

else

{

if (e.status == true)

{

if (rnd.Next(0, 100) < 70)//почти 50 на 50

{

device.IsAssembled = true;

form.Notify.ThrowMassage(new DeviceEventArgs("Ну тут было 50 на 50. Увы, мы проиграли, теперь одна дорога - в мастерскую"));

}

else

form.Notify.ThrowMassage(new DeviceEventArgs("Неважно с какой попытки, но удача обернулась нужным местом: мы самостоятельно собрали устройство. Не забыть бы лишние винтики в шкаф отнести к остальным"));

}

else

form.Notify.ThrowMassage(new DeviceEventArgs("Собирай - не собирай, все одно - оно и так целое, непонятно что-ли??"));

}

}

break;

case UserEventArgs.PropertyType.DisassShop:

{

try { form.cts?.Cancel(); }

finally

{

if (device.IsPluggedIn && e.status != false)

form.ErrNotify.ThrowMassage(new DeviceEventArgs( "Вы его разобрали, а оно все еще включено сеть? Магия какая-то"));

else

form.Notify.ThrowMassage(new DeviceEventArgs("В мастеркой не знают, как чинить это. Ладно, шутка. С вас 4К за ремонт, можно без сдачи"));

if (e.status == false)

form.Notify.ThrowMassage(new DeviceEventArgs("Кстати, устройство и так было собрано. Лучше бы не носить его ни куда"));

}

}

break;

case UserEventArgs.PropertyType.ThrowDev:

{

try { form.cts?.Cancel(); }

finally

{

if (form.DevInventary.Length == 1)

form.DevInventary = null;

else

{

Device[] result = form.DevInventary.Where(dev => dev != form.MainDev).ToArray();

form.DevInventary = result;

}

form.MainDev = null;

form.MainImageIndex = 0;

form.UpdateMainInfo();

form.MainFormEnabled();

switch (rnd.Next(0, 5))

{

case 0: form.Notify.ThrowMassage(new DeviceEventArgs( $"Нельзя же такое просто так выбрасывать. Лучше подарть {device.Manufacturer} Потапову на НГ")); break;

case 1: form.Notify.ThrowMassage(new DeviceEventArgs("Вы выкинули устройство в окно. Полиция уже едет за вами")); break;

case 2: form.Notify.ThrowMassage(new DeviceEventArgs( $"Вы положили {device.Manufacturer} на балкон. Оно потеряно навсегда")); break;

case 3: form.Notify.ThrowMassage(new DeviceEventArgs( $"Вы увезли {device.Manufacturer} на дачу. Туда ему и дорога, пусть паучки в нем живут")); break;

case 4: form.Notify.ThrowMassage(new DeviceEventArgs("Ну чтож, может авито кому-нибудь понравится. Попробуем продать за двойную цену, товар ведь импортный")); break;

default: form.Notify.ThrowMassage(new DeviceEventArgs("Вы просто выкинули устройство. Без шуток")); break;

}

}

}

break;

case UserEventArgs.PropertyType.StartCopying:

{

try { form.cts?.Cancel(); }

finally

{

form.Notify.ThrowMassage(new DeviceEventArgs("Копирование начинается"));

}

}

break;

case UserEventArgs.PropertyType.Printing:

{

try { form.cts?.Cancel(); }

finally

{

form.cts = new System.Threading.CancellationTokenSource();

form.Notify.ThrowMassage(new DeviceEventArgs("Печать в процессе в процессе"));

for (int i = 0; i < 5; ++i)

{

await Task.Delay(500, form.cts.Token);

form.Notify.ThrowMassage(new DeviceEventArgs(" . ", false));

}

form.Notify.ThrowMassage(new DeviceEventArgs("Печать завершена", false));

}

}

break;

case UserEventArgs.PropertyType.Scanning:

