

# Лабораторная работа № 1 Разработка ПО и фиксация этапов

## Цель:

Создание стабильной, функциональной и поддерживаемой прикладной программы, с использованием системы контроля версий Git для фиксации этапов разработки, обеспечения прозрачности изменений, упрощения сотрудничества в команде и автоматизации процессов сборки, тестирования и деплоя.

## Задание:

Создать прикладную программу(ПП) по теме варианта. При создании ПП должны быть следующие шаги фиксации в системе контроля версий:

1. Инициализирован проект.
2. Добавлен новый класс(ы).
3. Создан графический интерфейс с элементами управления.
4. Реализован дизайн формы. (опционально)
5. Версия программы.

Созданная программа должна храниться на удаленном репозитории.

## Пример:

В примере указаны шаги фиксации при выполнении задания на 3(Удовлетворительно), так как фиксации выполняются в основной ветке. Если вы выполняете на оценку выше у вас должны быть в удаленном репозитории 2 ветки (main и develop), а в локальном репозитории по необходимости(feature, release, hotFix).

**### Тема: Программа для управления товарами в магазине**

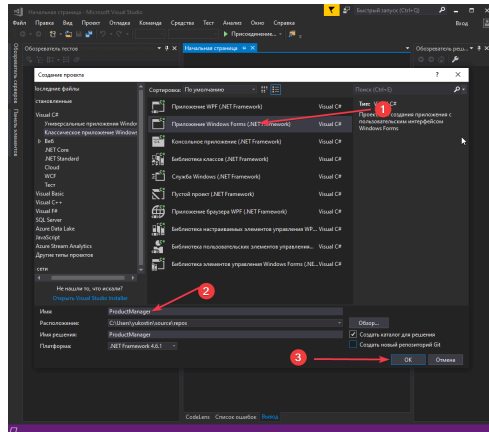
Программа должна предоставлять следующие функции:

1. Добавление товара.
2. Удаление товара.
3. Проверка наличия товара.
4. Отображение списка товаров.

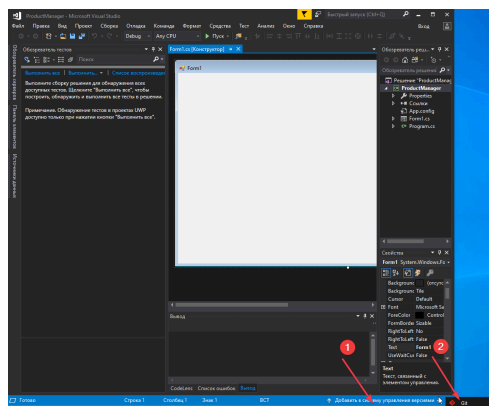
5. Проверка граничных значений (например, цена не может быть отрицательной, количество не может быть отрицательным и т.д.).

---

### ### Создание проекта



### ### Добавление и инициализация системы управления версиями



### ### Код программы:

#### ##### 1. Создание класса "Товар"

Создадим класс `Product.cs`, который будет содержать свойства и методы для работы с товаром.

```
```csharp
using System;

namespace ProductManager
{
    public class Product
    {
        public string Name { get; set; }
        public decimal Price { get; set; }
    }
}
```

```
public int Quantity { get; set; }
```

```
public Product(string name, decimal price, int quantity)
```

```
{
    Name = name;
    Price = price;
    Quantity = quantity;
}
```

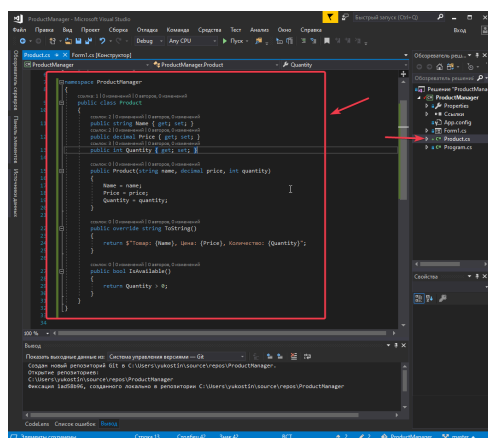
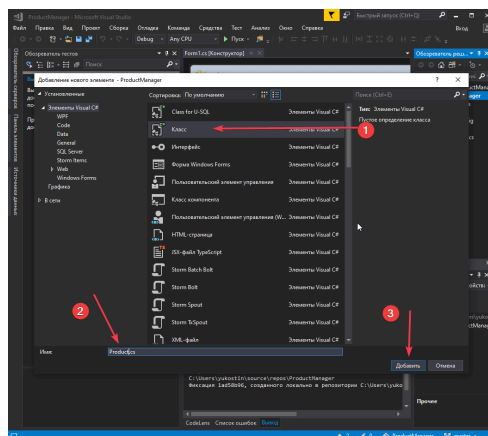
```
public override string ToString()
```

```
{
    return $"Товар: {Name}, Цена: {Price}, Количество: {Quantity}";
}
```

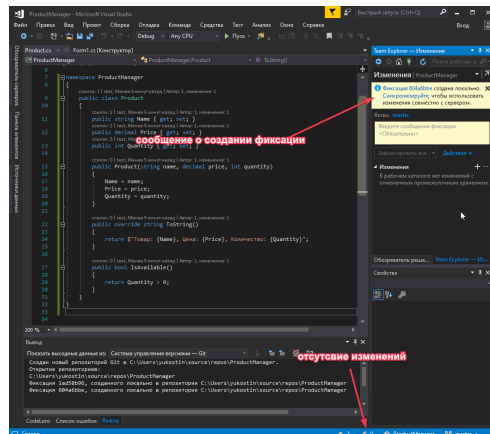
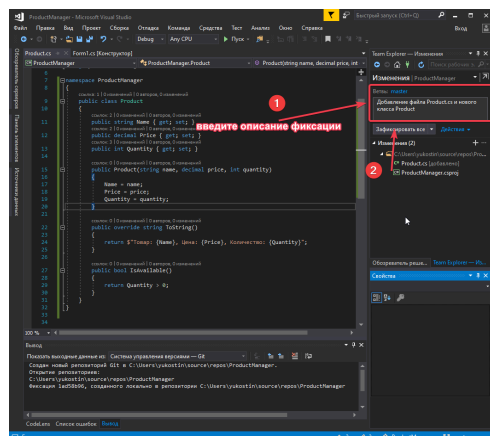
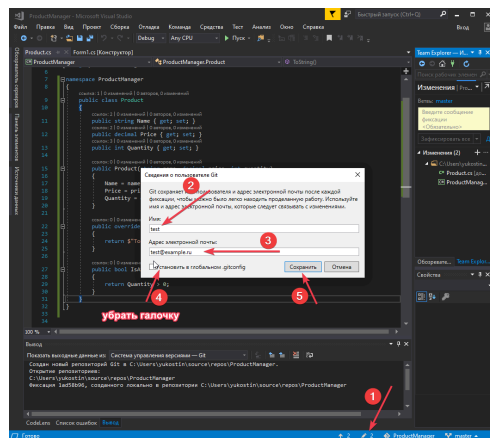
```
public bool IsAvailable()
```

```
{
    return Quantity > 0;
}
```

```
}
...
}
```



### Фиксация добавления нового класса



## #### 2. Создание графического интерфейса

Создадим форму `MainWindow.cs`, которая будет содержать элементы управления для ввода данных и отображения результата.

```

csharp
using System;
using System.Windows.Forms;

namespace ProductManager

```

```

{
    public partial class MainWindow : Form
    {
        private List<Product> products = new List<Product>();

        public MainWindow()
        {
            InitializeComponent();
        }

        private void addButton_Click(object sender, EventArgs e)
        {
            try
            {
                string name = nameTextBox.Text;
                decimal price = numericUpDown1.Value;
                int quantity = (int)numericUpDown2.Value;

                if (price <= 0)
                {
                    MessageBox.Show("Цена должна быть положительной!", "Ошибка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }

                if (quantity < 0)
                {
                    MessageBox.Show("Количество не может быть отрицательным!", "Ошибка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }

                Product product = new Product(name, price, quantity);
                products.Add(product);
                MessageBox.Show("Товар добавлен!", "Успех", MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
                UpdateProductList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }

        private void removeButton_Click(object sender, EventArgs e)
        {
            try

```

```

    {
        if (productsList.SelectedIndex >= 0)
        {
            products.RemoveAt(productsList.SelectedIndex);
            MessageBox.Show("Товар удален!", "Успех", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
            UpdateProductList();
        }
        else
        {
            MessageBox.Show("Выберите товар для удаления!", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
}

```

```

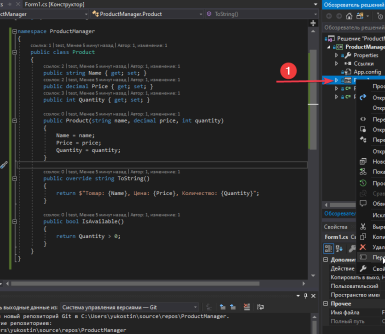
private void checkButton_Click(object sender, EventArgs e)
{
    try
    {
        if (productsList.SelectedIndex >= 0)
        {
            Product product = products[productsList.SelectedIndex];
            string message = product.IsAvailable() ? "Товар доступен!" : "Товар
                недоступен!";
            MessageBox.Show(message, "Статус", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show("Выберите товар для проверки!", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
}

```

```

private void UpdateProductList()
{

```



The screenshot shows the Visual Studio IDE with the following components:

- Menu Bar:** File, Edit, Project, View, Code, Window, Help, Build, Debug, Test, Tools, Extensions, Window, Help.
- Toolbar:** Includes icons for File, Edit, Project, View, Code, Window, Help, Build, Debug, Test, Tools, Extensions, Window, Help.
- Project Explorer (left):** Shows the 'ProductManager' project under the 'Solution' folder. The 'ProductManager' folder is highlighted with a red circle (1).
- Code Editor (center):** Displays the 'ProductManager.cs' file. The code defines a 'ProductManager' class with methods for adding, deleting, and updating products. The code is as follows:
 

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ProductManager
{
    class ProductManager
    {
        public List<Product> products;

        public ProductManager()
        {
            products = new List<Product>();
        }

        public void AddProduct(Product p)
        {
            products.Add(p);
        }

        public void DeleteProduct(Product p)
        {
            products.Remove(p);
        }

        public void UpdateProduct(Product p)
        {
            products[p.Id] = p;
        }

        public void GetProducts()
        {
            foreach (Product p in products)
            {
                Console.WriteLine(p.Id + " " + p.Name + " " + p.Price + " " + p.Quantity);
            }
        }

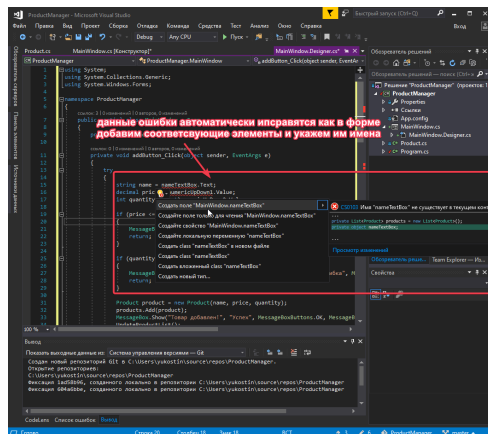
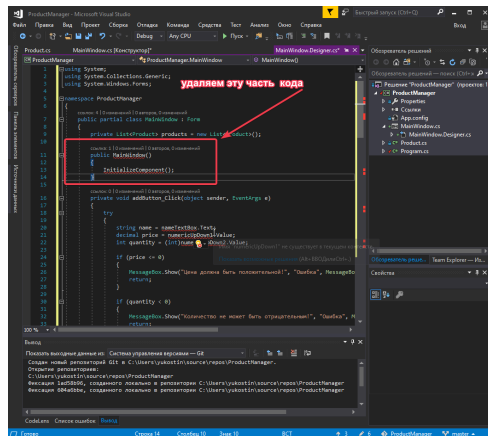
        public void GetProductById(int id)
        {
            Product p = products.FirstOrDefault(p => p.Id == id);
            if (p != null)
            {
                Console.WriteLine(p.Id + " " + p.Name + " " + p.Price + " " + p.Quantity);
            }
            else
            {
                Console.WriteLine("Product not found");
            }
        }

        public void GetProductsByName(string name)
        {
            foreach (Product p in products)
            {
                if (p.Name == name)
                {
                    Console.WriteLine(p.Id + " " + p.Name + " " + p.Price + " " + p.Quantity);
                }
            }
        }

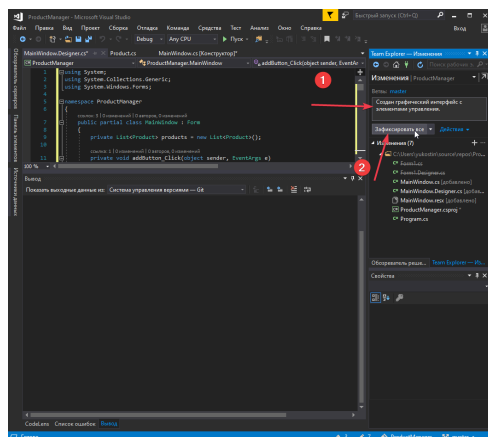
        public void GetProductsByPrice(int price)
        {
            foreach (Product p in products)
            {
                if (p.Price == price)
                {
                    Console.WriteLine(p.Id + " " + p.Name + " " + p.Price + " " + p.Quantity);
                }
            }
        }

        public void GetProductsByQuantity(int quantity)
        {
            foreach (Product p in products)
            {
                if (p.Quantity == quantity)
                {
                    Console.WriteLine(p.Id + " " + p.Name + " " + p.Price + " " + p.Quantity);
                }
            }
        }
    }
}
```
- Output Window (bottom):** Shows the output of the program, which is currently empty.
- Project Explorer (right):** Shows the 'ProductManager' project under the 'Solution' folder. The 'ProductManager' folder is highlighted with a red circle (2).

Видим наличие ошибок. Необходимо в зависимости от ситуации исправить ошибки.



## #### Фиксация Создание графического интерфейса

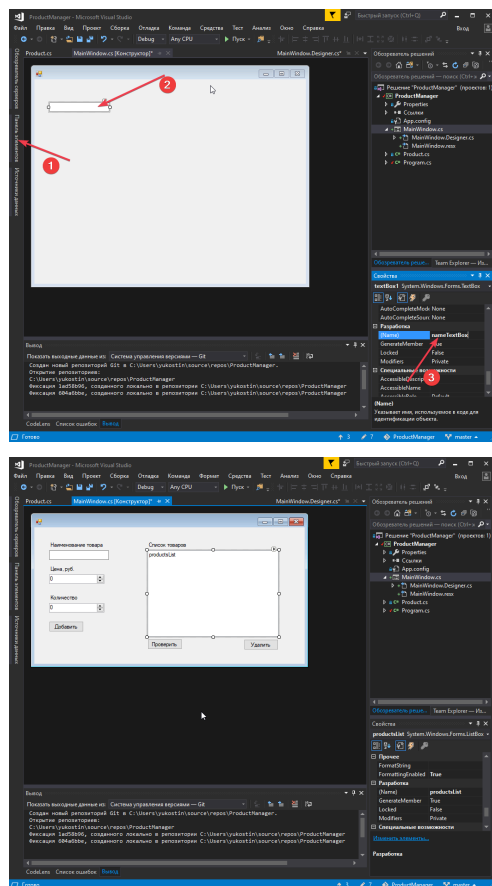


## #### 3. Дизайн формы

Создайте форму с следующими элементами:

- `TextBox` для ввода названия товара (`nameTextBox`).
- `NumericUpDown` для ввода цены (`numericUpDown1`).
- `NumericUpDown` для ввода количества (`numericUpDown2`).
- `ListBox` для отображения списка товаров (`productsList`).
- Кнопки `addButton`, `removeButton`, `checkButton` для добавления, удаления и проверки товара.





#### Автоматически сформированный код в файле **MainWindow.cs** после добавления элементов на форму

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ProductManager
{
    public partial class MainWindow : Form
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

```

private void InitializeComponent()
{
    this.nameTextBox = new System.Windows.Forms.TextBox();
    this.label1 = new System.Windows.Forms.Label();
    this.numericUpDown1 = new System.Windows.Forms.NumericUpDown();
    this.numericUpDown2 = new System.Windows.Forms.NumericUpDown();
    this.label2 = new System.Windows.Forms.Label();
    this.label3 = new System.Windows.Forms.Label();
    this.button1 = new System.Windows.Forms.Button();
    this.productsList = new System.Windows.Forms.ListBox();
    this.label4 = new System.Windows.Forms.Label();
    this.button2 = new System.Windows.Forms.Button();
    this.button3 = new System.Windows.Forms.Button();
    ((System.ComponentModel.ISupportInitialize)(this.numericUpDown1)).BeginInit();
    ((System.ComponentModel.ISupportInitialize)(this.numericUpDown2)).BeginInit();
    this.SuspendLayout();
    //
    // nameTextBox
    //
    this.nameTextBox.Location = new System.Drawing.Point(32, 50);
    this.nameTextBox.Name = "nameTextBox";
    this.nameTextBox.Size = new System.Drawing.Size(130, 20);
    this.nameTextBox.TabIndex = 0;
    //
    // label1
    //
    this.label1.AutoSize = true;
    this.label1.Location = new System.Drawing.Point(32, 31);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(121, 13);
    this.label1.TabIndex = 1;
    this.label1.Text = "Наименование товара";
    //
    // numericUpDown1
    //
    this.numericUpDown1.Location = new System.Drawing.Point(32, 102);
    this.numericUpDown1.Name = "numericUpDown1";
    this.numericUpDown1.Size = new System.Drawing.Size(120, 20);
    this.numericUpDown1.TabIndex = 2;
    //
    // numericUpDown2
    //
    this.numericUpDown2.Location = new System.Drawing.Point(32, 164);
    this.numericUpDown2.Name = "numericUpDown2";
    this.numericUpDown2.Size = new System.Drawing.Size(120, 20);
    this.numericUpDown2.TabIndex = 3;
    //
    // label2

```

```
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(32, 83);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(59, 13);
this.label2.TabIndex = 4;
this.label2.Text = "Цена, руб.";
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(32, 145);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(69, 13);
this.label3.TabIndex = 5;
this.label3.Text = "Количество ";
//
// button1
//
this.button1.Location = new System.Drawing.Point(32, 203);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(75, 23);
this.button1.TabIndex = 6;
this.button1.Text = "Добавить";
this.button1.UseVisualStyleBackColor = true;
this.button1.Click += new System.EventHandler(this.addButton_Click);
//
// productsList
//
this.productsList.FormattingEnabled = true;
this.productsList.Location = new System.Drawing.Point(246, 50);
this.productsList.Name = "productsList";
this.productsList.Size = new System.Drawing.Size(284, 186);
this.productsList.TabIndex = 7;
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(246, 31);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(88, 13);
this.label4.TabIndex = 8;
this.label4.Text = "Список товаров";
//
// button2
//
this.button2.Location = new System.Drawing.Point(455, 242);
this.button2.Name = "button2";
```

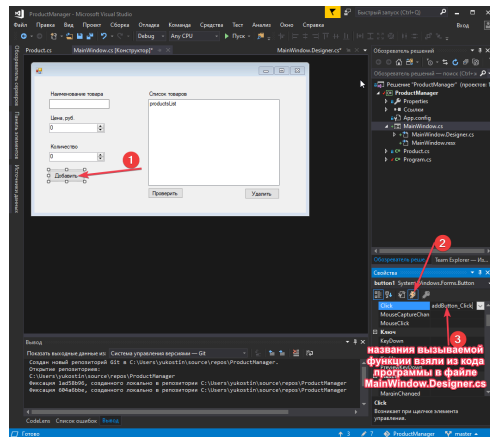
```

this.button2.Size = new System.Drawing.Size(75, 23);
this.button2.TabIndex = 9;
this.button2.Text = "Удалить";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.removeButton_Click);
//
// button3
//
this.button3.Location = new System.Drawing.Point(246, 241);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(75, 23);
this.button3.TabIndex = 10;
this.button3.Text = "Проверить";
this.button3.UseVisualStyleBackColor = true;
this.button3.Click += new System.EventHandler(this.checkButton_Click);
//
// MainWindow
//
this.ClientSize = new System.Drawing.Size(587, 295);
this.Controls.Add(this.button3);
this.Controls.Add(this.button2);
this.Controls.Add(this.label4);
this.Controls.Add(this.productsList);
this.Controls.Add(this.button1);
this.Controls.Add(this.label3);
this.Controls.Add(this.label2);
this.Controls.Add(this.numericUpDown2);
this.Controls.Add(this.numericUpDown1);
this.Controls.Add(this.label1);
this.Controls.Add(this.nameTextBox);
this.Name = "MainWindow";
((System.ComponentModel.ISupportInitialize)(this.numericUpDown1)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.numericUpDown2)).EndInit();
this.ResumeLayout(false);
this.PerformLayout();

}
}
}

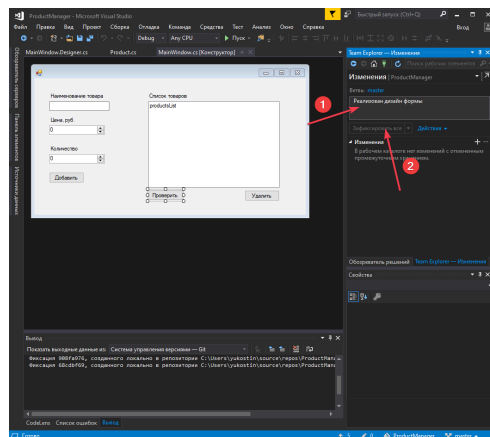
```

### Осталось прикрепить к событию “Click” кнопку соответствующие функции.



то же самое сделать с остальными кнопками

#### Фиксация Реализован дизайн формы



### Как использовать программу

1. Запустите программу.
2. Введите название товара, цену и количество.
3. Нажмите кнопку "Добавить", чтобы добавить товар.
4. Нажмите кнопку "Удалить", чтобы удалить товар из списка.
5. Нажмите кнопку "Проверить", чтобы проверить, доступен ли товар для продажи.

### Фиксация Версия программы

```
C:\Windows\SYSTEM32\cmd.exe

Microsoft Windows [Version 10.0.19045.5371]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\yukostin\source\repos\ProductManager>git tag v1.0.0

C:\Users\yukostin\source\repos\ProductManager>
```

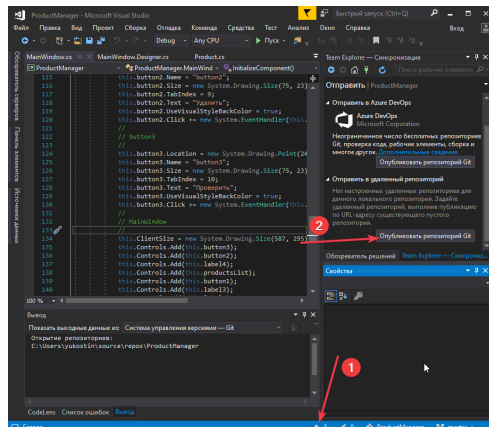
### ### Проверка граничных значений

В коде выше реализована проверка следующих граничных значений:

1. Цена товара должна быть положительной.
2. Количество товара не может быть отрицательным.
3. Проверка наличия товара (доступен ли товар для продажи).

---

### ### Отправка на удаленный репозиторий



## Варианты

### ### 1. Управление банковским счётом

Описание: Создать программу для управления банковским счётом, которая позволяет создавать новый счёт, пополнять его, снимать средства, проверять баланс и проверять граничные значения (например, отрицательные суммы, превышение лимита).

Код программы:

```
```csharp
using System;
using System.Windows.Forms;
```

```
public class BankAccount
{
    private string ownerName;
    private decimal balance;
    private const decimal maxAmount = 1000000;

    public BankAccount(string ownerName, decimal initialBalance)
    {
        this.ownerName = ownerName;
        this.balance = initialBalance;
    }

    public string GetOwnerName()
    {
        return ownerName;
    }

    public decimal GetBalance()
    {
        return balance;
    }

    public void Deposit(decimal amount)
    {
        if (amount < 0)
        {
            throw new ArgumentException("Сумма пополнения не может быть отрицательной.");
        }
        if (amount > maxAmount)
        {
            throw new ArgumentException($"Сумма пополнения не может превышать {maxAmount}.");
        }
        balance += amount;
    }

    public void Withdraw(decimal amount)
    {
        if (amount < 0)
        {
            throw new ArgumentException("Сумма снятия не может быть отрицательной.");
        }
        if (amount > maxAmount)
        {
            throw new ArgumentException($"Сумма снятия не может превышать {maxAmount}.");
        }
    }
}
```

```

    }
    if (amount > balance)
    {
        throw new InvalidOperationException("Недостаточно средств на счёте.");
    }
    balance -= amount;
}
}

```

```

public class BankAccountForm : Form
{
    private BankAccount account;
    private TextBox nameTextBox;
    private TextBox amountTextBox;
    private Label balanceLabel;
    private Button createAccountButton;
    private Button depositButton;
    private Button withdrawButton;

    public BankAccountForm()
    {
        this.Text = "Управление банковским счётом";
        this.Width = 400;
        this.Height = 300;

        nameTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 10),
            Width = 200,
            PlaceholderText = "Имя владельца"
        };

        amountTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 40),
            Width = 200,
            PlaceholderText = "Сумма"
        };

        createAccountButton = new Button
        {
            Location = new System.Drawing.Point(10, 70),
            Text = "Создать счёт",
            Width = 100
        };
        createAccountButton.Click += CreateAccountButton_Click;

        depositButton = new Button

```



```

{
    Location = new System.Drawing.Point(120, 70),
    Text = "Пополнить",
    Width = 100
};
depositButton.Click += DepositButton_Click;

withdrawButton = new Button
{
    Location = new System.Drawing.Point(10, 100),
    Text = "Снять",
    Width = 210
};
withdrawButton.Click += WithdrawButton_Click;

balanceLabel = new Label
{
    Location = new System.Drawing.Point(10, 130),
    Width = 200,
    Text = "Баланс: 0"
};

this.Controls.Add(nameTextBox);
this.Controls.Add(amountTextBox);
this.Controls.Add(createAccountButton);
this.Controls.Add(depositButton);
this.Controls.Add(withdrawButton);
this.Controls.Add(balanceLabel);
}

private void CreateAccountButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(nameTextBox.Text))
    {
        MessageBox.Show("Введите имя владельца!");
        return;
    }
    if (string.IsNullOrEmpty(amountTextBox.Text))
    {
        MessageBox.Show("Введите начальную сумму!");
        return;
    }
    decimal initialBalance;
    if (!decimal.TryParse(amountTextBox.Text, out initialBalance))
    {
        MessageBox.Show("Неверный формат суммы!");
        return;
    }
}

```

```
    account = new BankAccount(nameTextBox.Text, initialBalance);  
    balanceLabel.Text = $"Баланс: {initialBalance}";  
    MessageBox.Show("Счёт создан!");  
}
```

```
private void DepositButton_Click(object sender, EventArgs e)  
{  
    if (account == null)  
    {  
        MessageBox.Show("Сначала создайте счёт!");  
        return;  
    }  
    if (string.IsNullOrEmpty(amountTextBox.Text))  
    {  
        MessageBox.Show("Введите сумму для пополнения!");  
        return;  
    }  
    decimal amount;  
    if (!decimal.TryParse(amountTextBox.Text, out amount))  
    {  
        MessageBox.Show("Неверный формат суммы!");  
        return;  
    }  
    try  
    {  
        account.Deposit(amount);  
        balanceLabel.Text = $"Баланс: {account.GetBalance()}";  
        MessageBox.Show("Счёт пополнен!");  
    }  
    catch (Exception ex)  
    {  
        MessageBox.Show(ex.Message);  
    }  
}
```

```
private void WithdrawButton_Click(object sender, EventArgs e)  
{  
    if (account == null)  
    {  
        MessageBox.Show("Сначала создайте счёт!");  
        return;  
    }  
    if (string.IsNullOrEmpty(amountTextBox.Text))  
    {  
        MessageBox.Show("Введите сумму для снятия!");  
        return;  
    }  
    decimal amount;
```

```

        if (!decimal.TryParse(amountTextBox.Text, out amount))
        {
            MessageBox.Show("Неверный формат суммы!");
            return;
        }
        try
        {
            account.Withdraw(amount);
            balanceLabel.Text = $"Баланс: {account.GetBalance()}";
            MessageBox.Show("Средства сняты!");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new BankAccountForm());
    }
}
...

```

Объяснение:

1. Класс `BankAccount`:

- Имеет свойства для имени владельца и баланса.
- Методы для пополнения (`Deposit`), снятия (`Withdraw`) и получения баланса (`GetBalance`).
- Проверяет граничные значения: отрицательные суммы, превышение лимита (`maxAmount`), недостаток средств.

2. Графический интерфейс:

- Форма с полями для ввода имени и суммы.
- Кнопки для создания счёта, пополнения и снятия средств.
- Отображение текущего баланса.

3. Проверка граничных значений:

- Негативные суммы.
- Превышение максимальной суммы для операций.
- Недостаток средств при снятии.

---

## ### 2. Конвертер валют

Описание: Создать программу для конвертации валют. Пользователь может выбирать валюты для конвертации, вводить сумму и получать результат. Программа должна использовать актуальные курсы валют и проверять корректность ввода.

Код программы:

```
```csharp
using System;
using System.Windows.Forms;

public class CurrencyConverter
{
    private decimal usdToEur = 0.88m; // Примерный курс валют
    private decimal eurToUsd = 1.12m;

    public decimal Convert(decimal amount, string fromCurrency, string toCurrency)
    {
        if (amount < 0)
        {
            throw new ArgumentException("Сумма не может быть отрицательной.");
        }
        if (fromCurrency == "USD" && toCurrency == "EUR")
        {
            return amount * usdToEur;
        }
        else if (fromCurrency == "EUR" && toCurrency == "USD")
        {
            return amount * eurToUsd;
        }
        else
        {
            throw new NotSupportedException("Не поддерживаемая пара валют.");
        }
    }
}

public class CurrencyConverterForm : Form
{
    private CurrencyConverter converter;
    private ComboBox fromCurrencyComboBox;
    private ComboBox toCurrencyComboBox;
    private TextBox amountTextBox;
    private Button convertButton;
    private Label resultLabel;

    public CurrencyConverterForm()
```

```

{
    this.Text = "Конвертер валют";
    this.Width = 300;
    this.Height = 200;

    fromCurrencyComboBox = new ComboBox
    {
        Location = new System.Drawing.Point(10, 10),
        Width = 100,
        Items = { "USD", "EUR" }
    };

    toCurrencyComboBox = new ComboBox
    {
        Location = new System.Drawing.Point(120, 10),
        Width = 100,
        Items = { "USD", "EUR" }
    };

    amountTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 40),
        Width = 210,
        PlaceholderText = "Сумма"
    };

    convertButton = new Button
    {
        Location = new System.Drawing.Point(10, 70),
        Text = "Конвертировать",
        Width = 210
    };

    convertButton.Click += ConvertButton_Click;

    resultLabel = new Label
    {
        Location = new System.Drawing.Point(10, 100),
        Width = 210,
        Text = "Результат: "
    };

    this.Controls.Add(fromCurrencyComboBox);
    this.Controls.Add(toCurrencyComboBox);
    this.Controls.Add(amountTextBox);
    this.Controls.Add(convertButton);
    this.Controls.Add(resultLabel);
}

```

```

private void ConvertButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(amountTextBox.Text))
    {
        MessageBox.Show("Введите сумму для конвертации!");
        return;
    }
    decimal amount;
    if (!decimal.TryParse(amountTextBox.Text, out amount))
    {
        MessageBox.Show("Неверный формат суммы!");
        return;
    }
    string fromCurrency = fromCurrencyComboBox.SelectedItem.ToString();
    string toCurrency = toCurrencyComboBox.SelectedItem.ToString();

    try
    {
        decimal result = converter.Convert(amount, fromCurrency, toCurrency);
        resultLabel.Text = $"Результат: {amount} {fromCurrency} = {result} {toCurrency}";
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new CurrencyConverterForm());
}
...

```

Объяснение:

1. Класс `CurrencyConverter`:

- Реализует метод `Convert` для конвертации валют.
- Проверяет отрицательные суммы и поддерживает только определённые пары валют.

2. Графический интерфейс:

- Поля для выбора валют и ввода суммы.
- Кнопка для выполнения конвертации.
- Отображение результата.

### 3. Проверка граничных значений:

- Отрицательные суммы.
- Некорректные пары валют.

---

## ### 3. Управление задачами

Описание: Создать программу для управления задачами. Пользователь может добавлять, удалять, редактировать задачи и отмечать их как выполненные. Программа должна сохранять задачи в файл и загружать их при запуске.