{

try { form.cts?.Cancel(); }

finally

{

form.cts = new System.Threading.CancellationTokenSource();

form.Notify.ThrowMassage(new DeviceEventArgs("Сканирование в процессе"));

form.Notify.ThrowMassage(new DeviceEventArgs("Сканирование завершено", false));

}

}

break;

case UserEventArgs.PropertyType.PaperCash:

{

form.Notify.ThrowMassage(new DeviceEventArgs("Вы нафармили 1 листок бумаги"));

}

break;

}

form.UpdateGrid();

form.UpdateMainInfo();

}

catch (OperationCanceledException)

{

form.ErrNotify.ThrowMassage(new DeviceEventArgs( "Ну зачем же так резко выключать устройства из сети? Они могут заболеть"));

}

}

}

void FormStatus(DeviceEventArgs Msg)

{

if (!(Msg is FormEventArgs msg))

{

ErrNotify.ThrowMassage(new DeviceEventArgs( "Проблемы с полиморфимическим наследованием, звоните в поддержку"));

return;

}

switch (msg.UserProp)

{

case FormEventArgs.UserProperties.BuyDevice:

{

switch (msg.UserStat)

{

case FormEventArgs.Status.ShoppingDNS:

{

if (dgvListOfDevices.CurrentCell != null)

{

int selectedRowIndex = dgvListOfDevices.CurrentCell.RowIndex;

string Type = dgvListOfDevices.Rows[selectedRowIndex].Cells["Type"].Value.ToString();

int price = Convert.ToInt32(dgvListOfDevices.Rows[selectedRowIndex].Cells["Price"].Value);

string manuf = dgvListOfDevices.Rows[selectedRowIndex].Cells["Manufacturer"].Value.ToString();

Device result = null;

try

{

if (Type == Device.GetClass())

{

result = new Device(price, manuf, Notify, ErrNotify);

}

else if (Type == Printer.GetClass())

{

result = new Printer(price, manuf, Notify, ErrNotify);

}

else if (Type == Scanner.GetClass())

{

result = new Scanner(price, manuf, Notify, ErrNotify);

}

else if (Type == MFP.GetClass())

{

result = new MFP(price, manuf, Notify, ErrNotify);

}

else

ErrNotify.ThrowMassage(new DeviceEventArgs( "Не нашли подходящий тип данных, что происходит?"));

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs( string.Format(ex.Message)));

}

AddDevice(result);

}

else

{

ErrNotify.ThrowMassage(new DeviceEventArgs( "Чтобы что-нибудь купить, надо что-нибудь выбрать"));

}

}

break;

case FormEventArgs.Status.Shopping\_OtBaldy:

{

Device result = CreateDevice();

if (result != null)

AddDevice(result);

dgvListOfDevices.ClearSelection();

}

break;

case FormEventArgs.Status.Inventary:

{

FormStatusArg.UserProp = FormEventArgs.UserProperties.MainSwitch;

}

break;

}

}

break;

case FormEventArgs.UserProperties.MainSwitch:

{

if (dgvListOfDevices.CurrentCell != null)

{

int index = dgvListOfDevices.CurrentCell.RowIndex;

try

{

MainDev = DevInventary[index];

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs( string.Format(ex.Message)));

}

MainImageIndex = rnd.Next(1, image.Length);

}

else

{

ErrNotify.ThrowMassage(new DeviceEventArgs( "Чтобы выбрать мейна, надо выбрать мейна (в таблице)"));

}

UpdateMainInfo();

MainFormEnabled();

}

break;

case null:

{

switch (msg.UserStat)

{

case null:

break;

case FormEventArgs.Status.ShoppingDNS:

{

dgvListOfDevices.Rows.Clear();

gbListOfDevices.Text = "М.Видео";

gbShop\_OtBaldy.Enabled = false; gbShop\_OtBaldy.BackColor = Color.LightGray;

DisableButtons(bMainDeviceSwitch);

DisableButtons(gbListOfDevices.Controls.OfType<Button>().ToArray());