Код программы:

```
```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public class TaskManager
{
    public List<Task> Tasks { get; private set; }

    public TaskManager()
    {
        Tasks = new List<Task>();
        LoadTasks();
    }

    public void AddTask(string description)
    {
        if (string.IsNullOrEmpty(description))
        {
            throw new ArgumentException("Описание задачи不能为空.");
        }
        Tasks.Add(new Task(description));
        SaveTasks();
    }

    public void RemoveTask(int index)
    {
        if (index < 0 || index >= Tasks.Count)
        {
            throw new IndexOutOfRangeException("Некорректный индекс задачи.");
        }
    }
}
```

```

        Tasks.RemoveAt(index);
        SaveTasks();
    }

    public void ToggleTaskCompletion(int index)
    {
        if (index < 0 || index >= Tasks.Count)
        {
            throw new IndexOutOfRangeException("Некорректный индекс задачи.");
        }
        Tasks[index].IsCompleted = !Tasks[index].IsCompleted;
        SaveTasks();
    }

    private void SaveTasks()
    {
        File.WriteAllLines("tasks.txt", Tasks.Select(t => $"{t.IsCompleted}|{t.Description}"));
    }

    private void LoadTasks()
    {
        if (File.Exists("tasks.txt"))
        {
            var lines = File.ReadAllLines("tasks.txt");
            foreach (var line in lines)
            {
                var parts = line.Split('|');
                if (parts.Length == 2)
                {
                    bool isCompleted = bool.Parse(parts[0]);
                    string description = parts[1];
                    Tasks.Add(new Task(description) { IsCompleted = isCompleted });
                }
            }
        }
    }
}

public class Task
{
    public string Description { get; set; }
    public bool IsCompleted { get; set; }

    public Task(string description)
    {
        Description = description;
        IsCompleted = false;
    }
}

```



```
}
```

```
public class TaskManagerForm : Form
```

```
{
```

```
    private TaskManager taskManager;  
    private ListBox tasksListBox;  
    private TextBox descriptionTextBox;  
    private Button addTaskButton;  
    private Button removeTaskButton;  
    private Button toggleCompletionButton;
```

```
    public TaskManagerForm()
```

```
    {
```

```
        this.Text = "Управление задачами";  
        this.Width = 400;  
        this.Height = 400;
```

```
        tasksListBox = new ListBox
```

```
        {
```

```
            Location = new System.Drawing.Point(10, 10),  
            Width = 200,  
            Height = 200
```

```
        };
```

```
        descriptionTextBox = new TextBox
```

```
        {
```

```
            Location = new System.Drawing.Point(220, 10),  
            Width = 150
```

```
        };
```

```
        addTaskButton = new Button
```

```
        {
```

```
            Location = new System.Drawing.Point(220, 40),  
            Text = "Добавить",  
            Width = 70
```

```
        };
```

```
        addTaskButton.Click += AddTaskButton_Click;
```

```
        removeTaskButton = new Button
```

```
        {
```

```
            Location = new System.Drawing.Point(300, 40),  
            Text = "Удалить",  
            Width = 70
```

```
        };
```

```
        removeTaskButton.Click += RemoveTaskButton_Click;
```

```
        toggleCompletionButton = new Button
```

```
        {
```

```

        Location = new System.Drawing.Point(220, 70),
        Text = "Отметить",
        Width = 150
    };
    toggleCompletionButton.Click += ToggleCompletionButton_Click;

    this.Controls.Add(tasksListBox);
    this.Controls.Add(descriptionTextBox);
    this.Controls.Add(addTaskButton);
    this.Controls.Add(removeTaskButton);
    this.Controls.Add(toggleCompletionButton);

    taskManager = new TaskManager();
    UpdateTasksList();
}

private void UpdateTasksList()
{
    tasksListBox.Items.Clear();
    foreach (var task in taskManager.Tasks)
    {
        tasksListBox.Items.Add($"{(task.IsCompleted ? "[X]" : "[ ]")} {task.Description}");
    }
}

private void AddTaskButton_Click(object sender, EventArgs e)
{
    try
    {
        taskManager.AddTask(descriptionTextBox.Text);
        descriptionTextBox.Clear();
        UpdateTasksList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void RemoveTaskButton_Click(object sender, EventArgs e)
{
    if (tasksListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите задачу для удаления!");
        return;
    }
    try
    {

```

```

        taskManager.RemoveTask(tasksListBox.SelectedIndex);
        UpdateTasksList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void ToggleCompletionButton_Click(object sender, EventArgs e)
{
    if (tasksListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите задачу для изменения статуса!");
        return;
    }
    try
    {
        taskManager.ToggleTaskCompletion(tasksListBox.SelectedIndex);
        UpdateTasksList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new TaskManagerForm());
}
...

```

Объяснение:

1. Класс `TaskManager`:

- Управляет списком задач.
- Реализует методы для добавления, удаления и изменения статуса задач.
- Сохраняет и загружает задачи из файла.

2. Класс `Task`:

- Имеет описание и статус выполнения.

3. Графический интерфейс:

- Список задач.

- Поле для ввода нового описания задачи.
- Кнопки для добавления, удаления и изменения статуса задачи.

#### 4. Проверка граничных значений:

- Пустое описание задачи.
- Некорректный выбор задачи для удаления или изменения статуса.

---

### ### 4. Календарь событий

Описание: Создать программу, которая позволяет добавлять, удалять и редактировать события в календаре. События должны сохраняться в файл и загружаться при запуске.

Код программы:

```
```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;

public class Event
{
    public DateTime Date { get; set; }
    public string Description { get; set; }

    public Event(DateTime date, string description)
    {
        Date = date;
        Description = description;
    }
}

public class CalendarManager
{
    public List<Event> Events { get; private set; }

    public CalendarManager()
    {
        Events = new List<Event>();
        LoadEvents();
    }

    public void AddEvent(Event e)
    {
        if (e == null)
```

```

        {
            throw new ArgumentNullException(nameof(e));
        }
        Events.Add(e);
        SaveEvents();
    }

    public void RemoveEvent(Event e)
    {
        if (e == null)
        {
            throw new ArgumentNullException(nameof(e));
        }
        Events.Remove(e);
        SaveEvents();
    }

    public void SaveEvents()
    {
        File.WriteAllLines("events.txt", Events.Select(e =>
        $"{e.Date.ToString("yyyy-MM-dd")}|{e.Description}"));
    }

    public void LoadEvents()
    {
        if (File.Exists("events.txt"))
        {
            var lines = File.ReadAllLines("events.txt");
            foreach (var line in lines)
            {
                var parts = line.Split('|');
                if (parts.Length == 2)
                {
                    DateTime date;
                    if (DateTime.TryParse(parts[0], out date))
                    {
                        Events.Add(new Event(date, parts[1]));
                    }
                }
            }
        }
    }
}

public class CalendarForm : Form
{
    private CalendarManager calendarManager;
    private DateTimePicker datePicker;

```

```

private TextBox descriptionTextBox;
private Button addEventButton;
private ListBox eventsListBox;
private Button removeEventButton;

public CalendarForm()
{
    this.Text = "Календарь событий";
    this.Width = 400;
    this.Height = 400;

    datePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(10, 10)
    };

    descriptionTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 40),
        Width = 200
    };

    addEventButton = new Button
    {
        Location = new System.Drawing.Point(10, 70),
        Text = "Добавить",
        Width = 100
    };
    addEventButton.Click += AddEventButton_Click;

    eventsListBox = new ListBox
    {
        Location = new System.Drawing.Point(10, 100),
        Width = 200,
        Height = 200
    };

    removeEventButton = new Button
    {
        Location = new System.Drawing.Point(220, 100),
        Text = "Удалить",
        Width = 80,
        Height = 20
    };
    removeEventButton.Click += RemoveEventButton_Click;

    this.Controls.Add(datePicker);
    this.Controls.Add(descriptionTextBox);
}

```

```

        this.Controls.Add(addEventButton);
        this.Controls.Add(eventsListBox);
        this.Controls.Add(removeEventButton);

        calendarManager = new CalendarManager();
        UpdateEventsList();
    }

    private void UpdateEventsList()
    {
        eventsListBox.Items.Clear();
        foreach (var e in calendarManager.Events)
        {
            eventsListBox.Items.Add($"{e.Date.ToString("yyyy-MM-dd")} - {e.Description}");
        }
    }

    private void AddEventButton_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(descriptionTextBox.Text))
        {
            MessageBox.Show("Введите описание события!");
            return;
        }
        Event newEvent = new Event(datePicker.Value, descriptionTextBox.Text);
        try
        {
            calendarManager.AddEvent(newEvent);
            descriptionTextBox.Clear();
            UpdateEventsList();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void RemoveEventButton_Click(object sender, EventArgs e)
    {
        if (eventsListBox.SelectedIndex == -1)
        {
            MessageBox.Show("Выберите событие для удаления!");
            return;
        }
        string selectedItem = eventsListBox.SelectedItem.ToString();
        DateTime date;
        if (DateTime.TryParse(selectedItem.Split(new[] { '-' }, StringSplitOptions.None)[0], out
date))

```

```

    {
        var eventToRemove = calendarManager.Events.Find(e => e.Date == date);
        if (eventToRemove != null)
        {
            try
            {
                calendarManager.RemoveEvent(eventToRemove);
                UpdateEventsList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new CalendarForm());
}
}
...

```

Объяснение:

1. Класс `Event`:
  - Хранит дату и описание события.
2. Класс `CalendarManager`:
  - Управляет списком событий.
  - Сохраняет и загружает события из файла.
3. Графический интерфейс:
  - Поле для выбора даты.
  - Поле для ввода описания события.
  - Кнопки для добавления и удаления событий.
  - Список текущих событий.
4. Проверка граничных значений:
  - Пустое описание события.
  - Некорректный выбор события для удаления.

---



## ### 5. Управление книгами в библиотеке

Описание: Создать программу для управления книгами в библиотеке. Пользователь может добавлять, удалять, редактировать книги и искать по автору или названию. Программа должна сохранять данные в файл.

Код программы:

```
```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public class Book
{
    public string Author { get; set; }
    public string Title { get; set; }
    public string Year { get; set; }

    public Book(string author, string title, string year)
    {
        Author = author;
        Title = title;
        Year = year;
    }
}

public class LibraryManager
{
    public List<Book> Books { get; private set; }

    public LibraryManager()
    {
        Books = new List<Book>();
        LoadBooks();
    }

    public void AddBook(Book book)
    {
        if (book == null)
        {
            throw new ArgumentNullException(nameof(book));
        }
        Books.Add(book);
        SaveBooks();
    }
}
```

```

public void RemoveBook(Book book)
{
    if (book == null)
    {
        throw new ArgumentNullException(nameof(book));
    }
    Books.Remove(book);
    SaveBooks();
}

public List<Book> SearchBooks(string query)
{
    return Books.Where(b => b.Author.Contains(query) || b.Title.Contains(query)).ToList();
}

private void SaveBooks()
{
    File.WriteAllLines("books.txt", Books.Select(b => $"{b.Author}|{b.Title}|{b.Year}"));
}

private void LoadBooks()
{
    if (File.Exists("books.txt"))
    {
        var lines = File.ReadAllLines("books.txt");
        foreach (var line in lines)
        {
            var parts = line.Split('|');
            if (parts.Length == 3)
            {
                Books.Add(new Book(parts[0], parts[1], parts[2]));
            }
        }
    }
}

public class LibraryForm : Form
{
    private LibraryManager libraryManager;
    private TextBox authorTextBox;
    private TextBox titleTextBox;
    private TextBox yearTextBox;
    private Button addBookButton;
    private Button removeBookButton;
    private TextBox searchTextBox;
    private Button searchButton;
}

```

```
private ListBox booksListBox;

public LibraryForm()
{
    this.Text = "Управление книгами";
    this.Width = 500;
    this.Height = 400;

    authorTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 10),
        Width = 150,
        PlaceholderText = "Автор"
    };

    titleTextBox = new TextBox
    {
        Location = new System.Drawing.Point(170, 10),
        Width = 150,
        PlaceholderText = "Название"
    };

    yearTextBox = new TextBox
    {
        Location = new System.Drawing.Point(330, 10),
        Width = 80,
        PlaceholderText = "Год"
    };

    addBookButton = new Button
    {
        Location = new System.Drawing.Point(10, 40),
        Text = "Добавить",
        Width = 100
    };
    addBookButton.Click += AddBookButton_Click;

    removeBookButton = new Button
    {
        Location = new System.Drawing.Point(120, 40),
        Text = "Удалить",
        Width = 100
    };
    removeBookButton.Click += RemoveBookButton_Click;

    searchTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 70),
```

```

        Width = 200,
        PlaceholderText = "Поиск"
    };

    searchButton = new Button
    {
        Location = new System.Drawing.Point(220, 70),
        Text = "Искать",
        Width = 80
    };
    searchButton.Click += SearchButton_Click;

    booksListBox = new ListBox
    {
        Location = new System.Drawing.Point(10, 100),
        Width = 450,
        Height = 200
    };

    this.Controls.Add(authorTextBox);
    this.Controls.Add(titleTextBox);
    this.Controls.Add(yearTextBox);
    this.Controls.Add(addBookButton);
    this.Controls.Add(removeBookButton);
    this.Controls.Add(searchTextBox);
    this.Controls.Add(searchButton);
    this.Controls.Add(booksListBox);

    libraryManager = new LibraryManager();
    UpdateBooksList();
}

private void UpdateBooksList()
{
    booksListBox.Items.Clear();
    foreach (var book in libraryManager.Books)
    {
        booksListBox.Items.Add($"{book.Author} - {book.Title} ({book.Year})");
    }
}

private void AddBookButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(authorTextBox.Text) || string.IsNullOrEmpty(titleTextBox.Text) ||
        string.IsNullOrEmpty(yearTextBox.Text))
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
}

```

```

    }
    Book newBook = new Book(authorTextBox.Text, titleTextBox.Text, yearTextBox.Text);
    try
    {
        libraryManager.AddBook(newBook);
        authorTextBox.Clear();
        titleTextBox.Clear();
        yearTextBox.Clear();
        UpdateBooksList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void RemoveBookButton_Click(object sender, EventArgs e)
{
    if (booksListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите книгу для удаления!");
        return;
    }
    string selectedItem = booksListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string author = parts[0].Trim();
        string title = parts[1].Trim();
        var bookToRemove = libraryManager.Books.Find(b => b.Author == author && b.Title
== title);
        if (bookToRemove != null)
        {
            try
            {
                libraryManager.RemoveBook(bookToRemove);
                UpdateBooksList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

private void SearchButton_Click(object sender, EventArgs e)
{

```

```

        if (string.IsNullOrEmpty(searchTextBox.Text))
        {
            UpdateBooksList();
            return;
        }
        var searchResults = libraryManager.SearchBooks(searchTextBox.Text);
        booksListBox.Items.Clear();
        foreach (var book in searchResults)
        {
            booksListBox.Items.Add($"{book.Author} - {book.Title} ({book.Year})");
        }
    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new LibraryForm());
    }
}
...

```

Объяснение:

1. Класс `Book`:

- Хранит автора, название и год издания книги.

2. Класс `LibraryManager`:

- Управляет списком книг.
- Реализует методы для добавления, удаления и поиска книг.
- Сохраняет и загружает книги из файла.

3. Графический интерфейс:

- Поля для ввода автора, названия и года издания.
- Кнопки для добавления и удаления книг.
- Поле для поиска и кнопка для выполнения поиска.
- Список книг.

4. Проверка граничных значений:

- Пустые поля при добавлении книги.
- Некорректный выбор книги для удаления.

---

## ### 6. Управление заметками

Описание: Создать программу для управления заметками. Пользователь может добавлять, удалять, редактировать заметки и сохранять их в файл.

Код программы:

```
```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;

public class Note
{
    public string Title { get; set; }
    public string Content { get; set; }
    public DateTime Date { get; set; }

    public Note(string title, string content)
    {
        Title = title;
        Content = content;
        Date = DateTime.Now;
    }
}

public class NoteManager
{
    public List<Note> Notes { get; private set; }

    public NoteManager()
    {
        Notes = new List<Note>();
        LoadNotes();
    }

    public void AddNote(Note note)
    {
        if (note == null)
        {
            throw new ArgumentNullException(nameof(note));
        }
        Notes.Add(note);
        SaveNotes();
    }

    public void RemoveNote(Note note)
```

```

    {
        if (note == null)
        {
            throw new ArgumentNullException(nameof(note));
        }
        Notes.Remove(note);
        SaveNotes();
    }

    private void SaveNotes()
    {
        File.WriteAllLines("notes.txt", Notes.Select(n =>
${n.Title}{n.Content}{n.Date.ToString("yyyy-MM-dd HH:mm:ss")}"));
    }

    private void LoadNotes()
    {
        if (File.Exists("notes.txt"))
        {
            var lines = File.ReadAllLines("notes.txt");
            foreach (var line in lines)
            {
                var parts = line.Split('|');
                if (parts.Length == 3)
                {
                    DateTime date;
                    if (DateTime.TryParse(parts[2], out date))
                    {
                        Notes.Add(new Note(parts[0], parts[1], date));
                    }
                }
            }
        }
    }
}

public class NoteForm : Form
{
    private NoteManager noteManager;
    private TextBox titleTextBox;
    private TextBox contentTextBox;
    private Button addNoteButton;
    private ListBox notesListBox;
    private Button removeNoteButton;

    public NoteForm()
    {
        this.Text = "Управление заметками";
    }
}

```



```
this.Width = 500;
this.Height = 400;

titleTextBox = new TextBox
{
    Location = new System.Drawing.Point(10, 10),
    Width = 200,
    PlaceholderText = "Заголовок"
};

contentTextBox = new TextBox
{
    Location = new System.Drawing.Point(10, 40),
    Width = 200,
    Height = 100,
    Multiline = true,
    ScrollBars = ScrollBars.Both,
    PlaceholderText = "Содержание"
};

addNoteButton = new Button
{
    Location = new System.Drawing.Point(10, 150),
    Text = "Добавить",
    Width = 100
};
addNoteButton.Click += AddNoteButton_Click;

notesListBox = new ListBox
{
    Location = new System.Drawing.Point(220, 10),
    Width = 250,
    Height = 200
};

removeNoteButton = new Button
{
    Location = new System.Drawing.Point(220, 220),
    Text = "Удалить",
    Width = 100
};
removeNoteButton.Click += RemoveNoteButton_Click;

this.Controls.Add(titleTextBox);
this.Controls.Add(contentTextBox);
this.Controls.Add(addNoteButton);
this.Controls.Add(notesListBox);
this.Controls.Add(removeNoteButton);
```

```

        noteManager = new NoteManager();
        UpdateNotesList();
    }

    private void UpdateNotesList()
    {
        notesListBox.Items.Clear();
        foreach (var note in noteManager.Notes)
        {
            notesListBox.Items.Add($"{note.Title} ({note.Date.ToString("yyyy-MM-dd")})");
        }
    }

    private void AddNoteButton_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(titleTextBox.Text) || string.IsNullOrEmpty(contentTextBox.Text))
        {
            MessageBox.Show("Заполните все поля!");
            return;
        }
        Note newNote = new Note(titleTextBox.Text, contentTextBox.Text);
        try
        {
            noteManager.AddNote(newNote);
            titleTextBox.Clear();
            contentTextBox.Clear();
            UpdateNotesList();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void RemoveNoteButton_Click(object sender, EventArgs e)
    {
        if (notesListBox.SelectedIndex == -1)
        {
            MessageBox.Show("Выберите заметку для удаления!");
            return;
        }
        string selectedItem = notesListBox.SelectedItem.ToString();
        string[] parts = selectedItem.Split(new[] { '(' }, StringSplitOptions.None);
        if (parts.Length >= 2)
        {
            string title = parts[0];
            DateTime date;

```

```

        if (DateTime.TryParse(parts[1].Split(' ')[0], out date))
        {
            var noteToRemove = noteManager.Notes.Find(n => n.Title == title && n.Date.Date
== date.Date);
            if (noteToRemove != null)
            {
                try
                {
                    noteManager.RemoveNote(noteToRemove);
                    UpdateNotesList();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new NoteForm());
}
}
...

```

Объяснение:

1. Класс `Note`:

- Хранит заголовок, содержание и дату создания заметки.

2. Класс `NoteManager`:

- Управляет списком заметок.
- Сохраняет и загружает заметки из файла.

3. Графический интерфейс:

- Поля для ввода заголовка и содержания заметки.
- Кнопки для добавления и удаления заметок.
- Список текущих заметок.

4. Проверка граничных значений:

- Пустые поля при добавлении заметки.
- Некорректный выбор заметки для удаления.

---

## ### 7. Управление контактами

Описание: Создать программу для управления контактами. Пользователь может добавлять, удалять, редактировать контакты и искать по имени или номеру телефона. Программа должна сохранять контакты в файл.

Код программы:

```
``csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public class Contact
{
    public string Name { get; set; }
    public string PhoneNumber { get; set; }

    public Contact(string name, string phoneNumber)
    {
        Name = name;
        PhoneNumber = phoneNumber;
    }
}

public class ContactManager
{
    public List<Contact> Contacts { get; private set; }

    public ContactManager()
    {
        Contacts = new List<Contact>();
        LoadContacts();
    }

    public void AddContact(Contact contact)
    {
        if (contact == null)
        {
            throw new ArgumentNullException(nameof(contact));
        }
        Contacts.Add(contact);
        SaveContacts();
    }
}
```

```

public void RemoveContact(Contact contact)
{
    if (contact == null)
    {
        throw new ArgumentNullException(nameof(contact));
    }
    Contacts.Remove(contact);
    SaveContacts();
}

public List<Contact> SearchContacts(string query)
{
    return Contacts.Where(c => c.Name.Contains(query) ||
c.PhoneNumber.Contains(query)).ToList();
}

private void SaveContacts()
{
    File.WriteAllLines("contacts.txt", Contacts.Select(c => $"{c.Name}|{c.PhoneNumber}"));
}

private void LoadContacts()
{
    if (File.Exists("contacts.txt"))
    {
        var lines = File.ReadAllLines("contacts.txt");
        foreach (var line in lines)
        {
            var parts = line.Split('|');
            if (parts.Length == 2)
            {
                Contacts.Add(new Contact(parts[0], parts[1]));
            }
        }
    }
}

public class ContactForm : Form
{
    private ContactManager contactManager;
    private TextBox nameTextBox;
    private TextBox phoneNumberTextBox;
    private Button addContactButton;
    private Button removeContactButton;
    private TextBox searchTextBox;
    private Button searchButton;
}

```

```
private ListBox contactsListBox;

public ContactForm()
{
    this.Text = "Управление контактами";
    this.Width = 500;
    this.Height = 400;

    nameTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 10),
        Width = 150,
        PlaceholderText = "Имя"
    };

    phoneNumberTextBox = new TextBox
    {
        Location = new System.Drawing.Point(170, 10),
        Width = 150,
        PlaceholderText = "Телефон"
    };

    addContactButton = new Button
    {
        Location = new System.Drawing.Point(10, 40),
        Text = "Добавить",
        Width = 100
    };
    addContactButton.Click += AddContactButton_Click;

    removeContactButton = new Button
    {
        Location = new System.Drawing.Point(120, 40),
        Text = "Удалить",
        Width = 100
    };
    removeContactButton.Click += RemoveContactButton_Click;

    searchTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 70),
        Width = 200,
        PlaceholderText = "Поиск"
    };

    searchButton = new Button
    {
        Location = new System.Drawing.Point(220, 70),
```

```

        Text = "Искать",
        Width = 80
    };
    searchButton.Click += SearchButton_Click;

    contactsListBox = new ListBox
    {
        Location = new System.Drawing.Point(10, 100),
        Width = 450,
        Height = 200
    };

    this.Controls.Add(nameTextBox);
    this.Controls.Add(phoneNumberTextBox);
    this.Controls.Add(addContactButton);
    this.Controls.Add(removeContactButton);
    this.Controls.Add(searchTextBox);
    this.Controls.Add(searchButton);
    this.Controls.Add(contactsListBox);

    contactManager = new ContactManager();
    UpdateContactsList();
}

private void UpdateContactsList()
{
    contactsListBox.Items.Clear();
    foreach (var contact in contactManager.Contacts)
    {
        contactsListBox.Items.Add($"{contact.Name} - {contact.PhoneNumber}");
    }
}

private void AddContactButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(nameTextBox.Text) ||
        string.IsNullOrEmpty(phoneNumberTextBox.Text))
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    Contact newContact = new Contact(nameTextBox.Text, phoneNumberTextBox.Text);
    try
    {
        contactManager.AddContact(newContact);
        nameTextBox.Clear();
        phoneNumberTextBox.Clear();
        UpdateContactsList();
    }
    catch { }
}

```

```

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void RemoveContactButton_Click(object sender, EventArgs e)
{
    if (contactsListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите контакт для удаления!");
        return;
    }
    string selectedItem = contactsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string name = parts[0].Trim();
        string phoneNumber = parts[1].Trim();
        var contactToRemove = contactManager.Contacts.Find(c => c.Name == name &&
c.PhoneNumber == phoneNumber);
        if (contactToRemove != null)
        {
            try
            {
                contactManager.RemoveContact(contactToRemove);
                UpdateContactsList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

private void SearchButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(searchTextBox.Text))
    {
        UpdateContactsList();
        return;
    }
    var searchResults = contactManager.SearchContacts(searchTextBox.Text);
    contactsListBox.Items.Clear();
    foreach (var contact in searchResults)
    {

```



```

        contactsListBox.Items.Add($"{contact.Name} - {contact.PhoneNumber}");
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new ContactForm());
}
}
...

```

Объяснение:

1. Класс `Contact`:

- Хранит имя и номер телефона контакта.

2. Класс `ContactManager`:

- Управляет списком контактов.
- Реализует методы для добавления, удаления и поиска контактов.
- Сохраняет и загружает контакты из файла.

3. Графический интерфейс:

- Поля для ввода имени и номера телефона.
- Кнопки для добавления и удаления контактов.
- Поле для поиска и кнопка для выполнения поиска.
- Список контактов.

4. Проверка граничных значений:

- Пустые поля при добавлении контакта.
- Некорректный выбор контакта для удаления.

---

## ### 8. Управление покупками

Описание: Создать программу для управления покупками. Пользователь может добавлять, удалять, редактировать покупки и сортировать их по категориям. Программа должна сохранять данные в файл.

Код программы:

```

```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;

```

```

using System.IO;
using System.Linq;

public enum Category
{
    Продукты,
    Техника,
    Одежда,
    Прочее
}

public class Purchase
{
    public string Name { get; set; }
    public decimal Price { get; set; }
    public Category Category { get; set; }
    public DateTime Date { get; set; }

    public Purchase(string name, decimal price, Category category, DateTime date)
    {
        Name = name;
        Price = price;
        Category = category;
        Date = date;
    }
}

public class PurchaseManager
{
    public List<Purchase> Purchases { get; private set; }

    public PurchaseManager()
    {
        Purchases = new List<Purchase>();
        LoadPurchases();
    }

    public void AddPurchase(Purchase purchase)
    {
        if (purchase == null)
        {
            throw new ArgumentNullException(nameof(purchase));
        }
        Purchases.Add(purchase);
        SavePurchases();
    }

    public void RemovePurchase(Purchase purchase)

```

```

{
    if (purchase == null)
    {
        throw new ArgumentNullException(nameof(purchase));
    }
    Purchases.Remove(purchase);
    SavePurchases();
}

public List<Purchase> GetPurchasesByCategory(Category category)
{
    return Purchases.Where(p => p.Category == category).ToList();
}

private void SavePurchases()
{
    File.WriteAllLines("purchases.txt", Purchases.Select(p =>
    $"{p.Name}|{p.Price}|{(int)p.Category}|{p.Date.ToString("yyyy-MM-dd HH:mm:ss")}"));
}

private void LoadPurchases()
{
    if (File.Exists("purchases.txt"))
    {
        var lines = File.ReadAllLines("purchases.txt");
        foreach (var line in lines)
        {
            var parts = line.Split('|');
            if (parts.Length == 4)
            {
                decimal price;
                if (decimal.TryParse(parts[1], out price))
                {
                    Category category = (Category)Enum.Parse(typeof(Category), parts[2]);
                    DateTime date;
                    if (DateTime.TryParse(parts[3], out date))
                    {
                        Purchases.Add(new Purchase(parts[0], price, category, date));
                    }
                }
            }
        }
    }
}

public class PurchaseForm : Form
{

```

```

private PurchaseManager purchaseManager;
private TextBox nameTextBox;
private TextBox priceTextBox;
private ComboBox categoryComboBox;
private DateTimePicker datePicker;
private Button addPurchaseButton;
private Button removePurchaseButton;
private ComboBox categoryFilterComboBox;
private Button filterButton;
private ListBox purchasesListBox;

public PurchaseForm()
{
    this.Text = "Управление покупками";
    this.Width = 600;
    this.Height = 500;

    nameTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 10),
        Width = 150,
        PlaceholderText = "Название"
    };

    priceTextBox = new TextBox
    {
        Location = new System.Drawing.Point(170, 10),
        Width = 100,
        PlaceholderText = "Цена"
    };

    categoryComboBox = new ComboBox
    {
        Location = new System.Drawing.Point(280, 10),
        Width = 100,
        Items = { "Продукты", "Техника", "Одежда", "Прочее" }
    };

    datePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(390, 10)
    };

    addPurchaseButton = new Button
    {
        Location = new System.Drawing.Point(10, 40),
        Text = "Добавить",
        Width = 100
    }
}

```

```

};
addPurchaseButton.Click += AddPurchaseButton_Click;

removePurchaseButton = new Button
{
    Location = new System.Drawing.Point(120, 40),
    Text = "Удалить",
    Width = 100
};
removePurchaseButton.Click += RemovePurchaseButton_Click;

categoryFilterComboBox = new ComboBox
{
    Location = new System.Drawing.Point(10, 70),
    Width = 100,
    Items = { "Все", "Продукты", "Техника", "Одежда", "Прочее" }
};

filterButton = new Button
{
    Location = new System.Drawing.Point(120, 70),
    Text = "Фильтровать",
    Width = 100
};
filterButton.Click += FilterButton_Click;

purchasesListBox = new ListBox
{
    Location = new System.Drawing.Point(10, 100),
    Width = 560,
    Height = 300
};

this.Controls.Add(nameTextBox);
this.Controls.Add(priceTextBox);
this.Controls.Add(categoryComboBox);
this.Controls.Add(datePicker);
this.Controls.Add(addPurchaseButton);
this.Controls.Add(removePurchaseButton);
this.Controls.Add(categoryFilterComboBox);
this.Controls.Add(filterButton);
this.Controls.Add(purchasesListBox);

purchaseManager = new PurchaseManager();
UpdatePurchasesList();
}

private void UpdatePurchasesList()

```

```

{
    purchasesListBox.Items.Clear();
    foreach (var purchase in purchaseManager.Purchases)
    {
        purchasesListBox.Items.Add($"{purchase.Name} - {purchase.Price} руб.
({purchase.Category})");
    }
}

private void AddPurchaseButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(nameTextBox.Text) || string.IsNullOrEmpty(priceTextBox.Text))
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    decimal price;
    if (!decimal.TryParse(priceTextBox.Text, out price))
    {
        MessageBox.Show("Неверный формат цены!");
        return;
    }
    Category category = (Category)Enum.Parse(typeof(Category),
categoryComboBox.SelectedItem.ToString());
    DateTime date = datePicker.Value;
    Purchase newPurchase = new Purchase(nameTextBox.Text, price, category, date);
    try
    {
        purchaseManager.AddPurchase(newPurchase);
        nameTextBox.Clear();
        priceTextBox.Clear();
        UpdatePurchasesList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void RemovePurchaseButton_Click(object sender, EventArgs e)
{
    if (purchasesListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите покупку для удаления!");
        return;
    }
    string selectedItem = purchasesListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);

```

```

        if (parts.Length >= 2)
        {
            string name = parts[0].Trim();
            decimal price;
            if (decimal.TryParse(parts[1].Split(' ')[0], out price))
            {
                var purchaseToRemove = purchaseManager.Purchases.Find(p => p.Name ==
name && p.Price == price);
                if (purchaseToRemove != null)
                {
                    try
                    {
                        purchaseManager.RemovePurchase(purchaseToRemove);
                        UpdatePurchasesList();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.Message);
                    }
                }
            }
        }
    }
}

```

```

private void FilterButton_Click(object sender, EventArgs e)
{
    Category category = categoryFilterComboBox.SelectedIndex == 0 ?
Category.Продукты : (Category)Enum.Parse(typeof(Category),
categoryFilterComboBox.SelectedItem.ToString());
    var filteredPurchases = purchaseManager.GetPurchasesByCategory(category);
    purchasesListBox.Items.Clear();
    foreach (var purchase in filteredPurchases)
    {
        purchasesListBox.Items.Add($"{purchase.Name} - {purchase.Price} pyб.
({purchase.Category})");
    }
}

```

```

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new PurchaseForm());
}
}
...

```

Объяснение:

1. Класс `Purchase`:

- Хранит название, цену, категорию и дату покупки.

2. Класс `PurchaseManager`:

- Управляет списком покупок.
- Реализует методы для добавления, удаления и фильтрации покупок по категории.
- Сохраняет и загружает покупки из файла.

3. Графический интерфейс:

- Поля для ввода названия, цены, выбора категории и даты.
- Кнопки для добавления и удаления покупок.
- Комбо-박스 для фильтрации по категории и кнопка для применения фильтра.
- Список покупок.

4. Проверка граничных значений:

- Пустые поля при добавлении покупки.
- Некорректный формат цены.
- Некорректный выбор покупки для удаления.

---

## ### 9. Управление задачами с приоритетом

Описание: Создать программу для управления задачами с приоритетом. Пользователь может добавлять, удалять, редактировать задачи и сортировать их по приоритету.

Программа должна сохранять задачи в файл.

Код программы:

```
```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public enum Priority
{
    Низкий,
    Средний,
    Высокий
}

public class TaskWithPriority
{
    public string Description { get; set; }
```



```

public Priority Priority { get; set; }
public bool IsCompleted { get; set; }
public DateTime Deadline { get; set; }

public TaskWithPriority(string description, Priority priority, DateTime deadline)
{
    Description = description;
    Priority = priority;
    IsCompleted = false;
    Deadline = deadline;
}
}

public class TaskManagerWithPriority
{
    public List<TaskWithPriority> Tasks { get; private set; }

    public TaskManagerWithPriority()
    {
        Tasks = new List<TaskWithPriority>();
        LoadTasks();
    }

    public void AddTask(TaskWithPriority task)
    {
        if (task == null)
        {
            throw new ArgumentNullException(nameof(task));
        }
        Tasks.Add(task);
        SaveTasks();
    }

    public void RemoveTask(TaskWithPriority task)
    {
        if (task == null)
        {
            throw new ArgumentNullException(nameof(task));
        }
        Tasks.Remove(task);
        SaveTasks();
    }

    public void ToggleTaskCompletion(TaskWithPriority task)
    {
        if (task == null)
        {
            throw new ArgumentNullException(nameof(task));
        }
    }
}