EnableButtons(gbBuyLeave.Controls.OfType<Button>().ToArray());

try

{

Device[] dev = LoadDevicesFromFile();

if (dev != null)

{

LetsShopping(dev);

}

else

{

ErrNotify.ThrowMassage(new DeviceEventArgs( "Что-то пошло не так при походе в файловый ДНС"));

}

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs( string.Format(ex.Message)));

}

}

break;

case FormEventArgs.Status.Shopping\_OtBaldy:

{

dgvListOfDevices.Rows.Clear();

gbListOfDevices.Text = "Шоппинг на заказ";

cbDNS.Checked = false;

DisableButtons(bMainDeviceSwitch);

gbShop\_OtBaldy.Enabled = true; gbShop\_OtBaldy.BackColor = SystemColors.Control;

EnableButtons(gbListOfDevices.Controls.OfType<Button>().ToArray());

EnableButtons(gbBuyLeave.Controls.OfType<Button>().ToArray());

}

break;

case FormEventArgs.Status.Inventary:

{

dgvListOfDevices.Rows.Clear();

gbListOfDevices.Text = "Инвентарь";

cbDNS.Checked = false;

gbShop\_OtBaldy.Enabled = false; gbShop\_OtBaldy.BackColor = Color.LightGray;

DisableButtons(gbBuyLeave.Controls.OfType<Button>().ToArray());

EnableButtons(gbListOfDevices.Controls.OfType<Button>().ToArray());

UpdateGrid();

MainFormEnabled();

}

break;

default:

ErrNotify.ThrowMassage(new DeviceEventArgs( "Забыли обработать событие в отделе formstatus, но помощь уже в пути"));

break;

}

dgvListOfDevices.ClearSelection();

}

break;

default:

ErrNotify.ThrowMassage(new DeviceEventArgs( "Мы не знаем, что произошло в отделе userprop, но помощь уже в пути"));

break;

}

}

}

}

Листинг программы MainForm\_F.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Media;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Runtime.Remoting.Messaging;

namespace LR7\_LastOne

{

public partial class MainForm : Form

{

string filepath = "file.txt";//переменная для открывания файла с данными

int messagescount = 0;//счетчик сообщений

Device[] DevInventary;//список девайсов в инвентаре

Device MainDev;//текущий девайс

int MainImageIndex;

HandleDeviceEventsProperties Notify;//сообщения в черном цвете

HandleDeviceErrorsInWF ErrNotify;//смс об ошибках

FormEventArgs FormStatusArg;

Image[] image;

Random rnd;

CancellationTokenSource cts;

StringBuilder MsgB;

private void FormLoad()

{

MainDev = null;

DevInventary = null;

Notify = new HandleDeviceEventsProperties(this, MSGHandle);

ErrNotify = new HandleDeviceErrorsInWF(ErrorMSGHandle);

FormStatusArg = new FormEventArgs(FormStatus);

MainImageIndex = 0;

image = new Image[]

{

Image.FromFile("empty.jpg"),

Image.FromFile("printer1.jpg"),

Image.FromFile("printer2.jpg"),

Image.FromFile("printer3.jpg"),

Image.FromFile("printer4.jpg"),

Image.FromFile("printer5.jpg"),

Image.FromFile("printer6.jpg"),

Image.FromFile("printer7.jpg"),

Image.FromFile("printer8.jpg"),

Image.FromFile("printer9.jpg"),

Image.FromFile("printer10.jpg")

};

rnd = new Random();

MsgB = new StringBuilder(1000);

}

private void ShowImage(int index)

{

if (index > 0 && index < image.Length)

pictureBox1.Image = image[index];

else

pictureBox1.Image = image[0];

}

//обработчики сообщений

public void ErrorMSGHandle(DeviceEventArgs msg)

{

++messagescount;