```

```

    }
    task.IsCompleted = !task.IsCompleted;
    SaveTasks();
}

public List<TaskWithPriority> SortTasksByPriority()
{
    return Tasks.OrderByDescending(t => t.Priority).ToList();
}

private void SaveTasks()
{
    File.WriteAllLines("tasks.txt", Tasks.Select(t =>
    $"{t.Description}|{(int)t.Priority}|{t.IsCompleted}|{t.Deadline.ToString("yyyy-MM-dd
    HH:mm:ss")}"));
}

private void LoadTasks()
{
    if (File.Exists("tasks.txt"))
    {
        var lines = File.ReadAllLines("tasks.txt");
        foreach (var line in lines)
        {
            var parts = line.Split('|');
            if (parts.Length == 4)
            {
                int priority;
                bool isCompleted;
                DateTime deadline;
                if (int.TryParse(parts[1], out priority) && bool.TryParse(parts[2], out
                isCompleted) && DateTime.TryParse(parts[3], out deadline))
                {
                    TaskWithPriority task = new TaskWithPriority(parts[0], (Priority)priority,
                    deadline);
                    task.IsCompleted = isCompleted;
                    Tasks.Add(task);
                }
            }
        }
    }
}

public class TaskManagerForm : Form
{
    private TaskManagerWithPriority taskManager;
    private TextBox descriptionTextBox;

```

```

private ComboBox priorityComboBox;
private DateTimePicker deadlinePicker;
private Button addTaskButton;
private Button removeTaskButton;
private Button toggleCompletionButton;
private ComboBox sortByPriorityComboBox;
private Button sortByPriorityButton;
private ListBox tasksListBox;

public TaskManagerForm()
{
    this.Text = "Управление задачами с приоритетом";
    this.Width = 600;
    this.Height = 500;

    descriptionTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 10),
        Width = 200,
        PlaceholderText = "Описание"
    };

    priorityComboBox = new ComboBox
    {
        Location = new System.Drawing.Point(220, 10),
        Width = 100,
        Items = { "Низкий", "Средний", "Высокий" }
    };

    deadlinePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(330, 10)
    };

    addTaskButton = new Button
    {
        Location = new System.Drawing.Point(10, 40),
        Text = "Добавить",
        Width = 100
    };
    addTaskButton.Click += AddTaskButton_Click;

    removeTaskButton = new Button
    {
        Location = new System.Drawing.Point(120, 40),
        Text = "Удалить",
        Width = 100
    };
};

```

```

removeTaskButton.Click += RemoveTaskButton_Click;

toggleCompletionButton = new Button
{
    Location = new System.Drawing.Point(220, 40),
    Text = "Отметить",
    Width = 100
};
toggleCompletionButton.Click += ToggleCompletionButton_Click;

sortByPriorityComboBox = new ComboBox
{
    Location = new System.Drawing.Point(10, 70),
    Width = 100,
    Items = { "По приоритету" }
};

sortByPriorityButton = new Button
{
    Location = new System.Drawing.Point(120, 70),
    Text = "Сортировать",
    Width = 100
};
sortByPriorityButton.Click += SortByPriorityButton_Click;

tasksListBox = new ListBox
{
    Location = new System.Drawing.Point(10, 100),
    Width = 560,
    Height = 300
};

this.Controls.Add(descriptionTextBox);
this.Controls.Add(priorityComboBox);
this.Controls.Add(deadlinePicker);
this.Controls.Add(addTaskButton);
this.Controls.Add(removeTaskButton);
this.Controls.Add(toggleCompletionButton);
this.Controls.Add(sortByPriorityComboBox);
this.Controls.Add(sortByPriorityButton);
this.Controls.Add(tasksListBox);

taskManager = new TaskManagerWithPriority();
UpdateTasksList();
}

private void UpdateTasksList()
{

```

```

tasksListBox.Items.Clear();
foreach (var task in taskManager.Tasks)
{
    string status = task.IsCompleted ? "[X]" : "[ ]";
    tasksListBox.Items.Add($"{status} {task.Description} (Приоритет: {task.Priority})");
}
}

private void AddTaskButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(descriptionTextBox.Text))
    {
        MessageBox.Show("Введите описание задачи!");
        return;
    }
    Priority priority = (Priority)Enum.Parse(typeof(Priority),
priorityComboBox.SelectedItem.ToString());
    DateTime deadline = deadlinePicker.Value;
    TaskWithPriority newTask = new TaskWithPriority(descriptionTextBox.Text, priority,
deadline);
    try
    {
        taskManager.AddTask(newTask);
        descriptionTextBox.Clear();
        UpdateTasksList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void RemoveTaskButton_Click(object sender, EventArgs e)
{
    if (tasksListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите задачу для удаления!");
        return;
    }
    string selectedItem = tasksListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
    if (parts.Length >= 3)
    {
        string description = parts[2];
        var taskToRemove = taskManager.Tasks.Find(t => t.Description == description);
        if (taskToRemove != null)
        {
            try

```

```

        {
            taskManager.RemoveTask(taskToRemove);
            UpdateTasksList();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}

```

```

private void ToggleCompletionButton_Click(object sender, EventArgs e)
{
    if (tasksListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите задачу для изменения статуса!");
        return;
    }
    string selectedItem = tasksListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
    if (parts.Length >= 3)
    {
        string description = parts[2];
        var taskToToggle = taskManager.Tasks.Find(t => t.Description == description);
        if (taskToToggle != null)
        {
            try
            {
                taskManager.ToggleTaskCompletion(taskToToggle);
                UpdateTasksList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

```

```

private void SortByPriorityButton_Click(object sender, EventArgs e)
{
    var sortedTasks = taskManager.SortTasksByPriority();
    tasksListBox.Items.Clear();
    foreach (var task in sortedTasks)
    {
        string status = task.IsCompleted ? "[X]" : "[ ]";
        tasksListBox.Items.Add($"{status} {task.Description} (Приоритет: {task.Priority})");
    }
}

```

```

    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new TaskManagerForm());
}
}
...

```

Объяснение:

1. Класс `TaskWithPriority`:

- Хранит описание задачи, приоритет, статус выполнения и срок выполнения.

2. Класс `TaskManagerWithPriority`:

- Управляет списком задач.
- Реализует методы для добавления, удаления, изменения статуса и сортировки задач по приоритету.
- Сохраняет и загружает задачи из файла.

3. Графический интерфейс:

- Поле для ввода описания задачи.
- Комбо- для выбора приоритета.
- Поле для выбора срока выполнения.
- Кнопки для добавления, удаления и изменения статуса задачи.
- Комбо- и кнопка для сортировки задач по приоритету.
- Список задач.

4. Проверка граничных значений:

- Пустое описание задачи.
- Некорректный выбор задачи для удаления или изменения статуса.

---

## ### 10. Управление продажами

Описание: Создать программу для управления продажами. Пользователь может добавлять, удалять, редактировать продажи и генерировать отчёты. Программа должна сохранять данные в файл.

Код программы:

```

```csharp
using System;

```

```

using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public class Sale
{
    public string ProductName { get; set; }
    public decimal Price { get; set; }
    public int Quantity { get; set; }
    public DateTime Date { get; set; }

    public Sale(string productName, decimal price, int quantity, DateTime date)
    {
        ProductName = productName;
        Price = price;
        Quantity = quantity;
        Date = date;
    }

    public decimal TotalRevenue
    {
        get { return Price * Quantity; }
    }
}

public class SaleManager
{
    public List<Sale> Sales { get; private set; }

    public SaleManager()
    {
        Sales = new List<Sale>();
        LoadSales();
    }

    public void AddSale(Sale sale)
    {
        if (sale == null)
        {
            throw new ArgumentNullException(nameof(sale));
        }
        Sales.Add(sale);
        SaveSales();
    }

    public void RemoveSale(Sale sale)
    {

```



```

        if (sale == null)
        {
            throw new ArgumentNullException(nameof(sale));
        }
        Sales.Remove(sale);
        SaveSales();
    }

    public decimal TotalRevenue
    {
        get { return Sales.Sum(s => s.TotalRevenue); }
    }

    private void SaveSales()
    {
        File.WriteAllLines("sales.txt", Sales.Select(s =>
        $"{s.ProductName}|{s.Price}|{s.Quantity}|{s.Date.ToString("yyyy-MM-dd HH:mm:ss")}"));
    }

    private void LoadSales()
    {
        if (File.Exists("sales.txt"))
        {
            var lines = File.ReadAllLines("sales.txt");
            foreach (var line in lines)
            {
                var parts = line.Split('|');
                if (parts.Length == 4)
                {
                    decimal price;
                    int quantity;
                    DateTime date;
                    if (decimal.TryParse(parts[1], out price) && int.TryParse(parts[2], out quantity)
                        && DateTime.TryParse(parts[3], out date))
                    {
                        Sales.Add(new Sale(parts[0], price, quantity, date));
                    }
                }
            }
        }
    }

    public class SaleForm : Form
    {
        private SaleManager saleManager;
        private TextBox productNameTextBox;
        private TextBox priceTextBox;
    }

```

```

private TextBox quantityTextBox;
private DateTimePicker datePicker;
private Button addSaleButton;
private Button removeSaleButton;
private Button generateReportButton;
private ListBox salesListBox;
private Label totalRevenueLabel;

public SaleForm()
{
    this.Text = "Управление продажами";
    this.Width = 600;
    this.Height = 500;

    productNameTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 10),
        Width = 150,
        PlaceholderText = "Название продукта"
    };

    priceTextBox = new TextBox
    {
        Location = new System.Drawing.Point(170, 10),
        Width = 100,
        PlaceholderText = "Цена"
    };

    quantityTextBox = new TextBox
    {
        Location = new System.Drawing.Point(280, 10),
        Width = 80,
        PlaceholderText = "Количество"
    };

    datePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(370, 10)
    };

    addSaleButton = new Button
    {
        Location = new System.Drawing.Point(10, 40),
        Text = "Добавить",
        Width = 100
    };
    addSaleButton.Click += AddSaleButton_Click;
}

```

```

removeSaleButton = new Button
{
    Location = new System.Drawing.Point(120, 40),
    Text = "Удалить",
    Width = 100
};
removeSaleButton.Click += RemoveSaleButton_Click;

generateReportButton = new Button
{
    Location = new System.Drawing.Point(220, 40),
    Text = "Сформировать отчёт",
    Width = 120
};
generateReportButton.Click += GenerateReportButton_Click;

salesListBox = new ListBox
{
    Location = new System.Drawing.Point(10, 70),
    Width = 560,
    Height = 200
};

totalRevenueLabel = new Label
{
    Location = new System.Drawing.Point(10, 280),
    Width = 200,
    Text = "Общий доход: "
};

this.Controls.Add(productNameTextBox);
this.Controls.Add(priceTextBox);
this.Controls.Add(quantityTextBox);
this.Controls.Add(datePicker);
this.Controls.Add(addSaleButton);
this.Controls.Add(removeSaleButton);
this.Controls.Add(generateReportButton);
this.Controls.Add(salesListBox);
this.Controls.Add(totalRevenueLabel);

saleManager = new SaleManager();
UpdateSalesList();
UpdateTotalRevenue();
}

private void UpdateSalesList()
{
    salesListBox.Items.Clear();

```

```

        foreach (var sale in saleManager.Sales)
        {
            salesListBox.Items.Add($"{sale.ProductName} - {sale.Price} руб. x {sale.Quantity} = {sale.TotalRevenue} руб.");
        }
    }

    private void UpdateTotalRevenue()
    {
        totalRevenueLabel.Text = $"Общий доход: {saleManager.TotalRevenue} руб.";
    }

    private void AddSaleButton_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(productNameTextBox.Text) ||
            string.IsNullOrEmpty(priceTextBox.Text) || string.IsNullOrEmpty(quantityTextBox.Text))
        {
            MessageBox.Show("Заполните все поля!");
            return;
        }
        decimal price;
        int quantity;
        if (!decimal.TryParse(priceTextBox.Text, out price) || !int.TryParse(quantityTextBox.Text,
            out quantity))
        {
            MessageBox.Show("Неверный формат цены или количества!");
            return;
        }
        DateTime date = datePicker.Value;
        Sale newSale = new Sale(productNameTextBox.Text, price, quantity, date);
        try
        {
            saleManager.AddSale(newSale);
            productNameTextBox.Clear();
            priceTextBox.Clear();
            quantityTextBox.Clear();
            UpdateSalesList();
            UpdateTotalRevenue();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void RemoveSaleButton_Click(object sender, EventArgs e)
    {
        if (salesListBox.SelectedIndex == -1)

```

```

{
    MessageBox.Show("Выберите продажу для удаления!");
    return;
}
string selectedItem = salesListBox.SelectedItem.ToString();
string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
if (parts.Length >= 2)
{
    string productName = parts[0].Trim();
    decimal price;
    if (decimal.TryParse(parts[1].Split(' ')[0], out price))
    {
        var saleToRemove = saleManager.Sales.Find(s => s.ProductName ==
productName && s.Price == price);
        if (saleToRemove != null)
        {
            try
            {
                saleManager.RemoveSale(saleToRemove);
                UpdateSalesList();
                UpdateTotalRevenue();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}
}

private void GenerateReportButton_Click(object sender, EventArgs e)
{
    if (saleManager.Sales.Count == 0)
    {
        MessageBox.Show("Нет продаж для генерации отчёта!");
        return;
    }
    string report = "Отчёт по продажам:\n";
    foreach (var sale in saleManager.Sales)
    {
        report += $"Продукт: {sale.ProductName}\nЦена: {sale.Price} руб.\nКоличество:
{sale.Quantity}\nДата: {sale.Date.ToString("yyyy-MM-dd")}\nОбщий доход:
{sale.TotalRevenue} руб.\n\n";
    }
    report += $"Итого: {saleManager.TotalRevenue} руб.";
    File.WriteAllText("sales_report.txt", report);
    MessageBox.Show("Отчёт сформирован и сохранён в файл sales_report.txt!");
}

```

```

    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new SaleForm());
    }
}
...

```

Объяснение:

1. Класс `Sale`:

- Хранит название продукта, цену, количество и дату продажи.
- Вычисляет общий доход от продажи.

2. Класс `SaleManager`:

- Управляет списком продаж.
- Реализует методы для добавления, удаления продаж и генерации отчёта.
- Сохраняет и загружает продажи из файла.

3. Графический интерфейс:

- Поля для ввода названия продукта, цены, количества и выбора даты.
- Кнопки для добавления, удаления продаж и генерации отчёта.
- Список продаж.
- Отображение общего дохода.

4. Проверка граничных значений:

- Пустые поля при добавлении продажи.
- Некорректный формат цены или количества.
- Некорректный выбор продажи для удаления.

---

## ### 11. Управление инвентарём

Описание: Создать программу для управления инвентарём. Пользователь может добавлять, удалять, редактировать товары и отслеживать их количество. Программа должна сохранять данные в файл.

Код программы:

```

```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;

```

```
using System.IO;
using System.Linq;
```

```
public class InventoryItem
{
    public string Name { get; set; }
    public int Quantity { get; set; }
    public decimal Price { get; set; }
    public string Category { get; set; }

    public InventoryItem(string name, int quantity, decimal price, string category)
    {
        Name = name;
        Quantity = quantity;
        Price = price;
        Category = category;
    }
}
```

```
public class InventoryManager
{
    public List<InventoryItem> Items { get; private set; }

    public InventoryManager()
    {
        Items = new List<InventoryItem>();
        LoadItems();
    }

    public void AddItem(InventoryItem item)
    {
        if (item == null)
        {
            throw new ArgumentNullException(nameof(item));
        }
        Items.Add(item);
        SaveItems();
    }

    public void RemoveItem(InventoryItem item)
    {
        if (item == null)
        {
            throw new ArgumentNullException(nameof(item));
        }
        Items.Remove(item);
        SaveItems();
    }
}
```

```

public void UpdateItemQuantity(InventoryItem item, int newQuantity)
{
    if (item == null)
    {
        throw new ArgumentNullException(nameof(item));
    }
    item.Quantity = newQuantity;
    SaveItems();
}

private void SaveItems()
{
    File.WriteAllLines("inventory.txt", Items.Select(i =>
$"{i.Name}|{i.Quantity}|{i.Price}|{i.Category}"));
}

private void LoadItems()
{
    if (File.Exists("inventory.txt"))
    {
        var lines = File.ReadAllLines("inventory.txt");
        foreach (var line in lines)
        {
            var parts = line.Split('|');
            if (parts.Length == 4)
            {
                int quantity;
                decimal price;
                if (int.TryParse(parts[1], out quantity) && decimal.TryParse(parts[2], out price))
                {
                    Items.Add(new InventoryItem(parts[0], quantity, price, parts[3]));
                }
            }
        }
    }
}

public class InventoryForm : Form
{
    private InventoryManager inventoryManager;
    private TextBox nameTextBox;
    private TextBox quantityTextBox;
    private TextBox priceTextBox;
    private TextBox categoryTextBox;
    private Button addItemButton;
    private Button removeItemButton;
}

```



```

private Button updateQuantityButton;
private ListBox itemsListBox;

public InventoryForm()
{
    this.Text = "Управление инвентарём";
    this.Width = 500;
    this.Height = 400;

    nameTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 10),
        Width = 150,
        PlaceholderText = "Название"
    };

    quantityTextBox = new TextBox
    {
        Location = new System.Drawing.Point(170, 10),
        Width = 80,
        PlaceholderText = "Количество"
    };

    priceTextBox = new TextBox
    {
        Location = new System.Drawing.Point(260, 10),
        Width = 100,
        PlaceholderText = "Цена"
    };

    categoryTextBox = new TextBox
    {
        Location = new System.Drawing.Point(370, 10),
        Width = 100,
        PlaceholderText = "Категория"
    };

    addItemButton = new Button
    {
        Location = new System.Drawing.Point(10, 40),
        Text = "Добавить",
        Width = 100
    };
    addItemButton.Click += AddItemButton_Click;

    removeItemButton = new Button
    {
        Location = new System.Drawing.Point(120, 40),

```

```

        Text = "Удалить",
        Width = 100
    };
    removeItemButton.Click += RemoveItemButton_Click;

    updateQuantityButton = new Button
    {
        Location = new System.Drawing.Point(220, 40),
        Text = "Обновить",
        Width = 100
    };
    updateQuantityButton.Click += UpdateQuantityButton_Click;

    itemsListBox = new ListBox
    {
        Location = new System.Drawing.Point(10, 70),
        Width = 460,
        Height = 200
    };

    this.Controls.Add(nameTextBox);
    this.Controls.Add(quantityTextBox);
    this.Controls.Add(priceTextBox);
    this.Controls.Add(categoryTextBox);
    this.Controls.Add(addItemButton);
    this.Controls.Add(removeItemButton);
    this.Controls.Add(updateQuantityButton);
    this.Controls.Add(itemsListBox);

    inventoryManager = new InventoryManager();
    UpdateItemsList();
}

private void UpdateItemsList()
{
    itemsListBox.Items.Clear();
    foreach (var item in inventoryManager.Items)
    {
        itemsListBox.Items.Add($"{item.Name} - Количество: {item.Quantity} | Цена:
{item.Price} руб. | Категория: {item.Category}");
    }
}

private void AddItemButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(nameTextBox.Text) ||
    string.IsNullOrEmpty(quantityTextBox.Text) || string.IsNullOrEmpty(priceTextBox.Text) ||
    string.IsNullOrEmpty(categoryTextBox.Text))

```

```

    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    int quantity;
    decimal price;
    if (!int.TryParse(quantityTextBox.Text, out quantity) ||
        !decimal.TryParse(priceTextBox.Text, out price))
    {
        MessageBox.Show("Неверный формат количества или цены!");
        return;
    }
    InventoryItem newItem = new InventoryItem(nameTextBox.Text, quantity, price,
categoryTextBox.Text);
    try
    {
        inventoryManager.AddItem(newItem);
        nameTextBox.Clear();
        quantityTextBox.Clear();
        priceTextBox.Clear();
        categoryTextBox.Clear();
        UpdateItemsList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

```

```

private void RemoveItemButton_Click(object sender, EventArgs e)
{
    if (itemsListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите товар для удаления!");
        return;
    }
    string selectedItem = itemsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string name = parts[0].Trim();
        var itemToRemove = inventoryManager.Items.Find(i => i.Name == name);
        if (itemToRemove != null)
        {
            try
            {
                inventoryManager.RemoveItem(itemToRemove);
                UpdateItemsList();
            }
            catch { }
        }
    }
}

```

```

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
}

private void UpdateQuantityButton_Click(object sender, EventArgs e)
{
    if (itemsListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите товар для обновления!");
        return;
    }
    string selectedItem = itemsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string name = parts[0].Trim();
        var itemToUpdate = inventoryManager.Items.Find(i => i.Name == name);
        if (itemToUpdate != null)
        {
            if (string.IsNullOrEmpty(quantityTextBox.Text))
            {
                MessageBox.Show("Введите новое количество!");
                return;
            }
            int newQuantity;
            if (!int.TryParse(quantityTextBox.Text, out newQuantity))
            {
                MessageBox.Show("Неверный формат количества!");
                return;
            }
            try
            {
                inventoryManager.UpdateItemQuantity(itemToUpdate, newQuantity);
                UpdateItemsList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}
}

```

```

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new InventoryForm());
}
}
...

```

Объяснение:

1. Класс `InventoryItem`:

- Хранит название, количество, цену и категорию товара.

2. Класс `InventoryManager`:

- Управляет списком товаров.
- Реализует методы для добавления, удаления и обновления количества товаров.
- Сохраняет и загружает товары из файла.

3. Графический интерфейс:

- Поля для ввода названия, количества, цены и категории.
- Кнопки для добавления, удаления и обновления количества товаров.
- Список товаров.

4. Проверка граничных значений:

- Пустые поля при добавлении товара.
- Некорректный формат количества или цены.
- Некорректный выбор товара для удаления или обновления.

---

## ### 12. Управление заказами

Описание: Создать программу для управления заказами. Пользователь может добавлять, удалять, редактировать заказы и отслеживать их статус. Программа должна сохранять данные в файл.

Код программы:

```

```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public enum OrderStatus

```

```

{
    Новый,
    В_обработке,
    Завершён
}

public class Order
{
    public string CustomerName { get; set; }
    public string Description { get; set; }
    public OrderStatus Status { get; set; }
    public DateTime CreationDate { get; set; }

    public Order(string customerName, string description, DateTime creationDate)
    {
        CustomerName = customerName;
        Description = description;
        Status = OrderStatus.Новый;
        CreationDate = creationDate;
    }

    public void UpdateStatus(OrderStatus newStatus)
    {
        Status = newStatus;
    }
}

public class OrderManager
{
    public List<Order> Orders { get; private set; }

    public OrderManager()
    {
        Orders = new List<Order>();
        LoadOrders();
    }

    public void AddOrder(Order order)
    {
        if (order == null)
        {
            throw new ArgumentNullException(nameof(order));
        }
        Orders.Add(order);
        SaveOrders();
    }

    public void RemoveOrder(Order order)

```

```

{
    if (order == null)
    {
        throw new ArgumentNullException(nameof(order));
    }
    Orders.Remove(order);
    SaveOrders();
}

public void UpdateOrderStatus(Order order, OrderStatus newStatus)
{
    if (order == null)
    {
        throw new ArgumentNullException(nameof(order));
    }
    order.UpdateStatus(newStatus);
    SaveOrders();
}

private void SaveOrders()
{
    File.WriteAllLines("orders.txt", Orders.Select(o =>
    $"{o.CustomerName}|{o.Description}|{(int)o.Status}|{o.CreationDate.ToString("yyyy-MM-dd
    HH:mm:ss")}"));
}

private void LoadOrders()
{
    if (File.Exists("orders.txt"))
    {
        var lines = File.ReadAllLines("orders.txt");
        foreach (var line in lines)
        {
            var parts = line.Split('|');
            if (parts.Length == 4)
            {
                OrderStatus status = (OrderStatus)Enum.Parse(typeof(OrderStatus), parts[2]);
                DateTime date;
                if (DateTime.TryParse(parts[3], out date))
                {
                    Orders.Add(new Order(parts[0], parts[1], date));
                    Orders.Last().Status = status;
                }
            }
        }
    }
}
}

```

```

public class OrderForm : Form
{
    private OrderManager orderManager;
    private TextBox customerNameTextBox;
    private TextBox descriptionTextBox;
    private DateTimePicker creationDatePicker;
    private ComboBox statusComboBox;
    private Button addOrderButton;
    private Button removeOrderButton;
    private Button updateStatusButton;
    private ListBox ordersListBox;

    public OrderForm()
    {
        this.Text = "Управление заказами";
        this.Width = 600;
        this.Height = 500;

        customerNameTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 10),
            Width = 150,
            PlaceholderText = "Имя клиента"
        };

        descriptionTextBox = new TextBox
        {
            Location = new System.Drawing.Point(170, 10),
            Width = 200,
            PlaceholderText = "Описание"
        };

        creationDatePicker = new DateTimePicker
        {
            Location = new System.Drawing.Point(380, 10)
        };

        addOrderButton = new Button
        {
            Location = new System.Drawing.Point(10, 40),
            Text = "Добавить",
            Width = 100
        };
        addOrderButton.Click += AddOrderButton_Click;

        removeOrderButton = new Button
        {

```



```

        Location = new System.Drawing.Point(120, 40),
        Text = "Удалить",
        Width = 100
    };
    removeOrderButton.Click += RemoveOrderButton_Click;

    updateStatusButton = new Button
    {
        Location = new System.Drawing.Point(220, 40),
        Text = "Обновить статус",
        Width = 120
    };
    updateStatusButton.Click += UpdateStatusButton_Click;

    statusComboBox = new ComboBox
    {
        Location = new System.Drawing.Point(340, 40),
        Width = 100,
        Items = { "Новый", "В обработке", "Завершён" }
    };

    ordersListBox = new ListBox
    {
        Location = new System.Drawing.Point(10, 70),
        Width = 560,
        Height = 300
    };

    this.Controls.Add(customerNameTextBox);
    this.Controls.Add(descriptionTextBox);
    this.Controls.Add(creationDatePicker);
    this.Controls.Add(addOrderButton);
    this.Controls.Add(removeOrderButton);
    this.Controls.Add(updateStatusButton);
    this.Controls.Add(statusComboBox);
    this.Controls.Add(ordersListBox);

    orderManager = new OrderManager();
    UpdateOrdersList();
}

private void UpdateOrdersList()
{
    ordersListBox.Items.Clear();
    foreach (var order in orderManager.Orders)
    {
        ordersListBox.Items.Add($"{order.CustomerName} - {order.Description}
({order.Status})");
    }
}

```

```

    }
}

private void AddOrderButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(customerNameTextBox.Text) ||
string.IsNullOrEmpty(descriptionTextBox.Text))
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    DateTime creationDate = creationDatePicker.Value;
    Order newOrder = new Order(customerNameTextBox.Text, descriptionTextBox.Text,
creationDate);
    try
    {
        orderManager.AddOrder(newOrder);
        customerNameTextBox.Clear();
        descriptionTextBox.Clear();
        UpdateOrdersList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void RemoveOrderButton_Click(object sender, EventArgs e)
{
    if (ordersListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите заказ для удаления!");
        return;
    }
    string selectedItem = ordersListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string customerName = parts[0].Trim();
        string description = parts[1].Trim();
        var orderToRemove = orderManager.Orders.Find(o => o.CustomerName ==
customerName && o.Description == description);
        if (orderToRemove != null)
        {
            try
            {
                orderManager.RemoveOrder(orderToRemove);
                UpdateOrdersList();
            }

```

```

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
}

private void UpdateStatusButton_Click(object sender, EventArgs e)
{
    if (ordersListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите заказ для обновления статуса!");
        return;
    }
    string selectedItem = ordersListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string customerName = parts[0].Trim();
        string description = parts[1].Trim();
        var orderToUpdate = orderManager.Orders.Find(o => o.CustomerName ==
customerName && o.Description == description);
        if (orderToUpdate != null)
        {
            OrderStatus newStatus = (OrderStatus)Enum.Parse(typeof(OrderStatus),
statusComboBox.SelectedItem.ToString());
            try
            {
                orderManager.UpdateOrderStatus(orderToUpdate, newStatus);
                UpdateOrdersList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new OrderForm());
}

```

```
}  
...
```

Объяснение:

1. Класс `Order`:

- Хранит имя клиента, описание заказа, статус и дату создания.

2. Класс `OrderManager`:

- Управляет списком заказов.
- Реализует методы для добавления, удаления и обновления статуса заказов.
- Сохраняет и загружает заказы из файла.

3. Графический интерфейс:

- Поля для ввода имени клиента, описания заказа и выбора даты.
- Кнопки для добавления, удаления и обновления статуса заказов.
- Комбо-**박스** для выбора нового статуса.
- Список заказов.

4. Проверка граничных значений:

- Пустые поля при добавлении заказа.
- Некорректный выбор заказа для удаления или обновления статуса.

---

### ### 13. Управление сотрудниками

Описание: Создать программу для управления сотрудниками. Пользователь может добавлять, удалять, редактировать данные сотрудников и отслеживать их отпуска. Программа должна сохранять данные в файл.

Код программы:

```
```csharp  
using System;  
using System.Windows.Forms;  
using System.Collections.Generic;  
using System.IO;  
using System.Linq;  
  
public class Employee  
{  
    public string Name { get; set; }  
    public string Position { get; set; }  
    public DateTime HireDate { get; set; }  
    public DateTime? VacationStart { get; set; }  
    public DateTime? VacationEnd { get; set; }  
}
```

```

public Employee(string name, string position, DateTime hireDate)
{
    Name = name;
    Position = position;
    HireDate = hireDate;
    VacationStart = null;
    VacationEnd = null;
}

public bool IsOnVacation
{
    get
    {
        if (VacationStart == null || VacationEnd == null)
        {
            return false;
        }
        return DateTime.Now >= VacationStart.Value && DateTime.Now <=
VacationEnd.Value;
    }
}

public class EmployeeManager
{
    public List<Employee> Employees { get; private set; }

    public EmployeeManager()
    {
        Employees = new List<Employee>();
        LoadEmployees();
    }

    public void AddEmployee(Employee employee)
    {
        if (employee == null)
        {
            throw new ArgumentNullException(nameof(employee));
        }
        Employees.Add(employee);
        SaveEmployees();
    }

    public void RemoveEmployee(Employee employee)
    {
        if (employee == null)
        {
            throw new ArgumentNullException(nameof(employee));
        }
    }
}

```

```

    }
    Employees.Remove(employee);
    SaveEmployees();
}

public void UpdateVacation(Employee employee, DateTime? vacationStart, DateTime?
vacationEnd)
{
    if (employee == null)
    {
        throw new ArgumentNullException(nameof(employee));
    }
    employee.VacationStart = vacationStart;
    employee.VacationEnd = vacationEnd;
    SaveEmployees();
}

private void SaveEmployees()
{
    File.WriteAllLines("employees.txt", Employees.Select(e =>
"${e.Name}|{e.Position}|{e.HireDate.ToString("yyyy-MM-dd")}|{e.VacationStart?.ToString("yyy
y-MM-dd")}|{e.VacationEnd?.ToString("yyyy-MM-dd")}");
}

private void LoadEmployees()
{
    if (File.Exists("employees.txt"))
    {
        var lines = File.ReadAllLines("employees.txt");
        foreach (var line in lines)
        {
            var parts = line.Split('|');
            if (parts.Length == 5)
            {
                DateTime hireDate;
                DateTime? vacationStart = null;
                DateTime? vacationEnd = null;
                if (DateTime.TryParse(parts[2], out hireDate))
                {
                    if (parts[3] != "")
                    {
                        if (DateTime.TryParse(parts[3], out DateTime temp))
                        {
                            vacationStart = temp;
                        }
                    }
                    if (parts[4] != "")
                    {

```

```

        if (DateTime.TryParse(parts[4], out DateTime temp))
        {
            vacationEnd = temp;
        }
    }
    Employees.Add(new Employee(parts[0], parts[1], hireDate)
    {
        VacationStart = vacationStart,
        VacationEnd = vacationEnd
    });
}
}
}
}
}
}
}
}
}
}

```

```

public class EmployeeForm : Form
{
    private EmployeeManager employeeManager;
    private TextBox nameTextBox;
    private TextBox positionTextBox;
    private DateTimePicker hireDatePicker;
    private DateTimePicker vacationStartPicker;
    private DateTimePicker vacationEndPicker;
    private Button addEmployeeButton;
    private Button removeEmployeeButton;
    private Button updateVacationButton;
    private ListBox employeesListBox;

    public EmployeeForm()
    {
        this.Text = "Управление сотрудниками";
        this.Width = 600;
        this.Height = 500;

        nameTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 10),
            Width = 150,
            PlaceholderText = "Имя"
        };

        positionTextBox = new TextBox
        {
            Location = new System.Drawing.Point(170, 10),
            Width = 150,
            PlaceholderText = "Должность"
        };
    }
}

```

```

};

hireDatePicker = new DateTimePicker
{
    Location = new System.Drawing.Point(330, 10)
};

vacationStartPicker = new DateTimePicker
{
    Location = new System.Drawing.Point(10, 40)
};

vacationEndPicker = new DateTimePicker
{
    Location = new System.Drawing.Point(170, 40)
};

addEmployeeButton = new Button
{
    Location = new System.Drawing.Point(10, 70),
    Text = "Добавить",
    Width = 100
};
addEmployeeButton.Click += AddEmployeeButton_Click;

removeEmployeeButton = new Button
{
    Location = new System.Drawing.Point(120, 70),
    Text = "Удалить",
    Width = 100
};
removeEmployeeButton.Click += RemoveEmployeeButton_Click;

updateVacationButton = new Button
{
    Location = new System.Drawing.Point(220, 70),
    Text = "Обновить отпуск",
    Width = 120
};
updateVacationButton.Click += UpdateVacationButton_Click;

employeesListBox = new ListBox
{
    Location = new System.Drawing.Point(10, 100),
    Width = 560,
    Height = 250
};

```



```

this.Controls.Add(nameTextBox);
this.Controls.Add(positionTextBox);
this.Controls.Add(hireDatePicker);
this.Controls.Add(vacationStartPicker);
this.Controls.Add(vacationEndPicker);
this.Controls.Add(addEmployeeButton);
this.Controls.Add(removeEmployeeButton);
this.Controls.Add(updateVacationButton);
this.Controls.Add(employeesListBox);

employeeManager = new EmployeeManager();
UpdateEmployeesList();
}

private void UpdateEmployeesList()
{
    employeesListBox.Items.Clear();
    foreach (var employee in employeeManager.Employees)
    {
        string vacationStatus = employee.IsOnVacation ? "В отпуске" : "На работе";
        employeesListBox.Items.Add($"{employee.Name} - {employee.Position}
({{vacationStatus}})");
    }
}

private void AddEmployeeButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(nameTextBox.Text) ||
string.IsNullOrEmpty(positionTextBox.Text))
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    DateTime hireDate = hireDatePicker.Value;
    Employee newEmployee = new Employee(nameTextBox.Text, positionTextBox.Text,
hireDate);
    try
    {
        employeeManager.AddEmployee(newEmployee);
        nameTextBox.Clear();
        positionTextBox.Clear();
        UpdateEmployeesList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

```

```

private void RemoveEmployeeButton_Click(object sender, EventArgs e)
{
    if (employeesListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите сотрудника для удаления!");
        return;
    }
    string selectedItem = employeesListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string name = parts[0].Trim();
        string position = parts[1].Trim();
        var employeeToRemove = employeeManager.Employees.Find(e => e.Name ==
name && e.Position == position);
        if (employeeToRemove != null)
        {
            try
            {
                employeeManager.RemoveEmployee(employeeToRemove);
                UpdateEmployeesList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

```

```

private void UpdateVacationButton_Click(object sender, EventArgs e)
{
    if (employeesListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите сотрудника для обновления отпуска!");
        return;
    }
    string selectedItem = employeesListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string name = parts[0].Trim();
        string position = parts[1].Trim();
        var employeeToUpdate = employeeManager.Employees.Find(e => e.Name == name
&& e.Position == position);
        if (employeeToUpdate != null)
        {

```

```

        DateTime? vacationStart = vacationStartPicker.Value;
        DateTime? vacationEnd = vacationEndPicker.Value;
        if (vacationStart >= vacationEnd)
        {
            MessageBox.Show("Дата начала должна быть раньше даты окончания!");
            return;
        }
        try
        {
            employeeManager.UpdateVacation(employeeToUpdate, vacationStart,
vacationEnd);
            UpdateEmployeesList();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new EmployeeForm());
}
...