MsgB.Clear();MsgB.Append(messagescount.ToString() + ")" + msg.ToString() + "\r\n");

rtbMessage.SelectionStart = rtbMessage.Text.Length;

rtbMessage.SelectionLength = 0;

rtbMessage.SelectionColor = Color.Red;

rtbMessage.AppendText(MsgB.ToString());

rtbMessage.SelectionStart = rtbMessage.Text.Length;

rtbMessage.ScrollToCaret();

bInventoryShow.Focus();

}

public void MSGHandle(DeviceEventArgs msg)

{

MsgB.Clear();

if (msg.status == null)

{

++messagescount;

MsgB.Append(messagescount.ToString() + ")" + msg.ToString() + "\r\n");

}

else

{

MsgB.Append(msg.ToString() + "\r\n");

}

rtbMessage.AppendText(MsgB.ToString());

rtbMessage.SelectionStart = rtbMessage.Text.Length;

rtbMessage.ScrollToCaret();

bInventoryShow.Focus();

}

//обновляторы формы

void UpdateGrid()

{

dgvListOfDevices.Rows.Clear();

if (FormStatusArg.UserStat == FormEventArgs.Status.Inventary)

if (DevInventary != null)

for (int i = 0; i < DevInventary.Length; ++i)

{

AddGridRow(DevInventary[i], i);

}

else

{

ErrNotify.ThrowMassage(new DeviceEventArgs( "Инвентарь пуст, надо заскочить в магазин"));

}

dgvListOfDevices.ClearSelection();

}

void UpdateMainInfo()

{

if (MainDev != null)

{

panelPlugStat.BackColor = MainDev.IsPluggedIn ? Color.Green : Color.Red;

panelAssembleStat.BackColor = !MainDev.IsAssembled ? Color.Green : Color.Red;

tbMainType.Text = MainDev.ToString();

tbMainPrice.Text = MainDev.Price.ToString();

tbMainManuf.Text = MainDev.Manufacturer;

tbPaperCount.Text = MainDev is IPrinter printer ? printer.Papercount.ToString() : "--";

}

else

{

panelAssembleStat.BackColor = Color.LightGray;

panelPlugStat.BackColor = Color.LightGray;

tbPaperCount.Text = "--";

tbMainType.Text = "пусто";

tbMainPrice.Text = "";

tbMainManuf.Text = "";

}

}

void MainFormEnabled()

{

if (dgvListOfDevices.Rows.Count == 0)

{

bMainDeviceSwitch.BackColor = Color.LightGray;

bMainDeviceSwitch.Enabled = false;

}

else

{

bMainDeviceSwitch.Enabled = true;

bMainDeviceSwitch.BackColor = Color.LightSalmon;

}

if (MainDev == null)

{

gbMainDevice.BackColor = Color.LightGray;

DisableButtons(gbManagerBut.Controls.OfType<Button>().ToArray());

DisableButtons(gbPrintScanCopy.Controls.OfType<Button>().ToArray());

}

else if (MainDev is MFP)

{

gbMainDevice.BackColor = SystemColors.Control;

EnableButtons(gbManagerBut.Controls.OfType<Button>().ToArray());

EnableButtons(gbPrintScanCopy.Controls.OfType<Button>().ToArray());

}

else if (MainDev is Scanner)

{

gbMainDevice.BackColor = SystemColors.Control;

EnableButtons(gbManagerBut.Controls.OfType<Button>().ToArray());

DisableButtons(gbPrintScanCopy.Controls.OfType<Button>().ToArray());

EnableButtons(bScan);

}

else if (MainDev is Printer)

{

gbMainDevice.BackColor = SystemColors.Control;

EnableButtons(gbManagerBut.Controls.OfType<Button>().ToArray());

DisableButtons(gbPrintScanCopy.Controls.OfType<Button>().ToArray());

EnableButtons(bPrint, bPaperCash);

}

else if (MainDev is Device)

{

gbMainDevice.BackColor = SystemColors.Control;