```

Объяснение:

1. Класс `Employee`:

- Хранит имя, должность, дату приёма на работу, даты отпуска и статус нахождения в отпуске.

2. Класс `EmployeeManager`:

- Управляет списком сотрудников.
- Реализует методы для добавления, удаления и обновления данных о отпусках.
- Сохраняет и загружает сотрудников из файла.

3. Графический интерфейс:

- Поля для ввода имени, должности и выбора даты приёма.
- Поля для выбора дат начала и окончания отпуска.
- Кнопки для добавления, удаления и обновления отпусков.
- Список сотрудников с их статусом.

4. Проверка граничных значений:

- Пустые поля при добавлении сотрудника.
- Некорректный выбор сотрудника для удаления или обновления данных.
- Некорректные даты отпуска (дата начала после даты окончания).

---

## ### 14. Управление проектами

Описание: Создать программу для управления проектами. Пользователь может добавлять, удалять, редактировать проекты и отслеживать их прогресс. Программа должна сохранять данные в файл.

Код программы:

```

```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public class Project
{
    public string Name { get; set; }
    public string Description { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public int Progress { get; set; }

    public Project(string name, string description, DateTime startDate, DateTime endDate)
    {
        Name = name;
        Description = description;
        StartDate = startDate;
        EndDate = endDate;
        Progress = 0;
    }

    public void UpdateProgress(int newProgress)
    {
        if (newProgress < 0 || newProgress > 100)
        {
            throw new ArgumentOutOfRangeException(nameof(newProgress), "Прогресс
должен быть между 0 и 100.");
        }
        Progress = newProgress;
    }
}

```

```

}

public class ProjectManager
{
    public List<Project> Projects { get; private set; }

    public ProjectManager()
    {
        Projects = new List<Project>();
        LoadProjects();
    }

    public void AddProject(Project project)
    {
        if (project == null)
        {
            throw new ArgumentNullException(nameof(project));
        }
        Projects.Add(project);
        SaveProjects();
    }

    public void RemoveProject(Project project)
    {
        if (project == null)
        {
            throw new ArgumentNullException(nameof(project));
        }
        Projects.Remove(project);
        SaveProjects();
    }

    public void UpdateProjectProgress(Project project, int newProgress)
    {
        if (project == null)
        {
            throw new ArgumentNullException(nameof(project));
        }
        project.UpdateProgress(newProgress);
        SaveProjects();
    }

    private void SaveProjects()
    {
        File.WriteAllLines("projects.txt", Projects.Select(p =>
        $"{p.Name}|{p.Description}|{p.StartDate.ToString("yyyy-MM-dd")}|{p.EndDate.ToString("yyyy-
        MM-dd")}|{p.Progress}"));
    }
}

```

```

private void LoadProjects()
{
    if (File.Exists("projects.txt"))
    {
        var lines = File.ReadAllLines("projects.txt");
        foreach (var line in lines)
        {
            var parts = line.Split('|');
            if (parts.Length == 5)
            {
                DateTime startDate;
                DateTime endDate;
                int progress;
                if (DateTime.TryParse(parts[2], out startDate) && DateTime.TryParse(parts[3],
out endDate) && int.TryParse(parts[4], out progress))
                {
                    Project project = new Project(parts[0], parts[1], startDate, endDate);
                    project.Progress = progress;
                    Projects.Add(project);
                }
            }
        }
    }
}

```

```

public class ProjectForm : Form
{
    private ProjectManager projectManager;
    private TextBox nameTextBox;
    private TextBox descriptionTextBox;
    private DateTimePicker startDatePicker;
    private DateTimePicker endDatePicker;
    private TextBox progressTextBox;
    private Button addProjectButton;
    private Button removeProjectButton;
    private Button updateProgressButton;
    private ListBox projectsListBox;

    public ProjectForm()
    {
        this.Text = "Управление проектами";
        this.Width = 600;
        this.Height = 500;

        nameTextBox = new TextBox
        {

```

```

        Location = new System.Drawing.Point(10, 10),
        Width = 150,
        PlaceholderText = "Название"
    };

    descriptionTextBox = new TextBox
    {
        Location = new System.Drawing.Point(170, 10),
        Width = 200,
        PlaceholderText = "Описание"
    };

    startDatePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(10, 40)
    };

    endDatePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(170, 40)
    };

    progressTextBox = new TextBox
    {
        Location = new System.Drawing.Point(380, 40),
        Width = 50,
        PlaceholderText = "Прогресс"
    };

    addProjectButton = new Button
    {
        Location = new System.Drawing.Point(10, 70),
        Text = "Добавить",
        Width = 100
    };
    addProjectButton.Click += AddProjectButton_Click;

    removeProjectButton = new Button
    {
        Location = new System.Drawing.Point(120, 70),
        Text = "Удалить",
        Width = 100
    };
    removeProjectButton.Click += RemoveProjectButton_Click;

    updateProgressButton = new Button
    {
        Location = new System.Drawing.Point(220, 70),

```

```

        Text = "Обновить прогресс",
        Width = 120
    };
    updateProgressButton.Click += UpdateProgressButton_Click;

    projectsListBox = new ListBox
    {
        Location = new System.Drawing.Point(10, 100),
        Width = 560,
        Height = 250
    };

    this.Controls.Add(nameTextBox);
    this.Controls.Add(descriptionTextBox);
    this.Controls.Add(startDatePicker);
    this.Controls.Add(endDatePicker);
    this.Controls.Add(progressTextBox);
    this.Controls.Add(addProjectButton);
    this.Controls.Add(removeProjectButton);
    this.Controls.Add(updateProgressButton);
    this.Controls.Add(projectsListBox);

    projectManager = new ProjectManager();
    UpdateProjectsList();
}

private void UpdateProjectsList()
{
    projectsListBox.Items.Clear();
    foreach (var project in projectManager.Projects)
    {
        projectsListBox.Items.Add($"{project.Name} - Прогресс: {project.Progress}%");
    }
}

private void AddProjectButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(nameTextBox.Text) ||
        string.IsNullOrEmpty(descriptionTextBox.Text))
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    DateTime startDate = startDatePicker.Value;
    DateTime endDate = endDatePicker.Value;
    if (startDate > endDate)
    {
        MessageBox.Show("Дата начала должна быть раньше даты окончания!");
    }
}

```



```

        return;
    }
    Project newProject = new Project(nameTextBox.Text, descriptionTextBox.Text,
startDate, endDate);
    try
    {
        projectManager.AddProject(newProject);
        nameTextBox.Clear();
        descriptionTextBox.Clear();
        UpdateProjectsList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void RemoveProjectButton_Click(object sender, EventArgs e)
{
    if (projectsListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите проект для удаления!");
        return;
    }
    string selectedItem = projectsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string name = parts[0].Trim();
        var projectToRemove = projectManager.Projects.Find(p => p.Name == name);
        if (projectToRemove != null)
        {
            try
            {
                projectManager.RemoveProject(projectToRemove);
                UpdateProjectsList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

private void UpdateProgressButton_Click(object sender, EventArgs e)
{
    if (projectsListBox.SelectedIndex == -1)

```

```

    {
        MessageBox.Show("Выберите проект для обновления прогресса!");
        return;
    }
    string selectedItem = projectsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string name = parts[0].Trim();
        var projectToUpdate = projectManager.Projects.Find(p => p.Name == name);
        if (projectToUpdate != null)
        {
            if (string.IsNullOrEmpty(progressTextBox.Text))
            {
                MessageBox.Show("Введите новый прогресс!");
                return;
            }
            int newProgress;
            if (!int.TryParse(progressTextBox.Text, out newProgress))
            {
                MessageBox.Show("Неверный формат прогресса!");
                return;
            }
            try
            {
                projectManager.UpdateProjectProgress(projectToUpdate, newProgress);
                UpdateProjectsList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new ProjectForm());
}
...

```

Объяснение:

1. Класс `Project`:

- Хранит название, описание, даты начала и окончания, и прогресс.

## 2. Класс `ProjectManager`:

- Управляет списком проектов.
- Реализует методы для добавления, удаления и обновления прогресса проектов.
- Сохраняет и загружает проекты из файла.

## 3. Графический интерфейс:

- Поля для ввода названия, описания, выбора дат начала и окончания.
- Поле для ввода прогресса.
- Кнопки для добавления, удаления и обновления прогресса проектов.
- Список проектов.

## 4. Проверка граничных значений:

- Пустые поля при добавлении проекта.
- Некорректные даты (начало после окончания).
- Прогресс за пределами 0-100%.
- Некорректный выбор проекта для удаления или обновления.

---

# ### 15. Управление клиентами

Описание: Создать программу для управления клиентами. Пользователь может добавлять, удалять, редактировать данные клиентов и искать по различным критериям. Программа должна сохранять данные в файл.

Код программы:

```
```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public class Client
{
    public string Name { get; set; }
    public string Email { get; set; }
    public string Phone { get; set; }
    public string Address { get; set; }

    public Client(string name, string email, string phone, string address)
    {
        Name = name;
        Email = email;
    }
}
```

```

        Phone = phone;
        Address = address;
    }
}

public class ClientManager
{
    public List<Client> Clients { get; private set; }

    public ClientManager()
    {
        Clients = new List<Client>();
        LoadClients();
    }

    public void AddClient(Client client)
    {
        if (client == null)
        {
            throw new ArgumentNullException(nameof(client));
        }
        Clients.Add(client);
        SaveClients();
    }

    public void RemoveClient(Client client)
    {
        if (client == null)
        {
            throw new ArgumentNullException(nameof(client));
        }
        Clients.Remove(client);
        SaveClients();
    }

    public List<Client> SearchClients(string query)
    {
        return Clients.Where(c => c.Name.Contains(query) || c.Email.Contains(query) ||
c.Phone.Contains(query) || c.Address.Contains(query)).ToList();
    }

    private void SaveClients()
    {
        File.WriteAllLines("clients.txt", Clients.Select(c =>
$"{{c.Name}}|{{c.Email}}|{{c.Phone}}|{{c.Address}}"));
    }

    private void LoadClients()

```

```

{
    if (File.Exists("clients.txt"))
    {
        var lines = File.ReadAllLines("clients.txt");
        foreach (var line in lines)
        {
            var parts = line.Split('|');
            if (parts.Length == 4)
            {
                Clients.Add(new Client(parts[0], parts[1], parts[2], parts[3]));
            }
        }
    }
}
}

```

```

public class ClientForm : Form
{
    private ClientManager clientManager;
    private TextBox nameTextBox;
    private TextBox emailTextBox;
    private TextBox phoneTextBox;
    private TextBox addressTextBox;
    private Button addClientButton;
    private Button removeClientButton;
    private TextBox searchTextBox;
    private Button searchButton;
    private ListBox clientsListBox;

    public ClientForm()
    {
        this.Text = "Управление клиентами";
        this.Width = 500;
        this.Height = 400;

        nameTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 10),
            Width = 150,
            PlaceholderText = "Имя"
        };

        emailTextBox = new TextBox
        {
            Location = new System.Drawing.Point(170, 10),
            Width = 150,
            PlaceholderText = "Email"
        };
    }
}

```

```
phoneTextBox = new TextBox
{
    Location = new System.Drawing.Point(330, 10),
    Width = 100,
    PlaceholderText = "Телефон"
};

addressTextBox = new TextBox
{
    Location = new System.Drawing.Point(10, 40),
    Width = 450,
    Multiline = true,
    PlaceholderText = "Адрес"
};

addClientButton = new Button
{
    Location = new System.Drawing.Point(10, 80),
    Text = "Добавить",
    Width = 100
};
addClientButton.Click += AddClientButton_Click;

removeClientButton = new Button
{
    Location = new System.Drawing.Point(120, 80),
    Text = "Удалить",
    Width = 100
};
removeClientButton.Click += RemoveClientButton_Click;

searchTextBox = new TextBox
{
    Location = new System.Drawing.Point(10, 110),
    Width = 200,
    PlaceholderText = "Поиск"
};

searchButton = new Button
{
    Location = new System.Drawing.Point(220, 110),
    Text = "Искать",
    Width = 80
};
searchButton.Click += SearchButton_Click;

clientsListBox = new ListBox
```

```

{
    Location = new System.Drawing.Point(10, 140),
    Width = 450,
    Height = 200
};

this.Controls.Add(nameTextBox);
this.Controls.Add(emailTextBox);
this.Controls.Add(phoneTextBox);
this.Controls.Add(addressTextBox);
this.Controls.Add(addClientButton);
this.Controls.Add(removeClientButton);
this.Controls.Add(searchTextBox);
this.Controls.Add(searchButton);
this.Controls.Add(clientsListBox);

clientManager = new ClientManager();
UpdateClientsList();
}

private void UpdateClientsList()
{
    clientsListBox.Items.Clear();
    foreach (var client in clientManager.Clients)
    {
        clientsListBox.Items.Add($"{client.Name} - {client.Email} ({client.Phone})");
    }
}

private void AddClientButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(nameTextBox.Text) || string.IsNullOrEmpty(emailTextBox.Text)
    || string.IsNullOrEmpty(phoneTextBox.Text) || string.IsNullOrEmpty(addressTextBox.Text))
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    Client newClient = new Client(nameTextBox.Text, emailTextBox.Text,
    phoneTextBox.Text, addressTextBox.Text);
    try
    {
        clientManager.AddClient(newClient);
        nameTextBox.Clear();
        emailTextBox.Clear();
        phoneTextBox.Clear();
        addressTextBox.Clear();
        UpdateClientsList();
    }
}

```

```

        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void RemoveClientButton_Click(object sender, EventArgs e)
    {
        if (clientsListBox.SelectedIndex == -1)
        {
            MessageBox.Show("Выберите клиента для удаления!");
            return;
        }
        string selectedItem = clientsListBox.SelectedItem.ToString();
        string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
        if (parts.Length >= 2)
        {
            string name = parts[0].Trim();
            string email = parts[1].Trim();
            var clientToRemove = clientManager.Clients.Find(c => c.Name == name && c.Email
== email);
            if (clientToRemove != null)
            {
                try
                {
                    clientManager.RemoveClient(clientToRemove);
                    UpdateClientsList();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }
    }

    private void SearchButton_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(searchTextBox.Text))
        {
            UpdateClientsList();
            return;
        }
        var searchResults = clientManager.SearchClients(searchTextBox.Text);
        clientsListBox.Items.Clear();
        foreach (var client in searchResults)
        {
            clientsListBox.Items.Add($"{client.Name} - {client.Email} ({client.Phone})");
        }
    }

```



```

    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new ClientForm());
}
}
...

```

Объяснение:

1. Класс `Client`:

- Хранит имя, email, телефон и адрес клиента.

2. Класс `ClientManager`:

- Управляет списком клиентов.
- Реализует методы для добавления, удаления и поиска клиентов.
- Сохраняет и загружает клиентов из файла.

3. Графический интерфейс:

- Поля для ввода имени, email, телефона и адреса.
- Кнопки для добавления и удаления клиентов.
- Поле для поиска и кнопка для выполнения поиска.
- Список клиентов.

4. Проверка граничных значений:

- Пустые поля при добавлении клиента.
- Некорректный выбор клиента для удаления.

---

## ### 16. Управление доставкой

Описание: Создать программу для управления доставкой. Пользователь может добавлять, удалять, редактировать доставки и отслеживать их статус. Программа должна сохранять данные в файл.

Код программы:

```

```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;

```

```

using System.Linq;

public enum DeliveryStatus
{
    Новый,
    В_пути,
    Доставлен
}

public class Delivery
{
    public string CustomerName { get; set; }
    public string Address { get; set; }
    public DateTime DeliveryDate { get; set; }
    public DeliveryStatus Status { get; set; }

    public Delivery(string customerName, string address, DateTime deliveryDate)
    {
        CustomerName = customerName;
        Address = address;
        DeliveryDate = deliveryDate;
        Status = DeliveryStatus.Новый;
    }

    public void UpdateStatus(DeliveryStatus newStatus)
    {
        Status = newStatus;
    }
}

public class DeliveryManager
{
    public List<Delivery> Deliveries { get; private set; }

    public DeliveryManager()
    {
        Deliveries = new List<Delivery>();
        LoadDeliveries();
    }

    public void AddDelivery(Delivery delivery)
    {
        if (delivery == null)
        {
            throw new ArgumentNullException(nameof(delivery));
        }
        Deliveries.Add(delivery);
        SaveDeliveries();
    }
}

```

```

    }

    public void RemoveDelivery(Delivery delivery)
    {
        if (delivery == null)
        {
            throw new ArgumentNullException(nameof(delivery));
        }
        Deliveries.Remove(delivery);
        SaveDeliveries();
    }

    public void UpdateDeliveryStatus(Delivery delivery, DeliveryStatus newStatus)
    {
        if (delivery == null)
        {
            throw new ArgumentNullException(nameof(delivery));
        }
        delivery.UpdateStatus(newStatus);
        SaveDeliveries();
    }

    private void SaveDeliveries()
    {
        File.WriteAllLines("deliveries.txt", Deliveries.Select(d =>
        $"{d.CustomerName}|{d.Address}|{d.DeliveryDate.ToString("yyyy-MM-dd")}|{(int)d.Status}"));
    }

    private void LoadDeliveries()
    {
        if (File.Exists("deliveries.txt"))
        {
            var lines = File.ReadAllLines("deliveries.txt");
            foreach (var line in lines)
            {
                var parts = line.Split('|');
                if (parts.Length == 4)
                {
                    DateTime deliveryDate;
                    DeliveryStatus status;
                    if (DateTime.TryParse(parts[2], out deliveryDate) &&
                        Enum.TryParse(typeof(DeliveryStatus), parts[3], out status))
                    {
                        Deliveries.Add(new Delivery(parts[0], parts[1], deliveryDate));
                        Deliveries.Last().Status = status;
                    }
                }
            }
        }
    }

```

```

    }
}
}
}

```

```

public class DeliveryForm : Form
{
    private DeliveryManager deliveryManager;
    private TextBox customerNameTextBox;
    private TextBox addressTextBox;
    private DateTimePicker deliveryDatePicker;
    private ComboBox statusComboBox;
    private Button addDeliveryButton;
    private Button removeDeliveryButton;
    private Button updateStatusButton;
    private ListBox deliveriesListBox;

    public DeliveryForm()
    {
        this.Text = "Управление доставкой";
        this.Width = 600;
        this.Height = 500;

        customerNameTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 10),
            Width = 150,
            PlaceholderText = "Имя клиента"
        };

        addressTextBox = new TextBox
        {
            Location = new System.Drawing.Point(170, 10),
            Width = 200,
            PlaceholderText = "Адрес"
        };

        deliveryDatePicker = new DateTimePicker
        {
            Location = new System.Drawing.Point(380, 10)
        };

        statusComboBox = new ComboBox
        {
            Location = new System.Drawing.Point(10, 40),
            Width = 100,
            Items = { "Новый", "В пути", "Доставлен" }
        };
    }
}

```

```

addDeliveryButton = new Button
{
    Location = new System.Drawing.Point(10, 70),
    Text = "Добавить",
    Width = 100
};
addDeliveryButton.Click += AddDeliveryButton_Click;

removeDeliveryButton = new Button
{
    Location = new System.Drawing.Point(120, 70),
    Text = "Удалить",
    Width = 100
};
removeDeliveryButton.Click += RemoveDeliveryButton_Click;

updateStatusButton = new Button
{
    Location = new System.Drawing.Point(220, 70),
    Text = "Обновить статус",
    Width = 120
};
updateStatusButton.Click += UpdateStatusButton_Click;

deliveriesListBox = new ListBox
{
    Location = new System.Drawing.Point(10, 100),
    Width = 560,
    Height = 250
};

this.Controls.Add(customerNameTextBox);
this.Controls.Add(addressTextBox);
this.Controls.Add(deliveryDatePicker);
this.Controls.Add(statusComboBox);
this.Controls.Add(addDeliveryButton);
this.Controls.Add(removeDeliveryButton);
this.Controls.Add(updateStatusButton);
this.Controls.Add(deliveriesListBox);

deliveryManager = new DeliveryManager();
UpdateDeliveriesList();
}

private void UpdateDeliveriesList()
{
    deliveriesListBox.Items.Clear();

```

```

        foreach (var delivery in deliveryManager.Deliveries)
        {
            deliveriesListBox.Items.Add($"{delivery.CustomerName} - {delivery.Address}
({delivery.Status})");
        }
    }

    private void AddDeliveryButton_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(customerNameTextBox.Text) ||
            string.IsNullOrEmpty(addressTextBox.Text))
        {
            MessageBox.Show("Заполните все поля!");
            return;
        }
        DateTime deliveryDate = deliveryDatePicker.Value;
        Delivery newDelivery = new Delivery(customerNameTextBox.Text,
            addressTextBox.Text, deliveryDate);
        try
        {
            deliveryManager.AddDelivery(newDelivery);
            customerNameTextBox.Clear();
            addressTextBox.Clear();
            UpdateDeliveriesList();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void RemoveDeliveryButton_Click(object sender, EventArgs e)
    {
        if (deliveriesListBox.SelectedIndex == -1)
        {
            MessageBox.Show("Выберите доставку для удаления!");
            return;
        }
        string selectedItem = deliveriesListBox.SelectedItem.ToString();
        string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
        if (parts.Length >= 2)
        {
            string customerName = parts[0].Trim();
            string address = parts[1].Trim();
            var deliveryToRemove = deliveryManager.Deliveries.Find(d => d.CustomerName ==
customerName && d.Address == address);
            if (deliveryToRemove != null)
            {

```

```

        try
        {
            deliveryManager.RemoveDelivery(deliveryToRemove);
            UpdateDeliveriesList();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}

```

```

private void UpdateStatusButton_Click(object sender, EventArgs e)
{
    if (deliveriesListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите доставку для обновления статуса!");
        return;
    }
    string selectedItem = deliveriesListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string customerName = parts[0].Trim();
        string address = parts[1].Trim();
        var deliveryToUpdate = deliveryManager.Deliveries.Find(d => d.CustomerName ==
customerName && d.Address == address);
        if (deliveryToUpdate != null)
        {
            DeliveryStatus newStatus = (DeliveryStatus)Enum.Parse(typeof(DeliveryStatus),
statusComboBox.SelectedItem.ToString());
            try
            {
                deliveryManager.UpdateDeliveryStatus(deliveryToUpdate, newStatus);
                UpdateDeliveriesList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

```

```

[STAThread]
static void Main()
{

```

```

        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new DeliveryForm());
    }
}
...

```

Объяснение:

1. Класс `Delivery`:

- Хранит имя клиента, адрес, дату доставки и статус.

2. Класс `DeliveryManager`:

- Управляет списком доставок.
- Реализует методы для добавления, удаления и обновления статуса доставок.
- Сохраняет и загружает доставки из файла.

3. Графический интерфейс:

- Поля для ввода имени клиента, адреса и выбора даты доставки.
- Комбо- для выбора статуса.
- Кнопки для добавления, удаления и обновления статуса доставок.
- Список доставок.

4. Проверка граничных значений:

- Пустые поля при добавлении доставки.
- Некорректный выбор доставки для удаления или обновления статуса.

---

## ### 17. Управление отчётами

Описание: Создать программу для управления отчётами. Пользователь может генерировать, просматривать, редактировать и сохранять отчёты. Программа должна сохранять данные в файл.

Код программы:

```

```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public class Report
{
    public string Title { get; set; }
    public string Content { get; set; }
    public DateTime CreationDate { get; set; }
}

```



```

public Report(string title, string content, DateTime creationDate)
{
    Title = title;
    Content = content;
    CreationDate = creationDate;
}
}

public class ReportManager
{
    public List<Report> Reports { get; private set; }

    public ReportManager()
    {
        Reports = new List<Report>();
        LoadReports();
    }

    public void AddReport(Report report)
    {
        if (report == null)
        {
            throw new ArgumentNullException(nameof(report));
        }
        Reports.Add(report);
        SaveReports();
    }

    public void RemoveReport(Report report)
    {
        if (report == null)
        {
            throw new ArgumentNullException(nameof(report));
        }
        Reports.Remove(report);
        SaveReports();
    }

    public void UpdateReport(Report report, string newTitle, string newContent)
    {
        if (report == null)
        {
            throw new ArgumentNullException(nameof(report));
        }
        report.Title = newTitle;
        report.Content = newContent;
        SaveReports();
    }
}

```

```

    }

    private void SaveReports()
    {
        File.WriteAllLines("reports.txt", Reports.Select(r =>
        $"{r.Title}|{r.Content}|{r.CreationDate.ToString("yyyy-MM-dd HH:mm:ss")}"));
    }

    private void LoadReports()
    {
        if (File.Exists("reports.txt"))
        {
            var lines = File.ReadAllLines("reports.txt");
            foreach (var line in lines)
            {
                var parts = line.Split('|');
                if (parts.Length == 3)
                {
                    DateTime creationDate;
                    if (DateTime.TryParse(parts[2], out creationDate))
                    {
                        Reports.Add(new Report(parts[0], parts[1], creationDate));
                    }
                }
            }
        }
    }
}

```

```

public class ReportForm : Form
{
    private ReportManager reportManager;
    private TextBox titleTextBox;
    private TextBox contentTextBox;
    private Button addReportButton;
    private Button removeReportButton;
    private Button updateReportButton;
    private ListBox reportsListBox;

    public ReportForm()
    {
        this.Text = "Управление отчётами";
        this.Width = 600;
        this.Height = 500;

        titleTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 10),

```

```

        Width = 200,
        PlaceholderText = "Название"
    };

    contentTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 40),
        Width = 200,
        Height = 100,
        Multiline = true,
        ScrollBars = ScrollBars.Both,
        PlaceholderText = "Содержание"
    };

    addReportButton = new Button
    {
        Location = new System.Drawing.Point(10, 150),
        Text = "Добавить",
        Width = 100
    };
    addReportButton.Click += AddReportButton_Click;

    removeReportButton = new Button
    {
        Location = new System.Drawing.Point(120, 150),
        Text = "Удалить",
        Width = 100
    };
    removeReportButton.Click += RemoveReportButton_Click;

    updateReportButton = new Button
    {
        Location = new System.Drawing.Point(220, 150),
        Text = "Обновить",
        Width = 100
    };
    updateReportButton.Click += UpdateReportButton_Click;

    reportsListBox = new ListBox
    {
        Location = new System.Drawing.Point(10, 180),
        Width = 560,
        Height = 200
    };

    this.Controls.Add(titleTextBox);
    this.Controls.Add(contentTextBox);
    this.Controls.Add(addReportButton);

```

```

        this.Controls.Add(removeReportButton);
        this.Controls.Add(updateReportButton);
        this.Controls.Add(reportsListBox);

        reportManager = new ReportManager();
        UpdateReportsList();
    }

    private void UpdateReportsList()
    {
        reportsListBox.Items.Clear();
        foreach (var report in reportManager.Reports)
        {
            reportsListBox.Items.Add($"{report.Title}
({report.CreationDate.ToString("yyyy-MM-dd")}");
        }
    }

    private void AddReportButton_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(titleTextBox.Text) || string.IsNullOrEmpty(contentTextBox.Text))
        {
            MessageBox.Show("Заполните все поля!");
            return;
        }
        Report newReport = new Report(titleTextBox.Text, contentTextBox.Text,
DateTime.Now);
        try
        {
            reportManager.AddReport(newReport);
            titleTextBox.Clear();
            contentTextBox.Clear();
            UpdateReportsList();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void RemoveReportButton_Click(object sender, EventArgs e)
    {
        if (reportsListBox.SelectedIndex == -1)
        {
            MessageBox.Show("Выберите отчёт для удаления!");
            return;
        }
        string selectedItem = reportsListBox.SelectedItem.ToString();
    }

```

```

string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
if (parts.Length >= 2)
{
    string title = parts[0].Trim();
    DateTime date;
    if (DateTime.TryParse(parts[1].Split(' ')[0], out date))
    {
        var reportToRemove = reportManager.Reports.Find(r => r.Title == title &&
r.CreationDate.Date == date.Date);
        if (reportToRemove != null)
        {
            try
            {
                reportManager.RemoveReport(reportToRemove);
                UpdateReportsList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}
}

```

```

private void UpdateReportButton_Click(object sender, EventArgs e)
{
    if (reportsListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите отчёт для обновления!");
        return;
    }
    string selectedItem = reportsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string title = parts[0].Trim();
        DateTime date;
        if (DateTime.TryParse(parts[1].Split(' ')[0], out date))
        {
            var reportToUpdate = reportManager.Reports.Find(r => r.Title == title &&
r.CreationDate.Date == date.Date);
            if (reportToUpdate != null)
            {
                if (string.IsNullOrEmpty(titleTextBox.Text) ||
string.IsNullOrEmpty(contentTextBox.Text))
                {
                    MessageBox.Show("Заполните все поля!");
                }
            }
        }
    }
}

```

```

        return;
    }
    try
    {
        reportManager.UpdateReport(reportToUpdate, titleTextBox.Text,
contentTextBox.Text);
        UpdateReportsList();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
}
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new ReportForm());
}
}
...

```

Объяснение:

1. Класс `Report`:
  - Хранит название, содержание и дату создания отчёта.
2. Класс `ReportManager`:
  - Управляет списком отчётов.
  - Реализует методы для добавления, удаления и обновления отчётов.
  - Сохраняет и загружает отчёты из файла.
3. Графический интерфейс:
  - Поля для ввода названия и содержания отчёта.
  - Кнопки для добавления, удаления и обновления отчётов.
  - Список отчётов.
4. Проверка граничных значений:
  - Пустые поля при добавлении или обновлении отчёта.
  - Некорректный выбор отчёта для удаления или обновления.

---

### ### 18. Управление резервированием

Описание: Создать программу для управления резервированием. Пользователь может добавлять, удалять, редактировать резервы и отслеживать их статус. Программа должна сохранять данные в файл.