EnableButtons(gbManagerBut.Controls.OfType<Button>().ToArray());

DisableButtons(gbPrintScanCopy.Controls.OfType<Button>().ToArray());

}

ShowImage(MainImageIndex);

}

//оформление батонов

void DisableButtons(params Button[] buttons)

{

foreach (var button in buttons)

{

button.Enabled = false;

button.BackColor = Color.LightGray;

}

}

void EnableButtons(params Button[] buttons)

{

foreach (var button in buttons)

{

button.Enabled = true;

button.BackColor = Color.LightCyan;

}

}

//управление данными формы - составление новых объектов из файлов или таблиц

private void AddDevice(Device dev)

{

Device[] result = null;

if (DevInventary == null)

{

result = new Device[1];

result[0] = dev;

}

else

{

result = new Device[DevInventary.Length + 1];

for (int i = 0; i < DevInventary.Length; ++i)

{

result[i] = DevInventary[i];

}

result[result.Length - 1] = dev;

}

DevInventary = result;

}

private Device CreateDevice()

{

Device result = null;

try

{

if (rbDevice.Checked)

{

result = new Device((int)nudPrice\_OtBaldy.Value, tbManufact\_OtBaldy.Text, Notify, ErrNotify);

}

else if (rbPrinter.Checked)

{

result = new Printer((int)nudPrice\_OtBaldy.Value, tbManufact\_OtBaldy.Text, Notify, ErrNotify);

}

else if (rbScanner.Checked)

{

result = new Scanner((int)nudPrice\_OtBaldy.Value, tbManufact\_OtBaldy.Text, Notify, ErrNotify);

}

else if (rbMFP.Checked)

{

result = new MFP((int)nudPrice\_OtBaldy.Value, tbManufact\_OtBaldy.Text, Notify, ErrNotify);

}

else

ErrNotify.ThrowMassage(new DeviceEventArgs( "Как так получилось, что ничего не выбрано???"));

AddGridRow(result, dgvListOfDevices.Rows.Count);

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs( string.Format(ex.Message)));

}

return result;

}

private Device[] LoadDevicesFromFile()

{

try

{

Device[] result = null;

if (cbDNS.Checked)

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

// Открытие выбранного файла

filepath = openFileDialog1.FileName;

result = FileReader.GetDevices(Notify, ErrNotify, filepath);

}

}

else

{

result = FileReader.GetDevices(Notify, ErrNotify, filepath);

}

return result.Length != 0 ? result : null;

}

catch (ArgumentException ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs("FileError: " + ex.Message));

return null;

}

catch (FileNotFoundException ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs("FileError: " + ex.Message));

return null;

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs("FileError: " + ex.Message));

return null;

}

}

private int LetsShopping(Device[] device)

{

try

{

for (int i = 0; i < device.Length; ++i)

AddGridRow(device[i], i);

return 0;

}

catch (Exception ex)

{

ErrNotify.ThrowMassage(new DeviceEventArgs( string.Format(ex.Message)));

return 1;

}

}

//оформление таблиц

private void AddGridRow(Device device, int i)

{

Color plugcolor, assemblecolor;

dgvListOfDevices.Rows.Add(i + 1, device.ToString(), device.Manufacturer, device.Price, GetPlugStat(device), GetAssembleStat(device)); ;

plugcolor = device.IsPluggedIn ? Color.LightGreen : Color.LightPink;

assemblecolor = !device.IsAssembled ? Color.LightGreen : Color.LightPink;

dgvListOfDevices.Rows[i].Cells["PlugStatus"].Style.BackColor = plugcolor;

dgvListOfDevices.Rows[i].Cells["AssembleStatus"].Style.BackColor = assemblecolor;

}

private string GetPlugStat(Device dev) => dev.IsPluggedIn ? "Вкл" : "Выкл";

private string GetAssembleStat(Device dev) => dev.IsAssembled ? "разобран" : "собран";

}

}