Код программы:

```
```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public enum ReservationStatus
{
    Активно,
    Отменено,
    Завершено
}

public class Reservation
{
    public string CustomerName { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public ReservationStatus Status { get; set; }

    public Reservation(string customerName, DateTime startTime, DateTime endTime)
    {
        CustomerName = customerName;
        StartTime = startTime;
        EndTime = endTime;
        Status = ReservationStatus.Активно;
    }

    public void UpdateStatus(ReservationStatus newStatus)
    {
        Status = newStatus;
    }
}

public class ReservationManager
{
    public List<Reservation> Reservations { get; private set; }

    public ReservationManager()
```

```

{
    Reservations = new List<Reservation>();
    LoadReservations();
}

public void AddReservation(Reservation reservation)
{
    if (reservation == null)
    {
        throw new ArgumentNullException(nameof(reservation));
    }
    Reservations.Add(reservation);
    SaveReservations();
}

public void RemoveReservation(Reservation reservation)
{
    if (reservation == null)
    {
        throw new ArgumentNullException(nameof(reservation));
    }
    Reservations.Remove(reservation);
    SaveReservations();
}

public void UpdateReservationStatus(Reservation reservation, ReservationStatus
newStatus)
{
    if (reservation == null)
    {
        throw new ArgumentNullException(nameof(reservation));
    }
    reservation.UpdateStatus(newStatus);
    SaveReservations();
}

private void SaveReservations()
{
    File.WriteAllLines("reservations.txt", Reservations.Select(r =>
    $"{r.CustomerName}|{r.StartTime.ToString("yyyy-MM-dd
HH:mm")}|{r.EndTime.ToString("yyyy-MM-dd HH:mm")}|{(int)r.Status}"));
}

private void LoadReservations()
{
    if (File.Exists("reservations.txt"))
    {
        var lines = File.ReadAllLines("reservations.txt");
    }
}

```



```

foreach (var line in lines)
{
    var parts = line.Split('|');
    if (parts.Length == 4)
    {
        DateTime startTime;
        DateTime endTime;
        ReservationStatus status;
        if (DateTime.TryParse(parts[1], out startTime) && DateTime.TryParse(parts[2],
out endTime) && Enum.TryParse(typeof(ReservationStatus), parts[3], out status))
        {
            Reservations.Add(new Reservation(parts[0], startTime, endTime));
            Reservations.Last().Status = status;
        }
    }
}
}
}
}
}

```

```

public class ReservationForm : Form
{
    private ReservationManager reservationManager;
    private TextBox customerNameTextBox;
    private DateTimePicker startTimePicker;
    private DateTimePicker endTimePicker;
    private ComboBox statusComboBox;
    private Button addReservationButton;
    private Button removeReservationButton;
    private Button updateStatusButton;
    private ListBox reservationsListBox;

    public ReservationForm()
    {
        this.Text = "Управление резервированием";
        this.Width = 600;
        this.Height = 500;

        customerNameTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 10),
            Width = 150,
            PlaceholderText = "Имя клиента"
        };

        startTimePicker = new DateTimePicker
        {
            Location = new System.Drawing.Point(170, 10),

```

```

        Width = 150
    };

    endTimePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(330, 10),
        Width = 150
    };

    statusComboBox = new ComboBox
    {
        Location = new System.Drawing.Point(10, 40),
        Width = 100,
        Items = { "Активно", "Отменено", "Завершено" }
    };

    addReservationButton = new Button
    {
        Location = new System.Drawing.Point(10, 70),
        Text = "Добавить",
        Width = 100
    };
    addReservationButton.Click += AddReservationButton_Click;

    removeReservationButton = new Button
    {
        Location = new System.Drawing.Point(120, 70),
        Text = "Удалить",
        Width = 100
    };
    removeReservationButton.Click += RemoveReservationButton_Click;

    updateStatusButton = new Button
    {
        Location = new System.Drawing.Point(220, 70),
        Text = "Обновить статус",
        Width = 120
    };
    updateStatusButton.Click += UpdateStatusButton_Click;

    reservationsListBox = new ListBox
    {
        Location = new System.Drawing.Point(10, 100),
        Width = 560,
        Height = 250
    };

    this.Controls.Add(customerNameTextBox);

```

```

        this.Controls.Add(startTimePicker);
        this.Controls.Add(endTimePicker);
        this.Controls.Add(statusComboBox);
        this.Controls.Add(addReservationButton);
        this.Controls.Add(removeReservationButton);
        this.Controls.Add(updateStatusButton);
        this.Controls.Add(reservationsListBox);

        reservationManager = new ReservationManager();
        UpdateReservationsList();
    }

    private void UpdateReservationsList()
    {
        reservationsListBox.Items.Clear();
        foreach (var reservation in reservationManager.Reservations)
        {
            reservationsListBox.Items.Add($"{reservation.CustomerName} -
{reservation.StartTime.ToString("yyyy-MM-dd HH:mm")} - {reservation.Status}");
        }
    }

    private void AddReservationButton_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(customerNameTextBox.Text))
        {
            MessageBox.Show("Введите имя клиента!");
            return;
        }
        DateTime startTime = startTimePicker.Value;
        DateTime endTime = endTimePicker.Value;
        if (startTime >= endTime)
        {
            MessageBox.Show("Время начала должно быть раньше времени окончания!");
            return;
        }
        Reservation newReservation = new Reservation(customerNameTextBox.Text,
startTime, endTime);
        try
        {
            reservationManager.AddReservation(newReservation);
            customerNameTextBox.Clear();
            UpdateReservationsList();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

```

```

}

private void RemoveReservationButton_Click(object sender, EventArgs e)
{
    if (reservationsListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите резервирование для удаления!");
        return;
    }
    string selectedItem = reservationsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 3)
    {
        string customerName = parts[0].Trim();
        DateTime startTime;
        if (DateTime.TryParse(parts[1].Split(' ')[0], out startTime))
        {
            var reservationToRemove = reservationManager.Reservations.Find(r =>
r.CustomerName == customerName && r.StartTime == startTime);
            if (reservationToRemove != null)
            {
                try
                {
                    reservationManager.RemoveReservation(reservationToRemove);
                    UpdateReservationsList();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }
    }
}

```

```

private void UpdateStatusButton_Click(object sender, EventArgs e)
{
    if (reservationsListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите резервирование для обновления статуса!");
        return;
    }
    string selectedItem = reservationsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 3)
    {
        string customerName = parts[0].Trim();
        DateTime startTime;
    }
}

```

```

        if (DateTime.TryParse(parts[1].Split(' ')[0], out startTime))
        {
            var reservationToUpdate = reservationManager.Reservations.Find(r =>
r.CustomerName == customerName && r.StartTime == startTime);
            if (reservationToUpdate != null)
            {
                ReservationStatus newStatus =
(ReservationStatus)Enum.Parse(typeof(ReservationStatus),
statusComboBox.SelectedItem.ToString());
                try
                {
                    reservationManager.UpdateReservationStatus(reservationToUpdate,
newStatus);
                    UpdateReservationsList();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new ReservationForm());
}
}
...

```

Объяснение:

1. Класс `Reservation`:
  - Хранит имя клиента, время начала и окончания, а также статус резервирования.
2. Класс `ReservationManager`:
  - Управляет списком резервирований.
  - Реализует методы для добавления, удаления и обновления статуса резервирований.
  - Сохраняет и загружает резервирования из файла.
3. Графический интерфейс:
  - Поля для ввода имени клиента и выбора времени начала и окончания.
  - Комбо- для выбора статуса.
  - Кнопки для добавления, удаления и обновления статуса резервирований.

- Список резервирований.

#### 4. Проверка граничных значений:

- Пустое имя клиента при добавлении резервирования.
- Некорректные временные значения (начало после окончания).
- Некорректный выбор резервирования для удаления или обновления статуса.

---

## ### 19. Управление бюджетом

Описание: Создать программу для управления бюджетом. Пользователь может добавлять, удалять, редактировать доходы и расходы, а также отслеживать общий бюджет. Программа должна сохранять данные в файл.

Код программы:

```
```csharp
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Linq;

public enum TransactionType
{
    Доход,
    Расход
}

public class Transaction
{
    public string Description { get; set; }
    public decimal Amount { get; set; }
    public TransactionType Type { get; set; }
    public DateTime Date { get; set; }

    public Transaction(string description, decimal amount, TransactionType type, DateTime
date)
    {
        Description = description;
        Amount = amount;
        Type = type;
        Date = date;
    }
}
```

```

public class BudgetManager
{
    public List<Transaction> Transactions { get; private set; }

    public decimal TotalBudget
    {
        get
        {
            return Transactions.Sum(t => t.Type == TransactionType.Доход ? t.Amount :
-t.Amount);
        }
    }

    public BudgetManager()
    {
        Transactions = new List<Transaction>();
        LoadTransactions();
    }

    public void AddTransaction(Transaction transaction)
    {
        if (transaction == null)
        {
            throw new ArgumentNullException(nameof(transaction));
        }
        Transactions.Add(transaction);
        SaveTransactions();
    }

    public void RemoveTransaction(Transaction transaction)
    {
        if (transaction == null)
        {
            throw new ArgumentNullException(nameof(transaction));
        }
        Transactions.Remove(transaction);
        SaveTransactions();
    }

    public void UpdateTransaction(Transaction transaction, string newDescription, decimal
newAmount, TransactionType newType)
    {
        if (transaction == null)
        {
            throw new ArgumentNullException(nameof(transaction));
        }
        transaction.Description = newDescription;
        transaction.Amount = newAmount;
    }
}

```

```

        transaction.Type = newType;
        SaveTransactions();
    }

    private void SaveTransactions()
    {
        File.WriteAllLines("transactions.txt", Transactions.Select(t =>
        $"{t.Description}|{t.Amount}|{(int)t.Type}|{t.Date.ToString("yyyy-MM-dd HH:mm:ss")}"));
    }

    private void LoadTransactions()
    {
        if (File.Exists("transactions.txt"))
        {
            var lines = File.ReadAllLines("transactions.txt");
            foreach (var line in lines)
            {
                var parts = line.Split('|');
                if (parts.Length == 4)
                {
                    decimal amount;
                    TransactionType type;
                    DateTime date;
                    if (decimal.TryParse(parts[1], out amount) &&
                        Enum.TryParse(typeof(TransactionType), parts[2], out type) && DateTime.TryParse(parts[3],
                        out date))
                    {
                        Transactions.Add(new Transaction(parts[0], amount, type, date));
                    }
                }
            }
        }
    }
}

```

```

public class BudgetForm : Form
{
    private BudgetManager budgetManager;
    private TextBox descriptionTextBox;
    private TextBox amountTextBox;
    private ComboBox typeComboBox;
    private DateTimePicker datePicker;
    private Button addTransactionButton;
    private Button removeTransactionButton;
    private Button updateTransactionButton;
    private ListBox transactionsListBox;
    private Label totalBudgetLabel;
}

```



```

public BudgetForm()
{
    this.Text = "Управление бюджетом";
    this.Width = 600;
    this.Height = 500;

    descriptionTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 10),
        Width = 150,
        PlaceholderText = "Описание"
    };

    amountTextBox = new TextBox
    {
        Location = new System.Drawing.Point(170, 10),
        Width = 100,
        PlaceholderText = "Сумма"
    };

    typeComboBox = new ComboBox
    {
        Location = new System.Drawing.Point(280, 10),
        Width = 100,
        Items = { "Доход", "Расход" }
    };

    datePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(390, 10)
    };

    addTransactionButton = new Button
    {
        Location = new System.Drawing.Point(10, 40),
        Text = "Добавить",
        Width = 100
    };
    addTransactionButton.Click += AddTransactionButton_Click;

    removeTransactionButton = new Button
    {
        Location = new System.Drawing.Point(120, 40),
        Text = "Удалить",
        Width = 100
    };
    removeTransactionButton.Click += RemoveTransactionButton_Click;
}

```

```

updateTransactionButton = new Button
{
    Location = new System.Drawing.Point(220, 40),
    Text = "ОБНОВИТЬ",
    Width = 120
};
updateTransactionButton.Click += UpdateTransactionButton_Click;

transactionsListBox = new ListBox
{
    Location = new System.Drawing.Point(10, 70),
    Width = 560,
    Height = 200
};

totalBudgetLabel = new Label
{
    Location = new System.Drawing.Point(10, 280),
    Width = 200,
    Text = "Общий бюджет: "
};

this.Controls.Add(descriptionTextBox);
this.Controls.Add(amountTextBox);
this.Controls.Add(typeComboBox);
this.Controls.Add(datePicker);
this.Controls.Add(addTransactionButton);
this.Controls.Add(removeTransactionButton);
this.Controls.Add(updateTransactionButton);
this.Controls.Add(transactionsListBox);
this.Controls.Add(totalBudgetLabel);

budgetManager = new BudgetManager();
UpdateTransactionsList();
UpdateTotalBudget();
}

private void UpdateTransactionsList()
{
    transactionsListBox.Items.Clear();
    foreach (var transaction in budgetManager.Transactions)
    {
        string type = transaction.Type == TransactionType.Доход ? "Доход" : "Расход";
        transactionsListBox.Items.Add($"{transaction.Description} - {type}
({transaction.Amount} руб.)");
    }
}

```

```

private void UpdateTotalBudget()
{
    totalBudgetLabel.Text = $"Общий бюджет: {budgetManager.TotalBudget} руб.";
}

private void AddTransactionButton_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(descriptionTextBox.Text) ||
        string.IsNullOrEmpty(amountTextBox.Text))
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    decimal amount;
    if (!decimal.TryParse(amountTextBox.Text, out amount) || amount < 0)
    {
        MessageBox.Show("Неверная сумма!");
        return;
    }
    TransactionType type = (TransactionType)Enum.Parse(typeof(TransactionType),
        typeComboBox.SelectedItem.ToString());
    DateTime date = datePicker.Value;
    Transaction newTransaction = new Transaction(descriptionTextBox.Text, amount, type,
        date);
    try
    {
        budgetManager.AddTransaction(newTransaction);
        descriptionTextBox.Clear();
        amountTextBox.Clear();
        UpdateTransactionsList();
        UpdateTotalBudget();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void RemoveTransactionButton_Click(object sender, EventArgs e)
{
    if (transactionsListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите транзакцию для удаления!");
        return;
    }
    string selectedItem = transactionsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)

```

```

    {
        string description = parts[0].Trim();
        string amount = parts[1].Split(' ')[0];
        decimal amountValue;
        if (decimal.TryParse(amount, out amountValue))
        {
            var transactionToRemove = budgetManager.Transactions.Find(t => t.Description
== description && t.Amount == amountValue);
            if (transactionToRemove != null)
            {
                try
                {
                    budgetManager.RemoveTransaction(transactionToRemove);
                    UpdateTransactionsList();
                    UpdateTotalBudget();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }
    }
}

private void UpdateTransactionButton_Click(object sender, EventArgs e)
{
    if (transactionsListBox.SelectedIndex == -1)
    {
        MessageBox.Show("Выберите транзакцию для обновления!");
        return;
    }
    string selectedItem = transactionsListBox.SelectedItem.ToString();
    string[] parts = selectedItem.Split(new[] { '-' }, StringSplitOptions.None);
    if (parts.Length >= 2)
    {
        string description = parts[0].Trim();
        string amount = parts[1].Split(' ')[0];
        decimal amountValue;
        if (decimal.TryParse(amount, out amountValue))
        {
            var transactionToUpdate = budgetManager.Transactions.Find(t => t.Description ==
description && t.Amount == amountValue);
            if (transactionToUpdate != null)
            {
                if (string.IsNullOrEmpty(descriptionTextBox.Text) ||
string.IsNullOrEmpty(amountTextBox.Text))
                {

```

```

        MessageBox.Show("Заполните все поля!");
        return;
    }
    decimal newAmount;
    if (!decimal.TryParse(amountTextBox.Text, out newAmount) || newAmount < 0)
    {
        MessageBox.Show("Неверная сумма!");
        return;
    }
    TransactionType newType =
(TransactionType)Enum.Parse(typeof(TransactionType),
typeComboBox.SelectedItem.ToString());
    try
    {
        budgetManager.UpdateTransaction(transactionToUpdate,
descriptionTextBox.Text, newAmount, newType);
        UpdateTransactionsList();
        UpdateTotalBudget();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
}
}
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new BudgetForm());
}
}
...

```

Объяснение:

1. Класс `Transaction`:

- Хранит описание, сумму, тип (доход/расход) и дату транзакции.

2. Класс `BudgetManager`:

- Управляет списком транзакций.
- Реализует методы для добавления, удаления и обновления транзакций.
- Вычисляет общий бюджет.
- Сохраняет и загружает транзакции из файла.

### 3. Графический интерфейс:

- Поля для ввода описания, суммы, выбора типа и даты.
- Кнопки для добавления, удаления и обновления транзакций.
- Список транзакций.
- Отображение общего бюджета.

### 4. Проверка граничных значений:

- Пустые поля при добавлении или обновлении транзакции.
- Негативная сумма.
- Некорректный выбор транзакции для удаления или обновления.

—

## ### 20. Управление файлами и папками

### #### Описание:

Создать программу для управления файлами и папками на компьютере. Функционал должен включать создание, удаление, переименование файлов и папок, а также сортировку по различным критериям.

### #### Код программы:

```
```csharp
using System;
using System.IO;
using System.Windows.Forms;

public class FileManager
{
    private string currentDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
    private ListView listView;

    public FileManager(ListView listView)
    {
        this.listView = listView;
        LoadFilesAndFolders();
    }

    private void LoadFilesAndFolders()
    {
        listView.Items.Clear();
        var files = Directory.GetFiles(currentDirectory, "*", SearchOption.AllDirectories);
        var directories = Directory.GetDirectories(currentDirectory, "*",
SearchOption.AllDirectories);

        foreach (var file in files)
        {
            var fileInfo = new FileInfo(file);
```

```
        listView.Items.Add(new ListViewItem(
            new[] { fileInfo.Name, fileInfo.LastWriteTime.ToString(), fileInfo.Length.ToString()
        }));
    }
}
```

```
foreach (var dir in directories)
{
    var dirInfo = new DirectoryInfo(dir);
    listView.Items.Add(new ListViewItem(
        new[] { dirInfo.Name, dirInfo.LastWriteTime.ToString(), "Папка" })
        { ForeColor = System.Drawing.Color.Blue });
    }
}
```

```
public void CreateFile()
{
    using (var dialog = new SaveFileDialog())
    {
        dialog.InitialDirectory = currentDirectory;
        dialog.Title = "Создать файл";
        dialog.Filter = "Текстовые файлы (*.txt)|*.txt";
        if (dialog.ShowDialog() == DialogResult.OK)
        {
            File.Create(dialog.FileName).Dispose();
            LoadFilesAndFolders();
        }
    }
}
```

```
public void DeleteFile()
{
    if (listView.SelectedItems.Count == 0)
    {
        MessageBox.Show("Сначала выберите файл или папку для удаления.");
        return;
    }
}
```

```
var selectedItem = listView.SelectedItems[0];
var path = Path.Combine(currentDirectory, selectedItem.Text);
```

```
if (File.Exists(path))
{
    File.Delete(path);
}
else if (Directory.Exists(path))
{
    Directory.Delete(path, true);
}
```

```

        LoadFilesAndFolders();
    }

    public void RenameFile()
    {
        if (listView.SelectedItems.Count == 0)
        {
            MessageBox.Show("Сначала выберите файл или папку для переименования.");
            return;
        }

        var selectedItem = listView.SelectedItems[0];
        var oldPath = Path.Combine(currentDirectory, selectedItem.Text);

        using (var dialog = new SaveFileDialog())
        {
            dialog.InitialDirectory = currentDirectory;
            dialog.Title = "Переименовать";
            dialog.Filter = "*. *|*.*";
            if (dialog.ShowDialog() == DialogResult.OK)
            {
                var newPath = dialog.FileName;
                if (File.Exists(oldPath))
                {
                    File.Move(oldPath, newPath);
                }
                else if (Directory.Exists(oldPath))
                {
                    Directory.Move(oldPath, newPath);
                }
                LoadFilesAndFolders();
            }
        }
    }

    public void SortByDate()
    {
        var files = Directory.GetFiles(currentDirectory, "*", SearchOption.AllDirectories)
            .OrderBy(f => new FileInfo(f).LastWriteTime)
            .ToList();
        var directories = Directory.GetDirectories(currentDirectory, "*",
SearchOption.AllDirectories)
            .OrderBy(d => new DirectoryInfo(d).LastWriteTime)
            .ToList();

        listView.Items.Clear();
    }

```



```

        foreach (var file in files)
        {
            var fileInfo = new FileInfo(file);
            listView.Items.Add(new ListViewItem(
                new[] { fileInfo.Name, fileInfo.LastWriteTime.ToString(), fileInfo.Length.ToString()
            }));
        }

```

```

        foreach (var dir in directories)
        {
            var dirInfo = new DirectoryInfo(dir);
            listView.Items.Add(new ListViewItem(
                new[] { dirInfo.Name, dirInfo.LastWriteTime.ToString(), "Панка" })
                { ForeColor = System.Drawing.Color.Blue });
        }
    }

```

```

    public void SortByName()
    {
        var files = Directory.GetFiles(currentDirectory, "*", SearchOption.AllDirectories)
            .OrderBy(f => Path.GetFileName(f))
            .ToList();
        var directories = Directory.GetDirectories(currentDirectory, "*",
SearchOption.AllDirectories)
            .OrderBy(d => Path.GetFileName(d))
            .ToList();

```

```

        listView.Items.Clear();

```

```

        foreach (var file in files)
        {
            var fileInfo = new FileInfo(file);
            listView.Items.Add(new ListViewItem(
                new[] { fileInfo.Name, fileInfo.LastWriteTime.ToString(), fileInfo.Length.ToString()
            }));
        }

```

```

        foreach (var dir in directories)
        {
            var dirInfo = new DirectoryInfo(dir);
            listView.Items.Add(new ListViewItem(
                new[] { dirInfo.Name, dirInfo.LastWriteTime.ToString(), "Панка" })
                { ForeColor = System.Drawing.Color.Blue });
        }
    }
}

```

```

public class FileManagementForm : Form

```

```

{
    private FileManager fileManager;
    private ListView listView;
    private Button createFileButton;
    private Button deleteButton;
    private Button renameButton;
    private Button sortByDateButton;
    private Button sortByNameButton;

    public FileManagementForm()
    {
        this.Text = "Управление файлами и папками";
        this.Width = 800;
        this.Height = 600;
        CreateControls();
    }

    private void CreateControls()
    {
        listView = new ListView
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(760, 400),
            View = View.Details,
            FullRowSelect = true
        };
        listView.Columns.Add("Имя", 200);
        listView.Columns.Add("Дата изменения", 150);
        listView.Columns.Add("Размер/Тип", 150);

        createFileButton = new Button
        {
            Location = new System.Drawing.Point(10, 420),
            Text = "Создать файл",
            Size = new System.Drawing.Size(100, 25)
        };
        createFileButton.Click += CreateFileButton_Click;

        deleteButton = new Button
        {
            Location = new System.Drawing.Point(120, 420),
            Text = "Удалить",
            Size = new System.Drawing.Size(100, 25)
        };
        deleteButton.Click += DeleteButton_Click;

        renameButton = new Button
        {

```

```

        Location = new System.Drawing.Point(230, 420),
        Text = "Переименовать",
        Size = new System.Drawing.Size(100, 25)
    };
    renameButton.Click += RenameButton_Click;

    sortByDateButton = new Button
    {
        Location = new System.Drawing.Point(340, 420),
        Text = "Сортировать по дате",
        Size = new System.Drawing.Size(120, 25)
    };
    sortByDateButton.Click += SortByDateButton_Click;

    sortByNameButton = new Button
    {
        Location = new System.Drawing.Point(470, 420),
        Text = "Сортировать по имени",
        Size = new System.Drawing.Size(120, 25)
    };
    sortByNameButton.Click += SortByNameButton_Click;

    this.Controls.Add(listView);
    this.Controls.Add(createFileButton);
    this.Controls.Add(deleteButton);
    this.Controls.Add(renameButton);
    this.Controls.Add(sortByDateButton);
    this.Controls.Add(sortByNameButton);

    fileManager = new FileManager(listView);
}

private void CreateFileButton_Click(object sender, EventArgs e)
{
    fileManager.CreateFile();
}

private void DeleteButton_Click(object sender, EventArgs e)
{
    fileManager.DeleteFile();
}

private void RenameButton_Click(object sender, EventArgs e)
{
    fileManager.RenameFile();
}

private void SortByDateButton_Click(object sender, EventArgs e)

```

```

    {
        fileManager.SortByDate();
    }

    private void SortByNameButton_Click(object sender, EventArgs e)
    {
        fileManager.SortByName();
    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new FileManagementForm());
    }
}
...

```

#### Объяснение:

1. Класс `FileManager`:

- Обрабатывает операции с файлами и папками.
- Реализует методы для создания, удаления, переименования файлов и папок.
- Позволяет выполнять сортировку файлов по различным критериям, таким как имя и дата изменения.

2. Класс `FileManagementForm`:

- Предоставляет графический интерфейс для управления файлами и папками.
- Содержит элементы для отображения списка файлов и папок, а также кнопки для выполнения операций.

3. Графический интерфейс:

- Пользователь может просматривать список файлов и папок.
- Имеются кнопки для создания, удаления, переименования файлов и папок.
- Реализована возможность сортировки файлов по различным параметрам.

4. Особенности:

- Работа с файловой системой через потоки.
- Обработка исключений при выполнении операций с файлами и папками.
- Возможность сортировки файлов по имени и дате.

---

## ### 21. Управление системой безопасности данных

#### Описание:

Создать программу для управления безопасностью данных, включая шифрование, дешифрование и проверку целостности данных.

#### Код программы:

```
```csharp
using System;
using System.IO;
using System.Security.Cryptography;
using System.Windows.Forms;

public class DataSecurityManager
{
    private OpenFileDialog openFileDialog;
    private SaveFileDialog saveFileDialog;

    public void EncryptFile()
    {
        using (openFileDialog = new OpenFileDialog())
        {
            openFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
            openFileDialog.Title = "Выберите файл для шифрования";
            openFileDialog.Filter = "Все файлы (*.*)|*.*";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                var filePath = openFileDialog.FileName;
                var password = GetPassword();

                if (File.Exists(filePath))
                {
                    byte[] salt = new byte[32];
                    using (var rng = RandomNumberGenerator.Create())
                    {
                        rng.GetBytes(salt);
                    }

                    using (var aes = Aes.Create())
                    {
                        var key = new Rfc2898DeriveBytes(password, salt, 100000).GetBytes(32);
                        var iv = new Rfc2898DeriveBytes(password, salt, 100000).GetBytes(16);

                        aes.Key = key;
                        aes.IV = iv;

                        using (var encryptor = aes.CreateEncryptor(aes.Key, aes.IV))
                        {
                            using (var outputStream = File.Create(filePath + ".enc"))
```

```

        {
            using (var inputStream = File.OpenRead(filePath))
            {
                inputStream.CopyTo(outputStream);
            }
        }
    }
}

MessageBox.Show("Файл зашифрован.");
}
}
}

public void DecryptFile()
{
    using (openFileDialog = new OpenFileDialog())
    {
        openFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
        openFileDialog.Title = "Выберите зашифрованный файл";
        openFileDialog.Filter = "Зашифрованные файлы (*.enc)|*.enc";

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            var encryptedFilePath = openFileDialog.FileName;
            var password = GetPassword();

            if (File.Exists(encryptedFilePath))
            {
                using (var aes = Aes.Create())
                {
                    var salt = new byte[32];
                    var iv = new byte[16];

                    using (var inputStream = File.OpenRead(encryptedFilePath))
                    {
                        inputStream.Read(salt, 0, salt.Length);
                        inputStream.Read(iv, 0, iv.Length);

                        var key = new Rfc2898DeriveBytes(password, salt, 100000).GetBytes(32);

                        aes.Key = key;
                        aes.IV = iv;

                        using (var decryptor = aes.CreateDecryptor(aes.Key, aes.IV))
                        {

```

```

        using (var outputStream = File.Create(encryptedFilePath + ".dec"))
        {
            using (var stream = new CryptoStream(outputStream, decryptor,
CryptoStreamMode.Write))
            {
                inputStream.CopyTo(stream);
            }
        }
    }
}

MessageBox.Show("Файл расшифрован.");
}
}
}
}

```

```

private string GetPassword()
{
    using (var passwordForm = new PasswordForm())
    {
        if (passwordForm.ShowDialog() == DialogResult.OK)
        {
            return passwordForm.Password;
        }
        return string.Empty;
    }
}
}

```

```

public class PasswordForm : Form
{
    private TextBox passwordTextBox;
    private Button okButton;
    private Button cancelButton;

    public string Password { get; private set; }

    public PasswordForm()
    {
        this.Text = "Введите пароль";
        this.Width = 300;
        this.Height = 100;
        CreateControls();
    }

    private void CreateControls()

```

```

{
    passwordTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 20),
        Size = new System.Drawing.Size(260, 20),
        UseSystemPasswordChar = true
    };

    okButton = new Button
    {
        Location = new System.Drawing.Point(10, 50),
        Text = "OK",
        Size = new System.Drawing.Size(75, 25)
    };
    okButton.Click += (sender, e) =>
    {
        Password = passwordTextBox.Text;
        DialogResult = DialogResult.OK;
        Close();
    };

    cancelButton = new Button
    {
        Location = new System.Drawing.Point(95, 50),
        Text = "Отмена",
        Size = new System.Drawing.Size(75, 25)
    };
    cancelButton.Click += (sender, e) =>
    {
        DialogResult = DialogResult.Cancel;
        Close();
    };

    this.Controls.Add(passwordTextBox);
    this.Controls.Add(okButton);
    this.Controls.Add(cancelButton);
}
}

public class DataSecurityForm : Form
{
    private DataSecurityManager dataSecurityManager;
    private Button encryptButton;
    private Button decryptButton;

    public DataSecurityForm()
    {
        this.Text = "Управление безопасностью данных";
    }
}

```



```

        this.Width = 300;
        this.Height = 150;
        CreateControls();
        dataSecurityManager = new DataSecurityManager();
    }

    private void CreateControls()
    {
        encryptButton = new Button
        {
            Location = new System.Drawing.Point(10, 20),
            Text = "Зашифровать файл",
            Size = new System.Drawing.Size(120, 25)
        };
        encryptButton.Click += (sender, e) => dataSecurityManager.EncryptFile();

        decryptButton = new Button
        {
            Location = new System.Drawing.Point(140, 20),
            Text = "Расшифровать файл",
            Size = new System.Drawing.Size(120, 25)
        };
        decryptButton.Click += (sender, e) => dataSecurityManager.DecryptFile();

        this.Controls.Add(encryptButton);
        this.Controls.Add(decryptButton);
    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new DataSecurityForm());
    }
}
...

```

#### Объяснение:

1. Класс `DataSecurityManager`:

- Обрабатывает шифрование и дешифрование файлов.
- Использует алгоритм AES для шифрования данных.
- Генерирует случайную соль и derives ключ и вектор инициализации из пароля.
- Реализует методы для шифрования и дешифрования файлов.
- Использует диалоговые окна для выбора файлов и ввода пароля.

2. Класс `PasswordForm`:

- Предоставляет форму для ввода пароля.

- Содержит текстовое поле с маскированным вводом и кнопки "ОК" и "Отмена".

### 3. Класс `DataSecurityForm`:

- Предоставляет графический интерфейс для управления безопасностью данных.
- Содержит кнопки для шифрования и дешифрования файлов.
- Обрабатывает события кнопок, вызывая соответствующие методы `DataSecurityManager`.

### 4. Графический интерфейс:

- Пользователь может выбрать файл для шифрования или дешифрования.
- Вводит пароль через форму `PasswordForm`.
- Получает уведомления о завершении операций.

### 5. Особенности:

- Использование алгоритма AES для шифрования.
- Генерация случайной соли для ключа.
- Работа с файлами через потоки.
- Проверка корректности пароля.
- Обработка исключений при работе с файлами.

---

## ### 22. Управление здоровьем

### ##### Описание:

Создать программу для отслеживания здоровья, включая физическую активность, питание и сон.

### ##### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Windows.Forms;

public class HealthManager
{
    private Dictionary<string, decimal> activityTracking = new Dictionary<string, decimal>();
    private Dictionary<string, decimal> nutritionTracking = new Dictionary<string, decimal>();
    private Dictionary<string, decimal> sleepTracking = new Dictionary<string, decimal>();

    public void TrackActivity(string activityType, decimal duration)
    {
        if (activityTracking.ContainsKey(activityType))
        {
            activityTracking[activityType] += duration;
        }
        else
    
```

```

    {
        activityTracking.Add(activityType, duration);
    }
    MessageBox.Show($"Активность '{activityType}' отслежена на {duration} минут.");
}

public void TrackNutrition(string foodItem, decimal calories)
{
    if (nutritionTracking.ContainsKey(foodItem))
    {
        nutritionTracking[foodItem] += calories;
    }
    else
    {
        nutritionTracking.Add(foodItem, calories);
    }
    MessageBox.Show($"Пища '{foodItem}' отслежена: {calories} калорий.");
}

public void TrackSleep(string date, decimal hours)
{
    if (sleepTracking.ContainsKey(date))
    {
        sleepTracking[date] = hours;
    }
    else
    {
        sleepTracking.Add(date, hours);
    }
    MessageBox.Show($"Сон на {date} отслежен: {hours} часов.");
}

public void DisplayActivityReport()
{
    var reportForm = new ReportForm();
    reportForm.ActivityTracking = activityTracking;
    reportForm.NutritionTracking = nutritionTracking;
    reportForm.SleepTracking = sleepTracking;
    reportForm.ShowDialog();
}
}

public class ReportForm : Form
{
    private Dictionary<string, decimal> activityTracking;
    private Dictionary<string, decimal> nutritionTracking;
    private Dictionary<string, decimal> sleepTracking;

```

```

public Dictionary<string, decimal> ActivityTracking
{
    set { activityTracking = value; }
}

public Dictionary<string, decimal> NutritionTracking
{
    set { nutritionTracking = value; }
}

public Dictionary<string, decimal> SleepTracking
{
    set { sleepTracking = value; }
}

public ReportForm()
{
    this.Text = "Отчёт по здоровью";
    this.Width = 400;
    this.Height = 300;
    CreateControls();
}

private void CreateControls()
{
    var reportRichTextBox = new RichTextBox
    {
        Location = new System.Drawing.Point(10, 10),
        Size = new System.Drawing.Size(380, 280)
    };

    reportRichTextBox.AppendText("Отчёт по активностям:\n");
    foreach (var activity in activityTracking)
    {
        reportRichTextBox.AppendText($" {activity.Key}: {activity.Value} минут.\n");
    }

    reportRichTextBox.AppendText("\nОтчёт по питанию:\n");
    foreach (var food in nutritionTracking)
    {
        reportRichTextBox.AppendText($" {food.Key}: {food.Value} калорий.\n");
    }

    reportRichTextBox.AppendText("\nОтчёт по сну:\n");
    foreach (var sleep in sleepTracking)
    {
        reportRichTextBox.AppendText($" {sleep.Key}: {sleep.Value} часов.\n");
    }
}

```

```

        this.Controls.Add(reportRichTextBox);
    }
}

public class ActivityForm : Form
{
    private TextBox activityTypeTextBox;
    private TextBox durationTextBox;
    private Label activityTypeLabel;
    private Label durationLabel;

    public string ActivityType { get; private set; }
    public decimal Duration { get; private set; }

    public ActivityForm()
    {
        this.Text = "Добавить активность";
        this.Width = 250;
        this.Height = 150;
        CreateControls();
    }

    private void CreateControls()
    {
        activityTypeLabel = new Label
        {
            Text = "Тип активности:",
            Location = new System.Drawing.Point(10, 10)
        };

        activityTypeTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 30),
            Size = new System.Drawing.Size(200, 20)
        };

        durationLabel = new Label
        {
            Text = "Продолжительность (минут):",
            Location = new System.Drawing.Point(10, 55)
        };

        durationTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 75),
            Size = new System.Drawing.Size(200, 20)
        };
    }
}

```

```

var okButton = new Button
{
    Text = "OK",
    Location = new System.Drawing.Point(10, 100),
    Size = new System.Drawing.Size(80, 25)
};
okButton.Click += (sender, e) =>
{
    if (decimal.TryParse(durationTextBox.Text, out decimal duration))
    {
        ActivityType = activityTypeTextBox.Text;
        Duration = duration;
        DialogResult = DialogResult.OK;
        Close();
    }
    else
    {
        MessageBox.Show("Пожалуйста, введите корректное значение продолжительности.");
    }
};

var cancelButton = new Button
{
    Text = "Отмена",
    Location = new System.Drawing.Point(130, 100),
    Size = new System.Drawing.Size(80, 25)
};
cancelButton.Click += (sender, e) =>
{
    DialogResult = DialogResult.Cancel;
    Close();
};

this.Controls.Add(activityTypeLabel);
this.Controls.Add(activityTypeTextBox);
this.Controls.Add(durationLabel);
this.Controls.Add(durationTextBox);
this.Controls.Add(okButton);
this.Controls.Add(cancelButton);
}
}

public class NutritionForm : Form
{
    private TextBox foodItemTextBox;
    private TextBox caloriesTextBox;

```

```

private Label foodItemLabel;
private Label caloriesLabel;

public string FoodItem { get; private set; }
public decimal Calories { get; private set; }

public NutritionForm()
{
    this.Text = "Добавить питание";
    this.Width = 250;
    this.Height = 150;
    CreateControls();
}

private void CreateControls()
{
    foodItemLabel = new Label
    {
        Text = "Название пищи:",
        Location = new System.Drawing.Point(10, 10)
    };

    foodItemTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 30),
        Size = new System.Drawing.Size(200, 20)
    };

    caloriesLabel = new Label
    {
        Text = "Калорийность:",
        Location = new System.Drawing.Point(10, 55)
    };

    caloriesTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 75),
        Size = new System.Drawing.Size(200, 20)
    };

    var okButton = new Button
    {
        Text = "OK",
        Location = new System.Drawing.Point(10, 100),
        Size = new System.Drawing.Size(80, 25)
    };
    okButton.Click += (sender, e) =>
    {

```

```

        if (decimal.TryParse(caloriesTextBox.Text, out decimal calories))
        {
            FoodItem = foodItemTextBox.Text;
            Calories = calories;
            DialogResult = DialogResult.OK;
            Close();
        }
        else
        {
            MessageBox.Show("Пожалуйста, введите корректное значение калорийности.");
        }
    };

```

```

var cancelButton = new Button
{
    Text = "Отмена",
    Location = new System.Drawing.Point(130, 100),
    Size = new System.Drawing.Size(80, 25)
};
cancelButton.Click += (sender, e) =>
{
    DialogResult = DialogResult.Cancel;
    Close();
};

```

```

        this.Controls.Add(foodItemLabel);
        this.Controls.Add(foodItemTextBox);
        this.Controls.Add(caloriesLabel);
        this.Controls.Add(caloriesTextBox);
        this.Controls.Add(okButton);
        this.Controls.Add(cancelButton);
    }
}

```

```

public class SleepForm : Form
{
    private TextBox dateTextBox;
    private TextBox hoursTextBox;
    private Label dateLabel;
    private Label hoursLabel;

    public string Date { get; private set; }
    public decimal Hours { get; private set; }

    public SleepForm()
    {
        this.Text = "Добавить сон";
    }
}

```



```

        this.Width = 250;
        this.Height = 150;
        CreateControls();
    }

    private void CreateControls()
    {
        dateLabel = new Label
        {
            Text = "Дата:",
            Location = new System.Drawing.Point(10, 10)
        };

        dateTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 30),
            Size = new System.Drawing.Size(200, 20)
        };

        hoursLabel = new Label
        {
            Text = "Количество часов:",
            Location = new System.Drawing.Point(10, 55)
        };

        hoursTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 75),
            Size = new System.Drawing.Size(200, 20)
        };

        var okButton = new Button
        {
            Text = "OK",
            Location = new System.Drawing.Point(10, 100),
            Size = new System.Drawing.Size(80, 25)
        };
        okButton.Click += (sender, e) =>
        {
            if (decimal.TryParse(hoursTextBox.Text, out decimal hours))
            {
                Date = dateTextBox.Text;
                Hours = hours;
                DialogResult = DialogResult.OK;
                Close();
            }
            else
            {

```

```

        MessageBox.Show("Пожалуйста, введите корректное значение часов.");
    }
};

var cancelButton = new Button
{
    Text = "Отмена",
    Location = new System.Drawing.Point(130, 100),
    Size = new System.Drawing.Size(80, 25)
};
cancelButton.Click += (sender, e) =>
{
    DialogResult = DialogResult.Cancel;
    Close();
};

this.Controls.Add(dateLabel);
this.Controls.Add(dateTextBox);
this.Controls.Add(hoursLabel);
this.Controls.Add(hoursTextBox);
this.Controls.Add(okButton);
this.Controls.Add(cancelButton);
}
}

public class HealthForm : Form
{
    private HealthManager healthManager;
    private Button trackActivityButton;
    private Button trackNutritionButton;
    private Button trackSleepButton;
    private Button displayReportButton;

    public HealthForm()
    {
        this.Text = "Управление здоровьем";
        this.Width = 300;
        this.Height = 200;
        CreateControls();
        healthManager = new HealthManager();
    }

    private void CreateControls()
    {
        trackActivityButton = new Button
        {
            Location = new System.Drawing.Point(10, 20),
            Text = "Отслеживать активность",

```

```

        Size = new System.Drawing.Size(120, 25)
    };
    trackActivityButton.Click += (sender, e) =>
    {
        var activityForm = new ActivityForm();
        activityForm.ShowDialog();
        if (activityForm.DialogResult == DialogResult.OK)
        {
            healthManager.TrackActivity(activityForm.ActivityType, activityForm.Duration);
        }
    };

```

```

trackNutritionButton = new Button
{
    Location = new System.Drawing.Point(140, 20),
    Text = "Отслеживать питание",
    Size = new System.Drawing.Size(120, 25)
};
trackNutritionButton.Click += (sender, e) =>
{
    var nutritionForm = new NutritionForm();
    nutritionForm.ShowDialog();
    if (nutritionForm.DialogResult == DialogResult.OK)
    {
        healthManager.TrackNutrition(nutritionForm.FoodItem, nutritionForm.Calories);
    }
};

```

```

trackSleepButton = new Button
{
    Location = new System.Drawing.Point(10, 50),
    Text = "Отслеживать сон",
    Size = new System.Drawing.Size(120, 25)
};
trackSleepButton.Click += (sender, e) =>
{
    var sleepForm = new SleepForm();
    sleepForm.ShowDialog();
    if (sleepForm.DialogResult == DialogResult.OK)
    {
        healthManager.TrackSleep(sleepForm.Date, sleepForm.Hours);
    }
};

```

```

displayReportButton = new Button
{
    Location = new System.Drawing.Point(140, 50),
    Text = "Показать отчёт",

```

```

        Size = new System.Drawing.Size(120, 25)
    };
    displayReportButton.Click += (sender, e) => healthManager.DisplayActivityReport();

    this.Controls.Add(trackActivityButton);
    this.Controls.Add(trackNutritionButton);
    this.Controls.Add(trackSleepButton);
    this.Controls.Add(displayReportButton);
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new HealthForm());
}
}
...

```

#### Объяснение:

1. Класс `HealthManager`:
  - Отвечает за отслеживание физической активности, питания и сна.
  - Реализует методы для добавления данных о активности, питании и сне.
  - Позволяет формировать отчеты о здоровье.
2. Класс `ReportForm`:
  - Отображает отчеты о здоровье.
  - Содержит элементы для визуализации данных о активности, питании и сне.
3. Класс `HealthForm`:
  - Предоставляет графический интерфейс для ввода данных о здоровье.
  - Содержит поля для ввода информации о физической активности, питании и сне.
4. Графический интерфейс:
  - Пользователь может вводить данные о своей активности, питании и сне.
  - Отображаются отчеты о здоровье.
  - Имеются кнопки для добавления данных и просмотра отчетов.
5. Особенности:
  - Хранение данных в словарях или базе данных.
  - Формирование отчетов в удобном для пользователя формате.
  - Обработка пользовательского ввода для предотвращения ошибок.

---

## ### 23. Управление музыкальной коллекцией

### #### Описание:

Создать программу для управления музыкальной коллекцией, включая добавление, удаление, поиск и сортировку треков.

### #### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Windows.Forms;

public class MusicTrack
{
    public string Artist { get; set; }
    public string Title { get; set; }
    public string Genre { get; set; }
    public int Year { get; set; }

    public MusicTrack(string artist, string title, string genre, int year)
    {
        Artist = artist;
        Title = title;
        Genre = genre;
        Year = year;
    }

    public override string ToString()
    {
        return $"{Artist} - {Title} ({Year}) [{Genre}]";
    }
}

public class MusicCollection
{
    private List<MusicTrack> tracks = new List<MusicTrack>();
    private ListView listView;

    public MusicCollection(ListView listView)
    {
        this.listView = listView;
        LoadTracks();
    }

    private void LoadTracks()
    {
        listView.Items.Clear();
        foreach (var track in tracks)

```

```

        {
            listView.Items.Add(new ListViewItem(new[] { track.Artist, track.Title,
track.Genre.ToString(), track.Year.ToString() }));
        }
    }

    public void AddTrack(MusicTrack track)
    {
        tracks.Add(track);
        LoadTracks();
        MessageBox.Show("Трек добавлен.");
    }

    public void RemoveTrack(MusicTrack track)
    {
        if (tracks.Contains(track))
        {
            tracks.Remove(track);
            LoadTracks();
            MessageBox.Show("Трек удалён.");
        }
        else
        {
            MessageBox.Show("Трек не найден в коллекции.");
        }
    }

    public void SearchByArtist(string artist)
    {
        var foundTracks = tracks.Where(t => t.Artist.Contains(artist,
StringComparison.OrdinalIgnoreCase)).ToList();
        if (foundTracks.Any())
        {
            listView.Items.Clear();
            foreach (var track in foundTracks)
            {
                listView.Items.Add(new ListViewItem(new[] { track.Artist, track.Title,
track.Genre.ToString(), track.Year.ToString() }));
            }
            MessageBox.Show("Найденные треки:");
        }
        else
        {
            MessageBox.Show("Треки не найдены.");
        }
    }

    public void SortByYear()

```

```

{
    var sortedTracks = tracks.OrderBy(t => t.Year).ToList();
    listView.Items.Clear();
    foreach (var track in sortedTracks)
    {
        listView.Items.Add(new ListViewItem(new[] { track.Artist, track.Title,
track.Genre.ToString(), track.Year.ToString() }));
    }
    MessageBox.Show("Сортировка по году выполнена.");
}
}

```

```

public class AddTrackForm : Form

```

```

{
    public string Artist { get; set; }
    public string Title { get; set; }
    public string Genre { get; set; }
    public int Year { get; set; }

    public AddTrackForm()
    {
        this.Text = "Добавить трек";
        this.Width = 300;
        this.Height = 200;

        var artistLabel = new Label { Text = "Исполнитель:", Location = new
System.Drawing.Point(10, 10) };
        var titleLabel = new Label { Text = "Название:", Location = new
System.Drawing.Point(10, 40) };
        var genreLabel = new Label { Text = "Жанр:", Location = new
System.Drawing.Point(10, 70) };
        var yearLabel = new Label { Text = "Год:", Location = new System.Drawing.Point(10,
100) };

        var artistTextBox = new TextBox { Location = new System.Drawing.Point(120, 10),
Width = 150 };
        var titleTextBox = new TextBox { Location = new System.Drawing.Point(120, 40), Width
= 150 };
        var genreTextBox = new TextBox { Location = new System.Drawing.Point(120, 70),
Width = 150 };
        var yearTextBox = new TextBox { Location = new System.Drawing.Point(120, 100),
Width = 50 };

        var okButton = new Button { Text = "OK", Location = new System.Drawing.Point(10,
140), Size = new System.Drawing.Size(75, 23) };
        var cancelButton = new Button { Text = "Отмена", Location = new
System.Drawing.Point(95, 140), Size = new System.Drawing.Size(75, 23) };
    }
}

```

```

        okButton.Click += (sender, e) =>
        {
            if (int.TryParse(yearTextBox.Text, out int year))
            {
                Artist = artistTextBox.Text;
                Title = titleTextBox.Text;
                Genre = genreTextBox.Text;
                Year = year;
                DialogResult = DialogResult.OK;
                Close();
            }
            else
            {
                MessageBox.Show("Пожалуйста, введите корректный год.");
            }
        };

        cancelButton.Click += (sender, e) =>
        {
            DialogResult = DialogResult.Cancel;
            Close();
        };

        this.Controls.Add(artistLabel);
        this.Controls.Add(titleLabel);
        this.Controls.Add(genreLabel);
        this.Controls.Add(yearLabel);
        this.Controls.Add(artistTextBox);
        this.Controls.Add(titleTextBox);
        this.Controls.Add(genreTextBox);
        this.Controls.Add(yearTextBox);
        this.Controls.Add(okButton);
        this.Controls.Add(cancelButton);
    }
}

public class SearchArtistForm : Form
{
    public string Artist { get; set; }

    public SearchArtistForm()
    {
        this.Text = "Поиск по исполнителю";
        this.Width = 300;
        this.Height = 100;

        var artistLabel = new Label { Text = "Исполнитель:", Location = new
        System.Drawing.Point(10, 10) };

```



```

        var artistTextBox = new TextBox { Location = new System.Drawing.Point(10, 30), Width
= 260 };
        var okButton = new Button { Text = "OK", Location = new System.Drawing.Point(10,
60), Size = new System.Drawing.Size(75, 23) };
        var cancelButton = new Button { Text = "Отмена", Location = new
System.Drawing.Point(95, 60), Size = new System.Drawing.Size(75, 23) };

        okButton.Click += (sender, e) =>
        {
            Artist = artistTextBox.Text;
            DialogResult = DialogResult.OK;
            Close();
        };

        cancelButton.Click += (sender, e) =>
        {
            DialogResult = DialogResult.Cancel;
            Close();
        };

        this.Controls.Add(artistLabel);
        this.Controls.Add(artistTextBox);
        this.Controls.Add(okButton);
        this.Controls.Add(cancelButton);
    }
}

```

```

public class MusicCollectionForm : Form
{
    private MusicCollection musicCollection;
    private ListView listView;
    private Button addTrackButton;
    private Button removeTrackButton;
    private Button searchByArtistButton;
    private Button sortByYearButton;

    public MusicCollectionForm()
    {
        this.Text = "Управление музыкальной коллекцией";
        this.Width = 500;
        this.Height = 400;
        CreateControls();
        musicCollection = new MusicCollection(listView);
    }

    private void CreateControls()
    {
        listView = new ListView

```

```

{
    Location = new System.Drawing.Point(10, 10),
    Size = new System.Drawing.Size(480, 300),
    View = View.Details,
    FullRowSelect = true
};
listView.Columns.Add("Исполнитель", 150);
listView.Columns.Add("Название", 150);
listView.Columns.Add("Жанр", 100);
listView.Columns.Add("Год", 50);

addTrackButton = new Button
{
    Location = new System.Drawing.Point(10, 320),
    Text = "Добавить трек",
    Size = new System.Drawing.Size(100, 25)
};
addTrackButton.Click += (sender, e) =>
{
    var addTrackForm = new AddTrackForm();
    addTrackForm.ShowDialog();
    if (addTrackForm.DialogResult == DialogResult.OK)
    {
        var track = new MusicTrack(
            addTrackForm.Artist,
            addTrackForm.Title,
            addTrackForm.Genre,
            addTrackForm.Year);
        musicCollection.AddTrack(track);
    }
};

removeTrackButton = new Button
{
    Location = new System.Drawing.Point(120, 320),
    Text = "Удалить трек",
    Size = new System.Drawing.Size(100, 25)
};
removeTrackButton.Click += (sender, e) =>
{
    if (listView.SelectedItems.Count == 0)
    {
        MessageBox.Show("Сначала выберите трек для удаления.");
        return;
    }

    var track = new MusicTrack(
        listView.SelectedItems[0].SubItems[0].Text,

```

```

        listView.SelectedItems[0].SubItems[1].Text,
        listView.SelectedItems[0].SubItems[2].Text,
        int.Parse(listView.SelectedItems[0].SubItems[3].Text));

        musicCollection.RemoveTrack(track);
    };

    searchByArtistButton = new Button
    {
        Location = new System.Drawing.Point(230, 320),
        Text = "Поиск по исполнителю",
        Size = new System.Drawing.Size(120, 25)
    };
    searchByArtistButton.Click += (sender, e) =>
    {
        var searchForm = new SearchArtistForm();
        searchForm.ShowDialog();
        if (searchForm.DialogResult == DialogResult.OK)
        {
            musicCollection.SearchByArtist(searchForm.Artist);
        }
    };

    sortByYearButton = new Button
    {
        Location = new System.Drawing.Point(360, 320),
        Text = "Сортировать по году",
        Size = new System.Drawing.Size(120, 25)
    };
    sortByYearButton.Click += (sender, e) => musicCollection.SortByYear();

    this.Controls.Add(listView);
    this.Controls.Add(addTrackButton);
    this.Controls.Add(removeTrackButton);
    this.Controls.Add(searchByArtistButton);
    this.Controls.Add(sortByYearButton);
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new MusicCollectionForm());
}
...

```

#### Объяснение:

#### 1. Класс `MusicTrack`:

- Представляет музыкальный трек с атрибутами:
  - `Artist` — исполнитель.
  - `Title` — название трека.
  - `Genre` — жанр.
  - `Year` — год выпуска.
- Contains метод `ToString()`, который возвращает строковое представление трека в формате:  
`"{Artist} - {Title} ({Year}) [{Genre}]"`.

#### 2. Класс `MusicCollection`:

- Управляет коллекцией музыкальных треков.
- Основные методы:
  - `LoadTracks()` — загружает треки в `ListView`.
  - `AddTrack(MusicTrack track)` — добавляет новый трек в коллекцию.
  - `RemoveTrack(MusicTrack track)` — удаляет трек из коллекции.
  - `SearchByArtist(string artist)` — ищет треки по исполнителю.
  - `SortByYear()` — сортирует треки по году выпуска.

#### 3. Класс `MusicCollectionForm`:

- Предоставляет графический интерфейс для управления музыкальной коллекцией.
- Содержит:
  - `ListView` для отображения треков.
  - Кнопки для добавления, удаления, поиска и сортировки треков.
- Метод `CreateControls()` настраивает интерфейс и обрабатывает события кнопок.

#### 4. Графический интерфейс:

- Пользователь может просматривать список треков в `ListView`.
- Доступные кнопки:
  - "Добавить трек" — открывает форму `AddTrackForm` для ввода данных нового трека.
  - "Удалить трек" — удаляет выбранный трек из коллекции.
  - "Поиск по исполнителю" — открывает форму `SearchArtistForm` для поиска треков по исполнителю.
  - "Сортировать по году" — сортирует треки по году выпуска.

#### 5. Особенности:

- Работа с `ListView` для отображения данных.
- Реализация поиска по исполнителю с использованием форм.
- Сортировка данных по году выпуска.
- Добавление треков через форму с проверкой корректности ввода года.

#### 6. Формы:

- `AddTrackForm` — форма для добавления нового трека. Содержит поля для ввода имени исполнителя, названия трека, жанра и года выпуска.
- `SearchArtistForm` — форма для поиска треков по исполнителю. Содержит поле для ввода имени исполнителя.

#### Основные функции программы:

- Добавление новых треков.
- Удаление треков.
- Поиск треков по исполнителю.
- Сортировка треков по году выпуска.

## ### 24. Управление путешествиями

#### Описание:

Создать программу для планирования и управления путешествиями, включая маршруты, бронирование отелей и отслеживание расходов.

#### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Windows.Forms;

public class Trip
{
    public string Destination { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public decimal Budget { get; set; }
    public List<Expense> Expenses { get; set; }

    public Trip(string destination, DateTime startDate, DateTime endDate, decimal budget)
    {
        Destination = destination;
        StartDate = startDate;
        EndDate = endDate;
        Budget = budget;
        Expenses = new List<Expense>();
    }

    public void AddExpense(Expense expense)
    {
        Expenses.Add(expense);
        MessageBox.Show($"Расход добавлен: {expense.Amount} на {expense.Description}.");
    }

    public decimal CalculateTotalExpenses()
    {
        return Expenses.Sum(e => e.Amount);
    }
}
```

```

public decimal RemainingBudget()
{
    return Budget - CalculateTotalExpenses();
}

public void DisplayTripDetails()
{
    var reportForm = new TripReportForm();
    reportForm.Destination = Destination;
    reportForm.StartDate = StartDate;
    reportForm.EndDate = EndDate;
    reportForm.Budget = Budget;
    reportForm.Expenses = Expenses;
    reportForm.ShowDialog();
}
}

public class Expense
{
    public string Description { get; set; }
    public decimal Amount { get; set; }

    public Expense(string description, decimal amount)
    {
        Description = description;
        Amount = amount;
    }
}

public class TripForm : Form
{
    private Trip trip;
    private Button addExpenseButton;
    private Button displayDetailsButton;

    public TripForm(Trip trip)
    {
        this.trip = trip;
        this.Text = "Управление путешествием";
        this.Width = 300;
        this.Height = 150;
        CreateControls();
    }

    private void CreateControls()
    {
        addExpenseButton = new Button

```

```

{
    Location = new System.Drawing.Point(10, 20),
    Text = "Добавить расход",
    Size = new System.Drawing.Size(100, 25)
};
addExpenseButton.Click += (sender, e) =>
{
    var addExpenseForm = new AddExpenseForm();
    addExpenseForm.ShowDialog();
    if (addExpenseForm.DialogResult == DialogResult.OK)
    {
        var expense = new Expense(addExpenseForm.Description,
addExpenseForm.Amount);
        trip.AddExpense(expense);
    }
};

displayDetailsButton = new Button
{
    Location = new System.Drawing.Point(120, 20),
    Text = "Показать детали",
    Size = new System.Drawing.Size(100, 25)
};
displayDetailsButton.Click += (sender, e) => trip.DisplayTripDetails();

this.Controls.Add(addExpenseButton);
this.Controls.Add(displayDetailsButton);
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    var trip = new Trip("Париж", DateTime.Now, DateTime.Now.AddDays(7), 50000m);
    Application.Run(new TripForm(trip));
}
}
...

```

#### Объяснение:

1. Класс `Trip`:
  - Представляет путешествие с атрибутами, такими как маршрут, даты, бюджет.
2. Класс `Expense`:
  - Отвечает за расходы, связанные с путешествием.
  - Содержит информацию о сумме, категории и дате расхода.

### 3. Класс `TripForm`:

- Предоставляет интерфейс для планирования и управления путешествиями.
- Содержит поля для ввода деталей путешествия и кнопки для добавления расходов.

### 4. Графический интерфейс:

- Пользователь может вводить данные о путешествии.
- Отображается список расходов и общий бюджет.
- Имеются кнопки для добавления расходов и просмотра деталей путешествия.

### 5. Особенности:

- Хранение данных о путешествии и расходах.
- Расчет общего бюджета и остатка средств.
- Обработка пользовательского ввода для корректного форматирования данных.

---

## ### 25. Управление фотографиями

### #### Описание:

Создать программу для управления фотографиями, включая их организацию в альбомы, добавление описаний и сортировку.

### #### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;

public class Photo
{
    public string Path { get; set; }
    public string Description { get; set; }
    public DateTime DateTaken { get; set; }

    public Photo(string path, string description, DateTime dateTaken)
    {
        Path = path;
        Description = description;
        DateTaken = dateTaken;
    }

    public override string ToString()
    {
        return $"{Path} - {Description} ({DateTaken.ToString("dd.MM.yyyy")}";
    }
}
```



```

public class PhotoAlbum
{
    private List<Photo> photos = new List<Photo>();
    private ListView listView;

    public PhotoAlbum(ListView listView)
    {
        this.listView = listView;
        LoadPhotos();
    }

    private void LoadPhotos()
    {
        listView.Items.Clear();
        foreach (var photo in photos)
        {
            listView.Items.Add(new ListViewItem(new[] { photo.Path, photo.Description,
photo.DateTaken.ToString("dd.MM.yyyy") }));
        }
    }

    public void AddPhoto()
    {
        using (var openFileDialog = new OpenFileDialog())
        {
            openFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);
            openFileDialog.Title = "Выберите фото";
            openFileDialog.Filter = "Изображения (*.jpg;*.png;*.jpeg)|*.jpg;*.png;*.jpeg";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                var photoPath = openFileDialog.FileName;
                var description = GetDescription();
                var dateTaken = DateTime.ParseExact(description, "dd.MM.yyyy", null);

                photos.Add(new Photo(photoPath, description, dateTaken));
                LoadPhotos();
                MessageBox.Show("Фото добавлено.");
            }
        }
    }

    public void RemovePhoto()
    {
        if (listView.SelectedItems.Count == 0)
        {

```

```

        MessageBox.Show("Сначала выберите фото для удаления.");
        return;
    }

    var photoPath = listView.SelectedItems[0].SubItems[0].Text;
    photos.RemoveAll(p => p.Path == photoPath);
    LoadPhotos();
    MessageBox.Show("Фото удалено.");
}

public void SortPhotosByDate()
{
    var sortedPhotos = photos.OrderBy(p => p.DateTaken).ToList();
    photos = new List<Photo>(sortedPhotos);
    LoadPhotos();
    MessageBox.Show("Фото отсортированы по дате.");
}

private string GetDescription()
{
    using (var descriptionForm = new DescriptionForm())
    {
        descriptionForm.ShowDialog();
        return descriptionForm.Description;
    }
}

public class PhotoAlbumForm : Form
{
    private PhotoAlbum photoAlbum;
    private ListView listView;
    private Button addPhotoButton;
    private Button removePhotoButton;
    private Button sortByDateButton;

    public PhotoAlbumForm()
    {
        this.Text = "Управление фотографиями";
        this.Width = 600;
        this.Height = 400;
        CreateControls();
        photoAlbum = new PhotoAlbum(listView);
    }

    private void CreateControls()
    {
        listView = new ListView

```

```

{
    Location = new System.Drawing.Point(10, 10),
    Size = new System.Drawing.Size(580, 350),
    View = View.Details,
    FullRowSelect = true
};
listView.Columns.Add("Путь", 300);
listView.Columns.Add("Описание", 200);
listView.Columns.Add("Дата съёмки", 100);

addPhotoButton = new Button
{
    Location = new System.Drawing.Point(10, 370),
    Text = "Добавить фото",
    Size = new System.Drawing.Size(100, 25)
};
addPhotoButton.Click += (sender, e) => photoAlbum.AddPhoto();

removePhotoButton = new Button
{
    Location = new System.Drawing.Point(120, 370),
    Text = "Удалить фото",
    Size = new System.Drawing.Size(100, 25)
};
removePhotoButton.Click += (sender, e) => photoAlbum.RemovePhoto();

sortByDateButton = new Button
{
    Location = new System.Drawing.Point(230, 370),
    Text = "Отсортировать по дате",
    Size = new System.Drawing.Size(120, 25)
};
sortByDateButton.Click += (sender, e) => photoAlbum.SortPhotosByDate();

this.Controls.Add(listView);
this.Controls.Add(addPhotoButton);
this.Controls.Add(removePhotoButton);
this.Controls.Add(sortByDateButton);
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new PhotoAlbumForm());
}
}

```

...

#### Объяснение:

1. Класс `Photo`:

- Представляет фотографию с атрибутами, такими как название, описание, дата съемки.

2. Класс `PhotoAlbum`:

- Управляет коллекцией фотографий.
- Реализует методы для добавления, удаления и сортировки фотографий.

3. Класс `PhotoAlbumForm`:

- Предоставляет интерфейс для управления фотографиями.
- Содержит список фотографий и кнопки для выполнения операций.

4. Графический интерфейс:

- Пользователь может просматривать список фотографий.
- Имеются кнопки для добавления, удаления и сортировки фотографий.
- Реализован просмотр изображений в формате 预览.

5. Особенности:

- Работа с изображениями и их форматами.
- Сортировка данных по дате и названию.
- Обработка исключений при выполнении операций с фотографиями.

---

## ### 26. Управление рецептами и планированием меню

#### Описание:

Создать программу для управления кулинарными рецептами и планирования меню на неделю.

#### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Windows.Forms;

public class Recipe
{
    public string Name { get; set; }
    public string Description { get; set; }
    public List<string> Ingredients { get; set; }
    public List<string> Instructions { get; set; }
    public int Calories { get; set; }
}
```

```

    public Recipe(string name, string description, List<string> ingredients, List<string>
instructions, int calories)
    {
        Name = name;
        Description = description;
        Ingredients = ingredients;
        Instructions = instructions;
        Calories = calories;
    }

    public override string ToString()
    {
        return Name;
    }
}

public class MealPlan
{
    private Dictionary<DateTime, Recipe> plan = new Dictionary<DateTime, Recipe>();
    private ListView listView;

    public MealPlan(ListView listView)
    {
        this.listView = listView;
        LoadPlan();
    }

    private void LoadPlan()
    {
        listView.Items.Clear();
        foreach (var entry in plan)
        {
            listView.Items.Add(new ListViewItem(new[] { entry.Key.ToString("dd.MM.yyyy"),
entry.Value.Name }));
        }
    }

    public void AddRecipeToPlan()
    {
        var addRecipeForm = new AddRecipeForm();
        addRecipeForm.ShowDialog();
        if (addRecipeForm.DialogResult == DialogResult.OK)
        {
            var recipe = new Recipe(
                addRecipeForm.Name,
                addRecipeForm.Description,
                addRecipeForm.Ingredients,
                addRecipeForm.Instructions,

```

```

        addRecipeForm.Calories);

    if (plan.ContainsKey(addRecipeForm.Date))
    {
        MessageBox.Show("На эту дату рецепт уже добавлен.");
    }
    else
    {
        plan.Add(addRecipeForm.Date, recipe);
        LoadPlan();
        MessageBox.Show("Рецепт добавлен в план.");
    }
}

}

public void RemoveRecipeFromPlan()
{
    if (listView.SelectedItems.Count == 0)
    {
        MessageBox.Show("Сначала выберите рецепт для удаления.");
        return;
    }

    var date = DateTime.Parse(listView.SelectedItems[0].SubItems[0].Text);
    if (plan.ContainsKey(date))
    {
        plan.Remove(date);
        LoadPlan();
        MessageBox.Show("Рецепт удалён.");
    }
    else
    {
        MessageBox.Show("Рецепт не найден.");
    }
}

public void SearchRecipeByName()
{
    var searchForm = new SearchRecipeForm();
    searchForm.ShowDialog();
    if (searchForm.DialogResult == DialogResult.OK)
    {
        var foundRecipe = plan.Values.FirstOrDefault(r =>
r.Name.Contains(searchForm.Name, StringComparison.OrdinalIgnoreCase));
        if (foundRecipe != null)
        {
            var recipeForm = new RecipeForm();
            recipeForm.Recipe = foundRecipe;

```

```

        recipeForm.ShowDialog();
    }
    else
    {
        MessageBox.Show("Рецепт не найден.");
    }
}
}
}

```

```

public class MealPlanForm : Form
{
    private MealPlan mealPlan;
    private ListView listView;
    private Button addRecipeButton;
    private Button removeRecipeButton;
    private Button searchRecipeButton;

    public MealPlanForm()
    {
        this.Text = "Планирование меню";
        this.Width = 400;
        this.Height = 350;
        CreateControls();
        mealPlan = new MealPlan(listView);
    }

    private void CreateControls()
    {
        listView = new ListView
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(380, 280),
            View = View.Details,
            FullRowSelect = true
        };
        listView.Columns.Add("Дата", 100);
        listView.Columns.Add("Название", 270);

        addRecipeButton = new Button
        {
            Location = new System.Drawing.Point(10, 300),
            Text = "Добавить рецепт",
            Size = new System.Drawing.Size(100, 25)
        };
        addRecipeButton.Click += (sender, e) => mealPlan.AddRecipeToPlan();

        removeRecipeButton = new Button
    }
}

```

```

{
    Location = new System.Drawing.Point(120, 300),
    Text = "Удалить рецепт",
    Size = new System.Drawing.Size(100, 25)
};
removeRecipeButton.Click += (sender, e) => mealPlan.RemoveRecipeFromPlan();

searchRecipeButton = new Button
{
    Location = new System.Drawing.Point(230, 300),
    Text = "Поиск рецепта",
    Size = new System.Drawing.Size(100, 25)
};
searchRecipeButton.Click += (sender, e) => mealPlan.SearchRecipeByName();

this.Controls.Add(listView);
this.Controls.Add(addRecipeButton);
this.Controls.Add(removeRecipeButton);
this.Controls.Add(searchRecipeButton);
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new MealPlanForm());
}
...

```

#### Объяснение:

1. Класс `Recipe`:
  - Представляет рецепт с атрибутами, такими как название, ингредиенты, инструкции.
2. Класс `MealPlan`:
  - Управляет планированием меню на неделю.
  - Содержит список рецептов и даты их применения.
3. Класс `MealPlanForm`:
  - Предоставляет интерфейс для управления рецептами и планированием меню.
  - Содержит список рецептов и кнопки для добавления, удаления и поиска.
4. Графический интерфейс:
  - Пользователь может просматривать список рецептов.
  - Имеются кнопки для добавления, удаления и поиска рецептов.
  - Отображается меню на неделю с указанием дней и блюд.



## 5. Особенности:

- Хранение данных о рецептах и меню.
- Возможность формировать и редактировать меню на неделю.
- Обработка пользовательского ввода для корректного форматирования данных.

---

## ### 27. Управление обучением и курсами

### #### Описание:

Создать программу для управления курсами и отслеживания прогресса обучения.

### #### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Windows.Forms;

public class Course
{
    public string Name { get; set; }
    public string Description { get; set; }
    public List<Module> Modules { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }

    public Course(string name, string description, DateTime startTime, DateTime endTime)
    {
        Name = name;
        Description = description;
        Modules = new List<Module>();
        StartTime = startTime;
        EndTime = endTime;
    }

    public void AddModule(Module module)
    {
        Modules.Add(module);
        MessageBox.Show($"Модуль '{module.Name}' добавлен к курсу '{Name}'.");
    }

    public void RemoveModule(Module module)
    {
        if (Modules.Contains(module))
        {
            Modules.Remove(module);
            MessageBox.Show("Модуль удалён.");
        }
    }
}
```

```

    }
    else
    {
        MessageBox.Show("Модуль не найден.");
    }
}

public void DisplayCourseInfo()
{
    var courseInfoForm = new CourseInfoForm();
    courseInfoForm.Course = this;
    courseInfoForm.ShowDialog();
}
}

public class Module
{
    public string Name { get; set; }
    public List<Topic> Topics { get; set; }
    public decimal Progress { get; set; }

    public Module(string name)
    {
        Name = name;
        Topics = new List<Topic>();
        Progress = 0;
    }

    public void AddTopic(Topic topic)
    {
        Topics.Add(topic);
        MessageBox.Show($"Тема '{topic.Name}' добавлена к модулю '{Name}'.");
    }

    public void RemoveTopic(Topic topic)
    {
        if (Topics.Contains(topic))
        {
            Topics.Remove(topic);
            MessageBox.Show("Тема удалена.");
        }
        else
        {
            MessageBox.Show("Тема не найдена.");
        }
    }

    public void UpdateProgress(decimal newProgress)

```

```

{
    if (newProgress >= 0 && newProgress <= 100)
    {
        Progress = newProgress;
        MessageBox.Show($"Прогресс модуля '{Name}' обновлён до {Progress}%.");
    }
    else
    {
        MessageBox.Show("Неверное значение прогресса.");
    }
}

public void DisplayModuleInfo()
{
    var moduleInfoForm = new ModuleInfoForm();
    moduleInfoForm.Module = this;
    moduleInfoForm.ShowDialog();
}
}

public class Topic
{
    public string Name { get; set; }

    public Topic(string name)
    {
        Name = name;
    }
}

public class CourseInfoForm : Form
{
    private Course course;

    public Course Course
    {
        set
        {
            course = value;
            InitializeComponent();
        }
    }

    private void InitializeComponent()
    {
        this.Text = "Информация о курсе";
        this.Width = 400;
        this.Height = 300;
    }
}

```

```

var nameLabel = new Label
{
    Text = $"Название: {course.Name}",
    Location = new System.Drawing.Point(10, 10),
    AutoSize = true
};

var descriptionLabel = new Label
{
    Text = $"Описание: {course.Description}",
    Location = new System.Drawing.Point(10, 30),
    AutoSize = true,
    MaximumSize = new System.Drawing.Size(380, 0)
};

var startTimeLabel = new Label
{
    Text = $"Дата начала: {course.StartTime.ToShortDateString()}",
    Location = new System.Drawing.Point(10, 50),
    AutoSize = true
};

var endTimeLabel = new Label
{
    Text = $"Дата окончания: {course.EndTime.ToShortDateString()}",
    Location = new System.Drawing.Point(10, 70),
    AutoSize = true
};

this.Controls.AddRange(new Control[] { nameLabel, descriptionLabel, startTimeLabel,
endTimeLabel });
}
}

public class ModuleInfoForm : Form
{
    private Module module;

    public Module Module
    {
        set
        {
            module = value;
            InitializeComponent();
        }
    }
}

```

```

private void InitializeComponent()
{
    this.Text = "Информация о модуле";
    this.Width = 400;
    this.Height = 300;

    var nameLabel = new Label
    {
        Text = $"Название: {module.Name}",
        Location = new System.Drawing.Point(10, 10),
        AutoSize = true
    };

    var topicsLabel = new Label
    {
        Text = $"Количество тем: {module.Topics.Count}",
        Location = new System.Drawing.Point(10, 30),
        AutoSize = true
    };

    var progressLabel = new Label
    {
        Text = $"Прогресс: {module.Progress}%",
        Location = new System.Drawing.Point(10, 50),
        AutoSize = true
    };

    this.Controls.AddRange(new Control[] { nameLabel, topicsLabel, progressLabel });
}
}

```

```

public class CreateCourseForm : Form
{
    public string Name { get; private set; }
    public string Description { get; private set; }
    public DateTime StartTime { get; private set; }
    public DateTime EndTime { get; private set; }

    public CreateCourseForm()
    {
        InitializeComponent();
    }

    private void InitializeComponent()
    {
        this.Text = "Создать курс";
        this.Width = 400;
        this.Height = 300;
    }
}

```

```
var nameLabel = new Label
{
    Text = "Название:",
    Location = new System.Drawing.Point(10, 10),
    AutoSize = true
};

var nameTextBox = new TextBox
{
    Location = new System.Drawing.Point(10, 30),
    Size = new System.Drawing.Size(200, 20)
};

var descriptionLabel = new Label
{
    Text = "Описание:",
    Location = new System.Drawing.Point(10, 50),
    AutoSize = true
};

var descriptionTextBox = new TextBox
{
    Location = new System.Drawing.Point(10, 70),
    Size = new System.Drawing.Size(200, 20),
    Multiline = true,
    ScrollBars = ScrollBars.Vertical
};

var startTimeLabel = new Label
{
    Text = "Дата начала:",
    Location = new System.Drawing.Point(10, 100),
    AutoSize = true
};

var startTimePicker = new DateTimePicker
{
    Location = new System.Drawing.Point(10, 120),
    Size = new System.Drawing.Size(200, 20)
};

var endTimeLabel = new Label
{
    Text = "Дата окончания:",
    Location = new System.Drawing.Point(10, 140),
    AutoSize = true
};
```

```

var endTimePicker = new DateTimePicker
{
    Location = new System.Drawing.Point(10, 160),
    Size = new System.Drawing.Size(200, 20)
};

var okButton = new Button
{
    Text = "OK",
    Location = new System.Drawing.Point(10, 190),
    Size = new System.Drawing.Size(75, 23)
};

var cancelButton = new Button
{
    Text = "Отмена",
    Location = new System.Drawing.Point(85, 190),
    Size = new System.Drawing.Size(75, 23)
};

okButton.Click += (sender, e) =>
{
    Name = nameTextBox.Text;
    Description = descriptionTextBox.Text;
    StartTime = startTimePicker.Value;
    EndTime = endTimePicker.Value;
    this.DialogResult = DialogResult.OK;
    this.Close();
};

cancelButton.Click += (sender, e) =>
{
    this.DialogResult = DialogResult.Cancel;
    this.Close();
};

this.Controls.AddRange(new Control[]
{
    nameLabel, nameTextBox,
    descriptionLabel, descriptionTextBox,
    startTimeLabel, startTimePicker,
    endTimeLabel, endTimePicker,
    okButton, cancelButton
});
}
}

```

```

public class AddModuleForm : Form
{
    public string Name { get; private set; }

    public AddModuleForm()
    {
        InitializeComponent();
    }

    private void InitializeComponent()
    {
        this.Text = "Добавить модуль";
        this.Width = 300;
        this.Height = 150;

        var nameLabel = new Label
        {
            Text = "Имя модуля:",
            Location = new System.Drawing.Point(10, 10),
            AutoSize = true
        };

        var nameTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 30),
            Size = new System.Drawing.Size(200, 20)
        };

        var okButton = new Button
        {
            Text = "OK",
            Location = new System.Drawing.Point(10, 60),
            Size = new System.Drawing.Size(75, 23)
        };

        var cancelButton = new Button
        {
            Text = "Отмена",
            Location = new System.Drawing.Point(85, 60),
            Size = new System.Drawing.Size(75, 23)
        };

        okButton.Click += (sender, e) =>
        {
            Name = nameTextBox.Text;
            this.DialogResult = DialogResult.OK;
            this.Close();
        };
    }
}

```



```

cancelButton.Click += (sender, e) =>
{
    this.DialogResult = DialogResult.Cancel;
    this.Close();
};

this.Controls.AddRange(new Control[] { nameLabel, nameTextBox, okButton,
cancelButton });
}
}

```

```

public class RemoveModuleForm : Form
{
    public Module Module { get; private set; }
    private List<Module> modules;

    public RemoveModuleForm(List<Module> modules)
    {
        this.modules = modules;
        InitializeComponent();
    }

    private void InitializeComponent()
    {
        this.Text = "Удалить модуль";
        this.Width = 300;
        this.Height = 200;

        var moduleListBox = new ListBox
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(200, 100)
        };
        moduleListBox.DataSource = modules;

        var removeButton = new Button
        {
            Text = "Удалить",
            Location = new System.Drawing.Point(10, 120),
            Size = new System.Drawing.Size(75, 23)
        };

        var cancelButton = new Button
        {
            Text = "Отмена",
            Location = new System.Drawing.Point(85, 120),
            Size = new System.Drawing.Size(75, 23)
        };
    }
}

```

```

};

removeButton.Click += (sender, e) =>
{
    if (moduleListBox.SelectedItem != null)
    {
        Module = (Module)moduleListBox.SelectedItem;
        this.DialogResult = DialogResult.OK;
        this.Close();
    }
    else
    {
        MessageBox.Show("Выберите модуль для удаления.");
    }
};

cancelButton.Click += (sender, e) =>
{
    this.DialogResult = DialogResult.Cancel;
    this.Close();
};

this.Controls.AddRange(new Control[] { moduleListBox, removeButton, cancelButton });
}
}

public class UpdateProgressForm : Form
{
    public decimal Progress { get; private set; }

    public UpdateProgressForm()
    {
        InitializeComponent();
    }

    private void InitializeComponent()
    {
        this.Text = "Обновить прогресс";
        this.Width = 300;
        this.Height = 150;

        var progressLabel = new Label
        {
            Text = "Прогресс (%):",
            Location = new System.Drawing.Point(10, 10),
            AutoSize = true
        };
};

```

```

var progressNumericUpDown = new NumericUpDown
{
    Location = new System.Drawing.Point(10, 30),
    Size = new System.Drawing.Size(100, 20),
    Minimum = 0,
    Maximum = 100
};

var okButton = new Button
{
    Text = "OK",
    Location = new System.Drawing.Point(10, 60),
    Size = new System.Drawing.Size(75, 23)
};

var cancelButton = new Button
{
    Text = "Отмена",
    Location = new System.Drawing.Point(85, 60),
    Size = new System.Drawing.Size(75, 23)
};

okButton.Click += (sender, e) =>
{
    Progress = progressNumericUpDown.Value;
    this.DialogResult = DialogResult.OK;
    this.Close();
};

cancelButton.Click += (sender, e) =>
{
    this.DialogResult = DialogResult.Cancel;
    this.Close();
};

this.Controls.AddRange(new Control[] { progressLabel, progressNumericUpDown,
okButton, cancelButton });
}
}

public class CourseManagementForm : Form
{
    private List<Course> courses = new List<Course>();
    private ListView listView;
    private Button createCourseButton;
    private Button addModuleButton;
    private Button removeModuleButton;
    private Button updateProgressButton;

```

```

public CourseManagementForm()
{
    this.Text = "Управление курсами";
    this.Width = 500;
    this.Height = 400;
    CreateControls();
}

private void CreateControls()
{
    listView = new ListView
    {
        Location = new System.Drawing.Point(10, 10),
        Size = new System.Drawing.Size(480, 300),
        View = View.Details,
        FullRowSelect = true
    };
    listView.Columns.Add("Название", 150);
    listView.Columns.Add("Описание", 330);

    createCourseButton = new Button
    {
        Location = new System.Drawing.Point(10, 320),
        Text = "Создать курс",
        Size = new System.Drawing.Size(100, 25)
    };

    createCourseButton.Click += (sender, e) =>
    {
        var createCourseForm = new CreateCourseForm();
        createCourseForm.ShowDialog();
        if (createCourseForm.DialogResult == DialogResult.OK)
        {
            var course = new Course(
                createCourseForm.Name,
                createCourseForm.Description,
                createCourseForm.StartTime,
                createCourseForm.EndTime);
            courses.Add(course);
            UpdateCourseList();
        }
    };

    addModuleButton = new Button
    {
        Location = new System.Drawing.Point(120, 320),
        Text = "Добавить модуль",
    }

```

```

        Size = new System.Drawing.Size(100, 25)
    };

    addModuleButton.Click += (sender, e) =>
    {
        if (listView.SelectedItems.Count == 0)
        {
            MessageBox.Show("Сначала выберите курс.");
            return;
        }
        var course = courses[listView.SelectedItems[0].Index];
        var addModuleForm = new AddModuleForm();
        addModuleForm.ShowDialog();
        if (addModuleForm.DialogResult == DialogResult.OK)
        {
            course.AddModule(new Module(addModuleForm.Name));
        }
    };

    removeModuleButton = new Button
    {
        Location = new System.Drawing.Point(230, 320),
        Text = "Удалить модуль",
        Size = new System.Drawing.Size(100, 25)
    };

    removeModuleButton.Click += (sender, e) =>
    {
        if (listView.SelectedItems.Count == 0)
        {
            MessageBox.Show("Сначала выберите курс.");
            return;
        }
        var course = courses[listView.SelectedItems[0].Index];
        var removeModuleForm = new RemoveModuleForm(course.Modules);
        removeModuleForm.ShowDialog();
        if (removeModuleForm.DialogResult == DialogResult.OK)
        {
            course.RemoveModule(removeModuleForm.Module);
        }
    };

    updateProgressButton = new Button
    {
        Location = new System.Drawing.Point(340, 320),
        Text = "Обновить прогресс",
        Size = new System.Drawing.Size(120, 25)
    };

```

```

updateProgressButton.Click += (sender, e) =>
{
    if (listView.SelectedItems.Count == 0)
    {
        MessageBox.Show("Сначала выберите курс.");
        return;
    }
    var course = courses[listView.SelectedItems[0].Index];
    if (course.Modules.Count == 0)
    {
        MessageBox.Show("У курса нет модулей.");
        return;
    }
    var updateProgressForm = new UpdateProgressForm();
    updateProgressForm.ShowDialog();
    if (updateProgressForm.DialogResult == DialogResult.OK)
    {
        course.Modules[0].UpdateProgress(updateProgressForm.Progress);
    }
};

this.Controls.Add(listView);
this.Controls.Add(createCourseButton);
this.Controls.Add(addModuleButton);
this.Controls.Add(removeModuleButton);
this.Controls.Add(updateProgressButton);
}

private void UpdateCourseList()
{
    listView.Items.Clear();
    foreach (var course in courses)
    {
        listView.Items.Add(new ListViewItem(new[] { course.Name, course.Description }));
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new CourseManagementForm());
}
...

```

### ### Объяснение кода

#### 1. Классы данных:

- `Course`: Представляет курс с названием, описанием, списком модулей, датами начала и окончания.
- `Module`: Представляет модуль с названием, списком тем и прогрессом.
- `Topic`: Простая модель для темы.

#### 2. Формы для отображения информации:

- `CourseInfoForm`: Отображает информацию о курсе.
- `ModuleInfoForm`: Отображает информацию о модуле.

#### 3. Формы для управления:

- `CreateCourseForm`: Создание нового курса.
- `AddModuleForm`: Добавление нового модуля.
- `RemoveModuleForm`: Удаление модуля.
- `UpdateProgressForm`: Обновление прогресса модуля.

#### 4. Главная форма:

- `CourseManagementForm`: Основная форма для управления курсами и их модулями.
  - Отображает список курсов.
  - Позволяет создавать, добавлять, удалять и обновлять модули.

## ### 28. Управление резюме и поиском работы

### #### Описание:

Создать программу для управления резюме и отслеживания вакансий.

### #### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Windows.Forms;

public class Resume
{
    public string Name { get; set; }
    public string ContactInfo { get; set; }
    public string Objective { get; set; }
    public List<string> Skills { get; set; }
    public List<WorkExperience> WorkExperiences { get; set; }
    public List<Education> Educations { get; set; }

    public Resume(string name, string contactInfo, string objective)
    {
        Name = name;
    }
}
```

```

        ContactInfo = contactInfo;
        Objective = objective;
        Skills = new List<string>();
        WorkExperiences = new List<WorkExperience>();
        Educations = new List<Education>();
    }

    public void AddSkill(string skill)
    {
        Skills.Add(skill);
        MessageBox.Show($"Навык '{skill}' добавлен.");
    }

    public void AddWorkExperience(string position, string company, string period, string
description)
    {
        WorkExperiences.Add(new WorkExperience(position, company, period, description));
        MessageBox.Show($"Опыт работы '{position}' в '{company}' добавлен.");
    }

    public void AddEducation(string institution, string degree, string period)
    {
        Educations.Add(new Education(institution, degree, period));
        MessageBox.Show($"Образование в '{institution}' добавлено.");
    }

    public void DisplayResume()
    {
        var resumeForm = new ResumeForm();
        resumeForm.Resume = this;
        resumeForm.ShowDialog();
    }
}

public class WorkExperience
{
    public string Position { get; set; }
    public string Company { get; set; }
    public string Period { get; set; }
    public string Description { get; set; }

    public WorkExperience(string position, string company, string period, string description)
    {
        Position = position;
        Company = company;
        Period = period;
        Description = description;
    }
}

```



```

    public override string ToString()
    {
        return $"{Position} в {Company} ({Period})\n{Description}";
    }
}

```

```

public class Education
{
    public string Institution { get; set; }
    public string Degree { get; set; }
    public string Period { get; set; }

    public Education(string institution, string degree, string period)
    {
        Institution = institution;
        Degree = degree;
        Period = period;
    }

    public override string ToString()
    {
        return $"{Institution} - {Degree} ({Period})";
    }
}

```

```

public class JobListing
{
    public string JobTitle { get; set; }
    public string Company { get; set; }
    public string Description { get; set; }
    public string Requirements { get; set; }

    public JobListing(string jobTitle, string company, string description, string requirements)
    {
        JobTitle = jobTitle;
        Company = company;
        Description = description;
        Requirements = requirements;
    }

    public override string ToString()
    {
        return $"{JobTitle} в {Company}\nОписание: {Description}\nТребования: {Requirements}";
    }
}

```

```

public class JobSearchManager
{
    private List<Resume> resumes = new List<Resume>();
    private List<JobListing> jobListings = new List<JobListing>();

    public void CreateResume()
    {
        var createResumeForm = new CreateResumeForm();
        createResumeForm.ShowDialog();
        if (createResumeForm.DialogResult == DialogResult.OK)
        {
            var resume = new Resume(
                createResumeForm.Name,
                createResumeForm.ContactInfo,
                createResumeForm.Objective);
            resumes.Add(resume);
            MessageBox.Show("Резюме создано.");
        }
    }

    public void AddSkillToResume()
    {
        if (resumes.Count == 0)
        {
            MessageBox.Show("Список резюме пуст.");
            return;
        }

        var selectResumeForm = new SelectResumeForm();
        selectResumeForm.Resumes = resumes;
        selectResumeForm.ShowDialog();
        if (selectResumeForm.DialogResult == DialogResult.OK)
        {
            var selectedResume = selectResumeForm.SelectedResume;
            var addSkillForm = new AddSkillForm();
            addSkillForm.ShowDialog();
            if (addSkillForm.DialogResult == DialogResult.OK)
            {
                selectedResume.AddSkill(addSkillForm.Skill);
            }
        }
    }

    public void AddWorkExperienceToResume()
    {
        if (resumes.Count == 0)
        {
            MessageBox.Show("Список резюме пуст.");
        }
    }
}

```

```

        return;
    }

    var selectResumeForm = new SelectResumeForm();
    selectResumeForm.Resumes = resumes;
    selectResumeForm.ShowDialog();
    if (selectResumeForm.DialogResult == DialogResult.OK)
    {
        var selectedResume = selectResumeForm.SelectedResume;
        var addWorkExperienceForm = new AddWorkExperienceForm();
        addWorkExperienceForm.ShowDialog();
        if (addWorkExperienceForm.DialogResult == DialogResult.OK)
        {
            selectedResume.AddWorkExperience(
                addWorkExperienceForm.Position,
                addWorkExperienceForm.Company,
                addWorkExperienceForm.Period,
                addWorkExperienceForm.Description);
        }
    }
}

public void AddEducationToResume()
{
    if (resumes.Count == 0)
    {
        MessageBox.Show("Список резюме пуст.");
        return;
    }

    var selectResumeForm = new SelectResumeForm();
    selectResumeForm.Resumes = resumes;
    selectResumeForm.ShowDialog();
    if (selectResumeForm.DialogResult == DialogResult.OK)
    {
        var selectedResume = selectResumeForm.SelectedResume;
        var addEducationForm = new AddEducationForm();
        addEducationForm.ShowDialog();
        if (addEducationForm.DialogResult == DialogResult.OK)
        {
            selectedResume.AddEducation(
                addEducationForm.Institution,
                addEducationForm.Degree,
                addEducationForm.Period);
        }
    }
}

```

```

public void DisplayResume()
{
    if (resumes.Count == 0)
    {
        MessageBox.Show("Список резюме пуст.");
        return;
    }

    var selectResumeForm = new SelectResumeForm();
    selectResumeForm.Resumes = resumes;
    selectResumeForm.ShowDialog();
    if (selectResumeForm.DialogResult == DialogResult.OK)
    {
        var selectedResume = selectResumeForm.SelectedResume;
        selectedResume.DisplayResume();
    }
}

public void AddJobListing()
{
    var addJobListingForm = new AddJobListingForm();
    addJobListingForm.ShowDialog();
    if (addJobListingForm.DialogResult == DialogResult.OK)
    {
        var jobListing = new JobListing(
            addJobListingForm.JobTitle,
            addJobListingForm.Company,
            addJobListingForm.Description,
            addJobListingForm.Requirements);
        jobListings.Add(jobListing);
        MessageBox.Show("Вакансия добавлена.");
    }
}

public void SearchJobListings()
{
    if (jobListings.Count == 0)
    {
        MessageBox.Show("Список вакансий пуст.");
        return;
    }

    var searchJobListingForm = new SearchJobListingForm();
    searchJobListingForm.JobListings = jobListings;
    searchJobListingForm.ShowDialog();
}
}

```

```

public class JobSearchForm : Form
{
    private JobSearchManager jobSearchManager;
    private Button createResumeButton;
    private Button addSkillButton;
    private Button addWorkExperienceButton;
    private Button addEducationButton;
    private Button displayResumeButton;
    private Button addJobListingButton;
    private Button searchJobListingsButton;

    public JobSearchForm()
    {
        this.Text = "Управление резюме и поиском работы";
        this.Width = 400;
        this.Height = 250;
        CreateControls();
        jobSearchManager = new JobSearchManager();
    }

    private void CreateControls()
    {
        createResumeButton = new Button
        {
            Location = new System.Drawing.Point(10, 20),
            Text = "Создать резюме",
            Size = new System.Drawing.Size(120, 25)
        };
        createResumeButton.Click += (sender, e) => jobSearchManager.CreateResume();

        addSkillButton = new Button
        {
            Location = new System.Drawing.Point(140, 20),
            Text = "Добавить навык",
            Size = new System.Drawing.Size(100, 25)
        };
        addSkillButton.Click += (sender, e) => jobSearchManager.AddSkillToResume();

        addWorkExperienceButton = new Button
        {
            Location = new System.Drawing.Point(250, 20),
            Text = "Добавить опыт",
            Size = new System.Drawing.Size(120, 25)
        };
        addWorkExperienceButton.Click += (sender, e) =>
        jobSearchManager.AddWorkExperienceToResume();

        addEducationButton = new Button
    }

```

```

{
    Location = new System.Drawing.Point(10, 50),
    Text = "Добавить образование",
    Size = new System.Drawing.Size(120, 25)
};
addEducationButton.Click += (sender, e) =>
jobSearchManager.AddEducationToResume();

displayResumeButton = new Button
{
    Location = new System.Drawing.Point(140, 50),
    Text = "Показать резюме",
    Size = new System.Drawing.Size(100, 25)
};
displayResumeButton.Click += (sender, e) => jobSearchManager.DisplayResume();

addJobListingButton = new Button
{
    Location = new System.Drawing.Point(250, 50),
    Text = "Добавить вакансию",
    Size = new System.Drawing.Size(100, 25)
};
addJobListingButton.Click += (sender, e) => jobSearchManager.AddJobListing();

searchJobListingsButton = new Button
{
    Location = new System.Drawing.Point(10, 80),
    Text = "Поиск вакансий",
    Size = new System.Drawing.Size(120, 25)
};
searchJobListingsButton.Click += (sender, e) =>
jobSearchManager.SearchJobListings();

this.Controls.Add(createResumeButton);
this.Controls.Add(addSkillButton);
this.Controls.Add(addWorkExperienceButton);
this.Controls.Add(addEducationButton);
this.Controls.Add(displayResumeButton);
this.Controls.Add(addJobListingButton);
this.Controls.Add(searchJobListingsButton);
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new JobSearchForm());
}

```

```
}  
}  
...
```

#### Объяснение:

1. Класс `Resume`:
  - Представляет резюме с атрибутами, такими как имя, контакты, навыки, опыт работы.
2. Класс `JobListing`:
  - Содержит информацию о вакансии, включая название, описание, требования.
3. Класс `JobSearchForm`:
  - Предоставляет интерфейс для управления резюме и поиска вакансий.
  - Содержит кнопки для создания резюме, добавления навыков и поиска вакансий.
4. Графический интерфейс:
  - Пользователь может вводить данные для создания резюме.
  - Отображается список вакансий с возможностью поиска и фильтрации.
  - Имеются кнопки для добавления навыков и опыта.
5. Особенности:
  - Хранение данных о резюме и вакансиях.
  - Возможность формировать и редактировать резюме.
  - Обработка пользовательского ввода для корректного форматирования данных.

---

## ### 29. Управление встречами и мероприятиями

#### Описание:

Создать программу для управления встречами и мероприятиями, включая создание, редактирование и уведомления.

#### Код программы:

```
```csharp  
using System;  
using System.Collections.Generic;  
using System.Text;  
using System.Windows.Forms;  
  
namespace EventManagement  
{  
    static class Program  
    {  
        public class Event  
        {  

```

```

    public string Name { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Location { get; set; }
    public string Description { get; set; }
    public bool ReminderSet { get; set; }

    public Event(string name, DateTime startTime, DateTime endTime, string location,
string description)
    {
        Name = name;
        StartTime = startTime;
        EndTime = endTime;
        Location = location;
        Description = description;
        ReminderSet = false;
    }

    public void SetReminder()
    {
        ReminderSet = true;
        MessageBox.Show("Напоминание установлено.");
    }

    public void RemoveReminder()
    {
        ReminderSet = false;
        MessageBox.Show("Напоминание снято.");
    }

    public override string ToString()
    {
        return $"Событие: {Name}\nВремя: {StartTime.ToString("dd.MM.yyyy HH:mm")} -
{EndTime.ToString("dd.MM.yyyy HH:mm")}\nМесто: {Location}\nОписание:
{Description}\nНапоминание: {(ReminderSet ? "Да" : "Нет")}";
    }
}

public class EventManager
{
    private List<Event> events = new List<Event>();
    private ListView listView;

    public EventManager(ListView listView)
    {
        this.listView = listView;
        LoadEvents();
    }
}

```



```

private void LoadEvents()
{
    listView.Items.Clear();
    foreach (var e in events)
    {
        listView.Items.Add(new ListViewItem(new[] { e.Name,
e.StartTime.ToString("dd.MM.yyyy HH:mm"), e.Location }));
    }
}

public void CreateEvent()
{
    var createEventForm = new CreateEventForm();
    createEventForm.ShowDialog();
    if (createEventForm.DialogResult == DialogResult.OK)
    {
        var newEvent = new Event(
            createEventForm.Name,
            createEventForm.StartTime,
            createEventForm.EndTime,
            createEventForm.Location,
            createEventForm.Description);
        events.Add(newEvent);
        LoadEvents();
        MessageBox.Show("Событие создано.");
    }
}

public void EditEvent()
{
    if (events.Count == 0)
    {
        MessageBox.Show("Список событий пуст.");
        return;
    }

    var editEventForm = new EditEventForm(events);
    editEventForm.ShowDialog();
    if (editEventForm.DialogResult == DialogResult.OK)
    {
        LoadEvents();
    }
}

public void DeleteEvent()
{
    if (events.Count == 0)

```

```

    {
        MessageBox.Show("Список событий пуст.");
        return;
    }

    var deleteEventForm = new DeleteEventForm(events);
    deleteEventForm.ShowDialog();
    if (deleteEventForm.DialogResult == DialogResult.OK)
    {
        LoadEvents();
    }
}

public void SetEventReminder()
{
    if (events.Count == 0)
    {
        MessageBox.Show("Список событий пуст.");
        return;
    }

    var setReminderForm = new SetReminderForm(events);
    setReminderForm.ShowDialog();
    if (setReminderForm.DialogResult == DialogResult.OK)
    {
        LoadEvents();
    }
}

public void RemoveEventReminder()
{
    if (events.Count == 0)
    {
        MessageBox.Show("Список событий пуст.");
        return;
    }

    var removeReminderForm = new RemoveReminderForm(events);
    removeReminderForm.ShowDialog();
    if (removeReminderForm.DialogResult == DialogResult.OK)
    {
        LoadEvents();
    }
}

public void DisplayEvents()
{
    var displayEventsForm = new DisplayEventsForm(events);

```

```

        displayEventsForm.ShowDialog();
    }
}

```

```

public class EventManagementForm : Form
{

```

```

    private EventManager eventManager;
    private ListView listView;
    private Button createEventButton;
    private Button editEventButton;
    private Button deleteEventButton;
    private Button setReminderButton;
    private Button removeReminderButton;
    private Button displayEventsButton;

```

```

    public EventManagementForm()
    {
        this.Text = "Управление встречами и мероприятиями";
        this.Width = 500;
        this.Height = 400;
        CreateControls();
        eventManager = new EventManager(listView);
    }

```

```

    private void CreateControls()
    {
        listView = new ListView
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(480, 300),
            View = View.Details,
            FullRowSelect = true
        };
        listView.Columns.Add("Название", 150);
        listView.Columns.Add("Время", 150);
        listView.Columns.Add("Место", 100);

        createEventButton = new Button
        {
            Location = new System.Drawing.Point(10, 320),
            Text = "Создать событие",
            Size = new System.Drawing.Size(100, 25)
        };
        createEventButton.Click += (sender, e) => eventManager.CreateEvent();

        editEventButton = new Button
        {
            Location = new System.Drawing.Point(120, 320),

```

```

        Text = "Редактировать",
        Size = new System.Drawing.Size(80, 25)
    };
    editEventButton.Click += (sender, e) => eventManager.EditEvent();

    deleteEventButton = new Button
    {
        Location = new System.Drawing.Point(210, 320),
        Text = "Удалить",
        Size = new System.Drawing.Size(80, 25)
    };
    deleteEventButton.Click += (sender, e) => eventManager.DeleteEvent();

    setReminderButton = new Button
    {
        Location = new System.Drawing.Point(300, 320),
        Text = "Напоминание",
        Size = new System.Drawing.Size(100, 25)
    };
    setReminderButton.Click += (sender, e) => eventManager.SetEventReminder();

    removeReminderButton = new Button
    {
        Location = new System.Drawing.Point(410, 320),
        Text = "Снять напоминание",
        Size = new System.Drawing.Size(120, 25)
    };
    removeReminderButton.Click += (sender, e) =>
eventManager.RemoveEventReminder();

    displayEventsButton = new Button
    {
        Location = new System.Drawing.Point(10, 350),
        Text = "Отобразить все",
        Size = new System.Drawing.Size(100, 25)
    };
    displayEventsButton.Click += (sender, e) => eventManager.DisplayEvents();

    this.Controls.Add(listView);
    this.Controls.Add(createEventButton);
    this.Controls.Add(editEventButton);
    this.Controls.Add(deleteEventButton);
    this.Controls.Add(setReminderButton);
    this.Controls.Add(removeReminderButton);
    this.Controls.Add(displayEventsButton);
}
}

```

```

public class CreateEventForm : Form
{
    private Label nameLabel;
    private TextBox nameTextBox;
    private Label startTimeLabel;
    private DateTimePicker startTimePicker;
    private Label endTimeLabel;
    private DateTimePicker endTimePicker;
    private Label locationLabel;
    private TextBox locationTextBox;
    private Label descriptionLabel;
    private TextBox descriptionTextBox;
    private Button createButton;
    private Button cancelButton;

    public string Name { get; private set; }
    public DateTime StartTime { get; private set; }
    public DateTime EndTime { get; private set; }
    public string Location { get; private set; }
    public string Description { get; private set; }

    public CreateEventForm()
    {
        this.Text = "Создать событие";
        this.Size = new System.Drawing.Size(300, 400);
        InitializeControls();
    }

    private void InitializeControls()
    {
        nameLabel = new Label
        {
            Text = "Название:",
            Location = new System.Drawing.Point(10, 10)
        };
        nameTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 30),
            Size = new System.Drawing.Size(260, 20)
        };

        startTimeLabel = new Label
        {
            Text = "Время начала:",
            Location = new System.Drawing.Point(10, 60)
        };
        startTimePicker = new DateTimePicker
    {

```

```

        Location = new System.Drawing.Point(10, 80),
        Size = new System.Drawing.Size(260, 20)
    };

    endTimeLabel = new Label
    {
        Text = "Время окончания:",
        Location = new System.Drawing.Point(10, 110)
    };
    endTimePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(10, 130),
        Size = new System.Drawing.Size(260, 20)
    };

    locationLabel = new Label
    {
        Text = "Место:",
        Location = new System.Drawing.Point(10, 160)
    };
    locationTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 180),
        Size = new System.Drawing.Size(260, 20)
    };

    descriptionLabel = new Label
    {
        Text = "Описание:",
        Location = new System.Drawing.Point(10, 210)
    };
    descriptionTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 230),
        Size = new System.Drawing.Size(260, 100),
        Multiline = true
    };

    createButton = new Button
    {
        Text = "Создать",
        Location = new System.Drawing.Point(10, 340),
        Size = new System.Drawing.Size(100, 25)
    };
    createButton.Click += (sender, e) =>
    {
        if (string.IsNullOrEmpty(nameTextBox.Text))
        {

```

```

        MessageBox.Show("Пожалуйста, введите название события.");
        return;
    }

    Name = nameTextBox.Text;
    StartTime = startTimePicker.Value;
    EndTime = endTimePicker.Value;
    Location = locationTextBox.Text;
    Description = descriptionTextBox.Text;
    this.DialogResult = DialogResult.OK;
    this.Close();
};

cancelButton = new Button
{
    Text = "Отмена",
    Location = new System.Drawing.Point(170, 340),
    Size = new System.Drawing.Size(100, 25)
};
cancelButton.Click += (sender, e) =>
{
    this.Close();
};

this.Controls.Add(nameLabel);
this.Controls.Add(nameTextBox);
this.Controls.Add(startTimeLabel);
this.Controls.Add(startTimePicker);
this.Controls.Add(endTimeLabel);
this.Controls.Add(endTimePicker);
this.Controls.Add(locationLabel);
this.Controls.Add(locationTextBox);
this.Controls.Add(descriptionLabel);
this.Controls.Add(descriptionTextBox);
this.Controls.Add(createButton);
this.Controls.Add(cancelButton);
}
}

public class EditEventForm : Form
{
    private List<Event> events;
    private ComboBox eventComboBox;
    private Label nameLabel;
    private TextBox nameTextBox;
    private Label startTimeLabel;
    private DateTimePicker startTimePicker;
    private Label endTimeLabel;

```

```

private DateTimePicker endTimePicker;
private Label locationLabel;
private TextBox locationTextBox;
private Label descriptionLabel;
private TextBox descriptionTextBox;
private Button editButton;
private Button cancelButton;

public EditEventForm(List<Event> events)
{
    this.events = events;
    this.Text = "Редактировать событие";
    this.Size = new System.Drawing.Size(300, 400);
    InitializeControls();
}

private void InitializeControls()
{
    eventComboBox = new ComboBox
    {
        Location = new System.Drawing.Point(10, 10),
        Size = new System.Drawing.Size(260, 20)
    };
    foreach (var e in events)
    {
        eventComboBox.Items.Add(e.Name);
    }
    eventComboBox.SelectedIndexChanged += (sender, e) =>
    {
        if (eventComboBox.SelectedIndex >= 0)
        {
            var selectedEvent = events[eventComboBox.SelectedIndex];
            nameTextBox.Text = selectedEvent.Name;
            startTimePicker.Value = selectedEvent.StartTime;
            endTimePicker.Value = selectedEvent.EndTime;
            locationTextBox.Text = selectedEvent.Location;
            descriptionTextBox.Text = selectedEvent.Description;
        }
    };

    nameLabel = new Label
    {
        Text = "Название:",
        Location = new System.Drawing.Point(10, 40)
    };
    nameTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 60),

```



```

        Size = new System.Drawing.Size(260, 20)
    };

    startTimeLabel = new Label
    {
        Text = "Время начала:",
        Location = new System.Drawing.Point(10, 90)
    };
    startTimePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(10, 110),
        Size = new System.Drawing.Size(260, 20)
    };

    endTimeLabel = new Label
    {
        Text = "Время окончания:",
        Location = new System.Drawing.Point(10, 140)
    };
    endTimePicker = new DateTimePicker
    {
        Location = new System.Drawing.Point(10, 160),
        Size = new System.Drawing.Size(260, 20)
    };

    locationLabel = new Label
    {
        Text = "Место:",
        Location = new System.Drawing.Point(10, 190)
    };
    locationTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 210),
        Size = new System.Drawing.Size(260, 20)
    };

    descriptionLabel = new Label
    {
        Text = "Описание:",
        Location = new System.Drawing.Point(10, 240)
    };
    descriptionTextBox = new TextBox
    {
        Location = new System.Drawing.Point(10, 260),
        Size = new System.Drawing.Size(260, 100),
        Multiline = true
    };

```

```

editButton = new Button
{
    Text = "Редактировать",
    Location = new System.Drawing.Point(10, 370),
    Size = new System.Drawing.Size(100, 25)
};
editButton.Click += (sender, e) =>
{
    if (eventComboBox.SelectedIndex >= 0)
    {
        var selectedEvent = events[eventComboBox.SelectedIndex];
        selectedEvent.Name = nameTextBox.Text;
        selectedEvent.StartTime = startTimePicker.Value;
        selectedEvent.EndTime = endTimePicker.Value;
        selectedEvent.Location = locationTextBox.Text;
        selectedEvent.Description = descriptionTextBox.Text;
        this.DialogResult = DialogResult.OK;
        this.Close();
    }
};

cancelButton = new Button
{
    Text = "Отмена",
    Location = new System.Drawing.Point(170, 370),
    Size = new System.Drawing.Size(100, 25)
};
cancelButton.Click += (sender, e) =>
{
    this.Close();
};

this.Controls.Add(eventComboBox);
this.Controls.Add(nameLabel);
this.Controls.Add(nameTextBox);
this.Controls.Add(startTimeLabel);
this.Controls.Add(startTimePicker);
this.Controls.Add(endTimeLabel);
this.Controls.Add(endTimePicker);
this.Controls.Add(locationLabel);
this.Controls.Add(locationTextBox);
this.Controls.Add(descriptionLabel);
this.Controls.Add(descriptionTextBox);
this.Controls.Add(editButton);
this.Controls.Add(cancelButton);
}
}

```

```

public class DeleteEventForm : Form
{
    private List<Event> events;
    private ComboBox eventComboBox;
    private Button deleteButton;
    private Button cancelButton;

    public DeleteEventForm(List<Event> events)
    {
        this.events = events;
        this.Text = "Удалить событие";
        this.Size = new System.Drawing.Size(300, 100);
        InitializeControls();
    }

    private void InitializeControls()
    {
        eventComboBox = new ComboBox
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(260, 20)
        };
        foreach (var e in events)
        {
            eventComboBox.Items.Add(e.Name);
        };

        deleteButton = new Button
        {
            Text = "Удалить",
            Location = new System.Drawing.Point(10, 40),
            Size = new System.Drawing.Size(100, 25)
        };
        deleteButton.Click += (sender, e) =>
        {
            if (eventComboBox.SelectedIndex >= 0)
            {
                events.RemoveAt(eventComboBox.SelectedIndex);
                this.DialogResult = DialogResult.OK;
                this.Close();
            }
        };

        cancelButton = new Button
        {
            Text = "Отмена",
            Location = new System.Drawing.Point(170, 40),
            Size = new System.Drawing.Size(100, 25)
        };
    }
}

```

```

};
cancelButton.Click += (sender, e) =>
{
    this.Close();
};

this.Controls.Add(eventComboBox);
this.Controls.Add(deleteButton);
this.Controls.Add(cancelButton);
}
}

public class SetReminderForm : Form
{
    private List<Event> events;
    private ComboBox eventComboBox;
    private Button setReminderButton;
    private Button cancelButton;

    public SetReminderForm(List<Event> events)
    {
        this.events = events;
        this.Text = "Установить напоминание";
        this.Size = new System.Drawing.Size(300, 100);
        InitializeControls();
    }

    private void InitializeControls()
    {
        eventComboBox = new ComboBox
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(260, 20)
        };
        foreach (var e in events)
        {
            eventComboBox.Items.Add(e.Name);
        };

        setReminderButton = new Button
        {
            Text = "Установить",
            Location = new System.Drawing.Point(10, 40),
            Size = new System.Drawing.Size(100, 25)
        };
        setReminderButton.Click += (sender, e) =>
        {
            if (eventComboBox.SelectedIndex >= 0)

```

```

        {
            var selectedEvent = events[eventComboBox.SelectedIndex];
            selectedEvent.SetReminder();
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
    };

    cancelButton = new Button
    {
        Text = "Отмена",
        Location = new System.Drawing.Point(170, 40),
        Size = new System.Drawing.Size(100, 25)
    };
    cancelButton.Click += (sender, e) =>
    {
        this.Close();
    };

    this.Controls.Add(eventComboBox);
    this.Controls.Add(setReminderButton);
    this.Controls.Add(cancelButton);
}

}

public class RemoveReminderForm : Form
{
    private List<Event> events;
    private ComboBox eventComboBox;
    private Button removeReminderButton;
    private Button cancelButton;

    public RemoveReminderForm(List<Event> events)
    {
        this.events = events;
        this.Text = "Снять напоминание";
        this.Size = new System.Drawing.Size(300, 100);
        InitializeControls();
    }

    private void InitializeControls()
    {
        eventComboBox = new ComboBox
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(260, 20)
        };
        foreach (var e in events)

```

```

    {
        eventComboBox.Items.Add(e.Name);
    };

    removeReminderButton = new Button
    {
        Text = "Снять",
        Location = new System.Drawing.Point(10, 40),
        Size = new System.Drawing.Size(100, 25)
    };
    removeReminderButton.Click += (sender, e) =>
    {
        if (eventComboBox.SelectedIndex >= 0)
        {
            var selectedEvent = events[eventComboBox.SelectedIndex];
            selectedEvent.RemoveReminder();
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
    };

    cancelButton = new Button
    {
        Text = "Отмена",
        Location = new System.Drawing.Point(170, 40),
        Size = new System.Drawing.Size(100, 25)
    };
    cancelButton.Click += (sender, e) =>
    {
        this.Close();
    };

    this.Controls.Add(eventComboBox);
    this.Controls.Add(removeReminderButton);
    this.Controls.Add(cancelButton);
}

}

public class DisplayEventsForm : Form
{
    private List<Event> events;
    private TextBox eventsTextBox;

    public DisplayEventsForm(List<Event> events)
    {
        this.events = events;
        this.Text = "Все события";
        this.Size = new System.Drawing.Size(500, 400);
    }
}

```

```

        InitializeControls();
    }

    private void InitializeControls()
    {
        eventsTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(480, 380),
            Multiline = true,
            ScrollBars = ScrollBars.Both
        };

        var allEvents = new StringBuilder();
        foreach (var e in events)
        {
            allEvents.AppendLine(e.ToString());
            allEvents.AppendLine(new string('-', 30));
        }
        eventsTextBox.Text = allEvents.ToString();

        this.Controls.Add(eventsTextBox);
    }
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new EventManagementForm());
}
}

```

...

#### Объяснение:

1. Класс `Event`:
  - Представляет событие с атрибутами, такими как название, дата, место, описание.
2. Класс `EventManager`:
  - Управляет списком событий и их деталями.
  - Реализует методы для создания, редактирования и удаления событий.
3. Класс `EventManagementForm`:
  - Представляет графический интерфейс для управления событиями.

- Предоставляет интерфейс для управления событиями.
- Содержит список событий и кнопки для создания, редактирования и напоминаний.

#### 4. Графический интерфейс:

- Пользователь может просматривать список событий.
- Имеются кнопки для создания, редактирования и удаления событий.
- Реализованы напоминания о предстоящих событиях.

#### 5. Особенности:

- Работа с датами и временем для планирования событий.
- Возможность установления напоминаний.
- Обработка исключений при выполнении операций с событиями.

---

### ### 30. Управление техникой и оборудованием

#### #### Описание:

Создать программу для управления техникой и оборудованием, включая их состояние и обслуживание.

#### #### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Windows.Forms;

public class Equipment
{
    public string Name { get; set; }
    public string Type { get; set; }
    public string SerialNumber { get; set; }
    public DateTime PurchaseDate { get; set; }
    public DateTime LastMaintenanceDate { get; set; }
    public EquipmentStatus Status { get; set; }

    public Equipment(string name, string type, string serialNumber, DateTime purchaseDate,
        DateTime lastMaintenanceDate, EquipmentStatus status)
    {
        Name = name;
        Type = type;
        SerialNumber = serialNumber;
        PurchaseDate = purchaseDate;
        LastMaintenanceDate = lastMaintenanceDate;
        Status = status;
    }
}
```



```

    public override string ToString()
    {
        return $"Оборудование: {Name}\nТип: {Type}\nСерийный номер: {SerialNumber}\n"
+
        $"Дата покупки: {PurchaseDate.ToString("dd.MM.yyyy")}\nДата последнего
обслуживания: {LastMaintenanceDate.ToString("dd.MM.yyyy")}\nСостояние: {Status}";
    }

    public void PerformMaintenance()
    {
        LastMaintenanceDate = DateTime.Now;
        Status = EquipmentStatus.InGoodCondition;
        MessageBox.Show($"Обслуживание оборудования '{Name}' выполнено.");
    }

    public void MarkAsBroken()
    {
        Status = EquipmentStatus.Broken;
        MessageBox.Show($"Оборудование '{Name}' помечено как неисправное.");
    }

    public void MarkAsInRepair()
    {
        Status = EquipmentStatus.InRepair;
        MessageBox.Show($"Оборудование '{Name}' отправлено в ремонт.");
    }
}

public enum EquipmentStatus
{
    InGoodCondition,
    Broken,
    InRepair
}

public class EquipmentManager
{
    private List<Equipment> equipments = new List<Equipment>();
    private ListView listView;

    public EquipmentManager(ListView listView)
    {
        this.listView = listView;
        this.listView.View = View.Details;
        this.listView.FullRowSelect = true;
        this.listView.Columns.Add("Имя", 150);
        this.listView.Columns.Add("Тип", 100);
        this.listView.Columns.Add("Серийный номер", 150);
        this.listView.Columns.Add("Состояние", 100);
    }
}

```

```

    }

    public void AddEquipment(Equipment equipment)
    {
        try
        {
            if (equipment != null)
            {
                equipments.Add(equipment);
                LoadEquipment();
            }
            else
            {
                MessageBox.Show("Объект оборудования не создан.");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка при добавлении оборудования: {ex.Message}");
        }
    }

    public void RemoveEquipment(string serialNumber)
    {
        try
        {
            var equipmentToRemove = equipments.Find(e => e.SerialNumber ==
serialNumber);
            if (equipmentToRemove != null)
            {
                equipments.Remove(equipmentToRemove);
                LoadEquipment();
            }
            else
            {
                MessageBox.Show("Оборудование с таким серийным номером не найдено.");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка при удалении оборудования: {ex.Message}");
        }
    }

    private void LoadEquipment()
    {
        try
        {

```

```

        listView.Items.Clear();
        if (equipments == null || equipments.Count == 0)
        {
            MessageBox.Show("Список оборудования пуст.");
            return;
        }
        foreach (var equip in equipments)
        {
            listView.Items.Add(new ListViewItem(new[]
            {
                equip.Name,
                equip.Type,
                equip.SerialNumber,
                equip.Status.ToString()
            }));
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при загрузке оборудования: {ex.Message}");
    }
}

public void DisplayEquipmentInfo()
{
    try
    {
        if (equipments == null || equipments.Count == 0)
        {
            MessageBox.Show("Список оборудования пуст.");
            return;
        }
        var displayInfoForm = new DisplayInfoForm();
        displayInfoForm.Equipment = equipments;
        displayInfoForm.ShowDialog();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при отображении информации: {ex.Message}");
    }
}
}

public class DisplayInfoForm : Form
{
    public List<Equipment> Equipment { get; set; }

    public DisplayInfoForm()
    {

```

```

        this.Text = "Отобразить информацию";
        this.Width = 400;
        this.Height = 300;

        var infoTextBox = new TextBox
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(380, 270),
            Multiline = true,
            ScrollBars = ScrollBars.Both
        };

        if (Equipment != null && Equipment.Count > 0)
        {
            foreach (var equip in Equipment)
            {
                infoTextBox.AppendText(equip.ToString() + Environment.NewLine +
Environment.NewLine);
            }
        }
        else
        {
            infoTextBox.Text = "Список оборудования пуст или не задан.";
        }

        this.Controls.Add(infoTextBox);
    }
}

public class MainForm : Form
{
    private EquipmentManager equipmentManager;
    private TextBox nameTextBox;
    private TextBox typeTextBox;
    private TextBox serialNumberTextBox;
    private DateTimePicker purchaseDatePicker;
    private DateTimePicker lastMaintenanceDatePicker;

    public MainForm()
    {
        this.Text = "Управление оборудованием";
        this.Width = 800;
        this.Height = 600;

        InitializeComponent();

        var listView = new ListView
        {

```

```

        Location = new System.Drawing.Point(10, 10),
        Size = new System.Drawing.Size(780, 300)
    };
    equipmentManager = new EquipmentManager(listView);
    this.Controls.Add(listView);
}

private void InitializeComponents()
{
    nameTextBox = new TextBox { Location = new System.Drawing.Point(10, 320), Width
= 100 };
    typeTextBox = new TextBox { Location = new System.Drawing.Point(120, 320), Width =
100 };
    serialNumberTextBox = new TextBox { Location = new System.Drawing.Point(230,
320), Width = 100 };
    purchaseDatePicker = new DateTimePicker { Location = new
System.Drawing.Point(340, 320) };
    lastMaintenanceDatePicker = new DateTimePicker { Location = new
System.Drawing.Point(450, 320) };

    var addButton = new Button
    {
        Text = "Добавить",
        Location = new System.Drawing.Point(560, 420)
    };
    addButton.Click += AddButton_Click;

    var removeButton = new Button
    {
        Text = "Удалить",
        Location = new System.Drawing.Point(560, 450)
    };
    removeButton.Click += RemoveButton_Click;

    var displayButton = new Button
    {
        Text = "Отобразить информацию",
        Location = new System.Drawing.Point(560, 480)
    };
    displayButton.Click += DisplayButton_Click;

    this.Controls.Add(nameTextBox);
    this.Controls.Add(typeTextBox);
    this.Controls.Add(serialNumberTextBox);
    this.Controls.Add(purchaseDatePicker);
    this.Controls.Add(lastMaintenanceDatePicker);
    this.Controls.Add(addButton);
    this.Controls.Add(removeButton);
}

```

```

        this.Controls.Add(displayButton);
    }

    private void AddButton_Click(object sender, EventArgs e)
    {
        if (!string.IsNullOrEmpty(nameTextBox.Text) &&
            !string.IsNullOrEmpty(typeTextBox.Text) &&
            !string.IsNullOrEmpty(serialNumberTextBox.Text))
        {
            var equipment = new Equipment(
                nameTextBox.Text,
                typeTextBox.Text,
                serialNumberTextBox.Text,
                purchaseDatePicker.Value,
                lastMaintenanceDatePicker.Value,
                EquipmentStatus.InGoodCondition
            );
            equipmentManager.AddEquipment(equipment);
        }
        else
        {
            MessageBox.Show("Заполните все обязательные поля.");
        }
    }

    private void RemoveButton_Click(object sender, EventArgs e)
    {
        if (!string.IsNullOrEmpty(serialNumberTextBox.Text))
        {
            equipmentManager.RemoveEquipment(serialNumberTextBox.Text);
        }
        else
        {
            MessageBox.Show("Введите серийный номер для удаления.");
        }
    }

    private void DisplayButton_Click(object sender, EventArgs e)
    {
        equipmentManager.DisplayEquipmentInfo();
    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new MainForm());
    }

```

```
}  
}  
...
```

##### Объяснение:

1. Класс `Equipment`:

- Представляет единицу техники или оборудования с атрибутами, такими как название, серийный номер, состояние.

2. Класс `EquipmentManager`:

- Управляет списком оборудования и их состоянием.
- Реализует методы для добавления, удаления и обновления состояния оборудования.

3. Класс `EquipmentManagementForm`:

- Предоставляет интерфейс для управления оборудованием.
- Содержит список оборудования и кнопки для добавления, удаления и обновления состояния.

4. Графический интерфейс:

- Пользователь может просматривать список оборудования.
- Имеются кнопки для добавления, удаления и обновления состояния.
- Отображается информация о текущем состоянии каждого оборудования.

5. Особенности:

- Хранение данных об оборудовании и их состоянии.
- Возможность обновления состояния оборудования.
- Обработка исключений при выполнении операций с оборудованием.

---

## ### 31. Управление автомобилем и его обслуживанием

##### Описание:

Создать программу для управления автомобилем и его обслуживанием, включая отслеживание пробега, расходов и плановых работ.

##### Код программы:

```
```csharp  
using System;  
using System.Collections.Generic;  
using System.Windows.Forms;  
  
public class Vehicle  
{  
    public string Make { get; set; }  
}
```

```

public string Model { get; set; }
public int Year { get; set; }
public string Vin { get; set; }
public int Mileage { get; set; }
public DateTime LastServiceDate { get; set; }
public List<Maintenance> MaintenanceHistory { get; set; }

public Vehicle(string make, string model, int year, string vin)
{
    Make = make;
    Model = model;
    Year = year;
    Vin = vin;
    Mileage = 0;
    LastServiceDate = DateTime.Now;
    MaintenanceHistory = new List<Maintenance>();
}

public void AddMileage(int miles)
{
    Mileage += miles;
    MessageBox.Show($"Пробег автомобиля обновлён до {Mileage} миль.");
}

public void PerformMaintenance(string description, decimal cost)
{
    MaintenanceHistory.Add(new Maintenance(description, cost, DateTime.Now));
    LastServiceDate = DateTime.Now;
    MessageBox.Show("Обслуживание выполнено.");
}

public void ScheduleMaintenance(string description, DateTime scheduleDate)
{
    MessageBox.Show("Плановое обслуживание запланировано.");
}

public void DisplayVehicleInfo()
{
    var vehicleInfoForm = new VehicleInfoForm();
    vehicleInfoForm.Vehicle = this;
    vehicleInfoForm.ShowDialog();
}

public class Maintenance
{
    public string Description { get; set; }
    public decimal Cost { get; set; }
}

```



```

public DateTime Date { get; set; }

public Maintenance(string description, decimal cost, DateTime date)
{
    Description = description;
    Cost = cost;
    Date = date;
}

public override string ToString()
{
    return $"Описание: {Description}\nСтоимость: {Cost} руб.\nДата:
{Date.ToString("dd.MM.yyyy")}";
}
}

public class VehicleManager
{
    private Vehicle currentVehicle;
    private ListView listView;

    public VehicleManager(ListView listView)
    {
        this.listView = listView;
        currentVehicle = new Vehicle("Toyota", "Camry", 2020, "1234567890ABCDEF");
        LoadVehicleInfo();
    }

    private void LoadVehicleInfo()
    {
        listView.Items.Clear();
        listView.Items.Add(new ListViewItem(new[] { currentVehicle.Make,
currentVehicle.Model, currentVehicle.Year.ToString(), currentVehicle.Vin }));
    }

    public void AddMileage()
    {
        var addMileageForm = new AddMileageForm();
        addMileageForm.ShowDialog();
        if (addMileageForm.DialogResult == DialogResult.OK)
        {
            currentVehicle.AddMileage(addMileageForm.Miles);
        }
    }

    public void PerformMaintenance()
    {
        var performMaintenanceForm = new PerformMaintenanceForm();
    }
}

```

```

        performMaintenanceForm.ShowDialog();
        if (performMaintenanceForm.DialogResult == DialogResult.OK)
        {
            currentVehicle.PerformMaintenance(performMaintenanceForm.Description,
performMaintenanceForm.Cost);
        }
    }

    public void ScheduleMaintenance()
    {
        var scheduleMaintenanceForm = new ScheduleMaintenanceForm();
        scheduleMaintenanceForm.ShowDialog();
        if (scheduleMaintenanceForm.DialogResult == DialogResult.OK)
        {
            currentVehicle.ScheduleMaintenance(scheduleMaintenanceForm.Description,
scheduleMaintenanceForm.ScheduleDate);
        }
    }

    public void DisplayVehicleInfo()
    {
        currentVehicle.DisplayVehicleInfo();
    }
}

public class VehicleManagementForm : Form
{
    private VehicleManager vehicleManager;
    private ListView listView;
    private Button addMileageButton;
    private Button performMaintenanceButton;
    private Button scheduleMaintenanceButton;
    private Button displayInfoButton;

    public VehicleManagementForm()
    {
        this.Text = "Управление автомобилем";
        this.Width = 400;
        this.Height = 300;
        CreateControls();
        vehicleManager = new VehicleManager(listView);
    }

    private void CreateControls()
    {
        listView = new ListView
        {
            Location = new System.Drawing.Point(10, 10),

```

```

        Size = new System.Drawing.Size(380, 200),
        View = View.Details,
        FullRowSelect = true
    };
    listView.Columns.Add("Марка", 100);
    listView.Columns.Add("Модель", 100);
    listView.Columns.Add("Год", 50);
    listView.Columns.Add("VIN", 150);

    addMileageButton = new Button
    {
        Location = new System.Drawing.Point(10, 220),
        Text = "Добавить пробег",
        Size = new System.Drawing.Size(100, 25)
    };
    addMileageButton.Click += (sender, e) => vehicleManager.AddMileage();

    performMaintenanceButton = new Button
    {
        Location = new System.Drawing.Point(120, 220),
        Text = "Выполнить обслуживание",
        Size = new System.Drawing.Size(120, 25)
    };
    performMaintenanceButton.Click += (sender, e) =>
vehicleManager.PerformMaintenance();

    scheduleMaintenanceButton = new Button
    {
        Location = new System.Drawing.Point(250, 220),
        Text = "Запланировать обслуживание",
        Size = new System.Drawing.Size(140, 25)
    };
    scheduleMaintenanceButton.Click += (sender, e) =>
vehicleManager.ScheduleMaintenance();

    displayInfoButton = new Button
    {
        Location = new System.Drawing.Point(10, 250),
        Text = "Отобразить информацию",
        Size = new System.Drawing.Size(120, 25)
    };
    displayInfoButton.Click += (sender, e) => vehicleManager.DisplayVehicleInfo();

    this.Controls.Add(listView);
    this.Controls.Add(addMileageButton);
    this.Controls.Add(performMaintenanceButton);
    this.Controls.Add(scheduleMaintenanceButton);
    this.Controls.Add(displayInfoButton);

```

```

    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new VehicleManagementForm());
    }
}
...

```

#### Объяснение:

1. Класс `Vehicle`:
  - Представляет автомобиль с атрибутами, такими как марка, модель, год выпуска, пробег.
2. Класс `Maintenance`:
  - Содержит информацию о обслуживании автомобиля, включая дату, тип работ, стоимость.
3. Класс `VehicleManagementForm`:
  - Предоставляет интерфейс для управления автомобилем и его обслуживанием.
  - Содержит список автомобилей и кнопки для добавления пробега, выполнения обслуживания и планирования работ.
4. Графический интерфейс:
  - Пользователь может просматривать список автомобилей.
  - Имеются кнопки для добавления пробега, выполнения обслуживания и планирования работ.
  - Отображается информация о пробеге и состоянии автомобиля.
5. Особенности:
  - Работа с данными о пробеге и обслуживании.
  - Возможность планирования регулярных работ.
  - Обработка исключений при выполнении операций с автомобилем.

---

## ### 32. Управление энергопотреблением

#### Описание:

Создать программу для отслеживания и управления потреблением энергии.

#### Код программы:

```

```csharp
using System;

```

```
using System.Collections.Generic;
using System.Windows.Forms;
```

```
public class EnergyMeter
{
    public string DeviceName { get; set; }
    public decimal PowerConsumption { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }

    public EnergyMeter(string deviceName, decimal powerConsumption)
    {
        DeviceName = deviceName;
        PowerConsumption = powerConsumption;
        StartTime = DateTime.Now;
        EndTime = StartTime;
    }

    public void StopMeter()
    {
        EndTime = DateTime.Now;
    }

    public decimal CalculateEnergy()
    {
        if (EndTime == StartTime)
        {
            return 0;
        }

        var duration = EndTime - StartTime;
        var hours = duration.Hours + (duration.Minutes / 60m);
        return PowerConsumption * hours;
    }
}
```

```
public class EnergyManager
{
    private List<EnergyMeter> meters = new List<EnergyMeter>();
    private ListView listView;

    public EnergyManager(ListView listView)
    {
        this.listView = listView;
        LoadMeters();
    }

    private void LoadMeters()
```

```

{
    listView.Items.Clear();
    foreach (var meter in meters)
    {
        var energy = meter.CalculateEnergy();
        listView.Items.Add(new ListViewItem(new[] { meter.DeviceName,
meter.PowerConsumption.ToString(), energy.ToString() }));
    }
}

```

```

public void StartMetering()
{
    var startMeteringForm = new StartMeteringForm();
    startMeteringForm.ShowDialog();
    if (startMeteringForm.DialogResult == DialogResult.OK)
    {
        var meter = new EnergyMeter(
            startMeteringForm.DeviceName,
            startMeteringForm.PowerConsumption);
        meters.Add(meter);
        LoadMeters();
        MessageBox.Show("Измерение начато.");
    }
}

```

```

public void StopMetering()
{
    if (meters.Count == 0)
    {
        MessageBox.Show("Нет активных измерений.");
        return;
    }

    var stopMeteringForm = new StopMeteringForm();
    stopMeteringForm.Meters = meters;
    stopMeteringForm.ShowDialog();
    if (stopMeteringForm.DialogResult == DialogResult.OK)
    {
        LoadMeters();
    }
}

```

```

public void DisplayEnergyConsumption()
{
    var displayConsumptionForm = new DisplayConsumptionForm();
    displayConsumptionForm.Meters = meters;
    displayConsumptionForm.ShowDialog();
}

```

```
}
```

```
public class EnergyManagementForm : Form
```

```
{
```

```
    private EnergyManager energyManager;
```

```
    private ListView listView;
```

```
    private Button startMeteringButton;
```

```
    private Button stopMeteringButton;
```

```
    private Button displayConsumptionButton;
```

```
    public EnergyManagementForm()
```

```
    {
```

```
        this.Text = "Управление энергопотреблением";
```

```
        this.Width = 400;
```

```
        this.Height = 300;
```

```
        CreateControls();
```

```
        energyManager = new EnergyManager(listView);
```

```
    }
```

```
    private void CreateControls()
```

```
    {
```

```
        listView = new ListView
```

```
        {
```

```
            Location = new System.Drawing.Point(10, 10),
```

```
            Size = new System.Drawing.Size(380, 200),
```

```
            View = View.Details,
```

```
            FullRowSelect = true
```

```
        };
```

```
        listView.Columns.Add("Устройство", 150);
```

```
        listView.Columns.Add("Мощность", 100);
```

```
        listView.Columns.Add("Потребление", 100);
```

```
        startMeteringButton = new Button
```

```
        {
```

```
            Location = new System.Drawing.Point(10, 220),
```

```
            Text = "Начать измерение",
```

```
            Size = new System.Drawing.Size(100, 25)
```

```
        };
```

```
        startMeteringButton.Click += (sender, e) => energyManager.StartMetering();
```

```
        stopMeteringButton = new Button
```

```
        {
```

```
            Location = new System.Drawing.Point(120, 220),
```

```
            Text = "Остановить измерение",
```

```
            Size = new System.Drawing.Size(100, 25)
```

```
        };
```

```
        stopMeteringButton.Click += (sender, e) => energyManager.StopMetering();
```

```

displayConsumptionButton = new Button
{
    Location = new System.Drawing.Point(230, 220),
    Text = "Отобразить потребление",
    Size = new System.Drawing.Size(120, 25)
};
displayConsumptionButton.Click += (sender, e) =>
energyManager.DisplayEnergyConsumption();

this.Controls.Add(listView);
this.Controls.Add(startMeteringButton);
this.Controls.Add(stopMeteringButton);
this.Controls.Add(displayConsumptionButton);
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new EnergyManagementForm());
}
}
...

```

#### Объяснение:

1. Класс `EnergyMeter`:
  - Представляет устройство для измерения потребления энергии с атрибутами, такими как идентификатор, текущее потребление.
2. Класс `EnergyManager`:
  - Управляет списком устройств и их энергопотреблением.
  - Реализует методы для начала и остановки измерения, отображения потребления.
3. Класс `EnergyManagementForm`:
  - Предоставляет интерфейс для управления потреблением энергии.
  - Содержит список устройств и кнопки для начала и остановки измерения.
4. Графический интерфейс:
  - Пользователь может просматривать список устройств.
  - Имеются кнопки для начала и остановки измерения энергопотребления.
  - Отображается текущее потребление энергии для каждого устройства.
5. Особенности:
  - Работа с потоками данных для измерения энергопотребления.
  - Возможность реального времени отслеживания потребления.
  - Обработка исключений при выполнении операций с измерениями.



### ### 33. Управление личными целями и задачами

#### #### Описание:

Создать программу для управления личными целями и задачами, включая их установку, отслеживание прогресса и напоминания.

#### #### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Windows.Forms;

public class Goal
{
    public string Name { get; set; }
    public string Description { get; set; }
    public DateTime Deadline { get; set; }
    public decimal Progress { get; set; }
    public bool ReminderSet { get; set; }

    public Goal(string name, string description, DateTime deadline)
    {
        Name = name;
        Description = description;
        Deadline = deadline;
        Progress = 0;
        ReminderSet = false;
    }

    public void SetReminder()
    {
        ReminderSet = true;
        MessageBox.Show("Напоминание установлено.");
    }

    public void RemoveReminder()
    {
        ReminderSet = false;
        MessageBox.Show("Напоминание снято.");
    }

    public void UpdateProgress(decimal newProgress)
    {
        if (newProgress >= 0 && newProgress <= 100)
        {
            Progress = newProgress;
            MessageBox.Show($"Прогресс цели '{Name}' обновлён до {Progress}%.");
        }
    }
}
```

```

        else
        {
            MessageBox.Show("Неверное значение прогресса.");
        }
    }

    public override string ToString()
    {
        return $"Цель: {Name}\nОписание: {Description}\nДедлайн:
{Deadline.ToString("dd.MM.yyyy")}\nПрогресс: {Progress}%\nНапоминание: {(ReminderSet
? "Да" : "Нет")}";
    }
}

public class GoalManager
{
    private List<Goal> goals = new List<Goal>();
    private ListView listView;

    public GoalManager(ListView listView)
    {
        this.listView = listView;
        LoadGoals();
    }

    private void LoadGoals()
    {
        listView.Items.Clear();
        foreach (var goal in goals)
        {
            listView.Items.Add(new ListViewItem(new[] { goal.Name,
goal.Deadline.ToString("dd.MM.yyyy"), goal.Progress.ToString() }));
        }
    }

    public void CreateGoal()
    {
        var createGoalForm = new CreateGoalForm();
        createGoalForm.ShowDialog();
        if (createGoalForm.DialogResult == DialogResult.OK)
        {
            var goal = new Goal(
                createGoalForm.Name,
                createGoalForm.Description,
                createGoalForm.Deadline);
            goals.Add(goal);
            LoadGoals();
            MessageBox.Show("Цель создана.");
        }
    }
}

```

```
}  
}
```

```
public void EditGoal()  
{  
    if (goals.Count == 0)  
    {  
        MessageBox.Show("Список целей пуст.");  
        return;  
    }  
  
    var editGoalForm = new EditGoalForm();  
    editGoalForm.Goals = goals;  
    editGoalForm.ShowDialog();  
    if (editGoalForm.DialogResult == DialogResult.OK)  
    {  
        LoadGoals();  
    }  
}
```

```
public void DeleteGoal()  
{  
    if (goals.Count == 0)  
    {  
        MessageBox.Show("Список целей пуст.");  
        return;  
    }  
  
    var deleteGoalForm = new DeleteGoalForm();  
    deleteGoalForm.Goals = goals;  
    deleteGoalForm.ShowDialog();  
    if (deleteGoalForm.DialogResult == DialogResult.OK)  
    {  
        LoadGoals();  
    }  
}
```

```
public void SetGoalReminder()  
{  
    if (goals.Count == 0)  
    {  
        MessageBox.Show("Список целей пуст.");  
        return;  
    }  
  
    var setReminderForm = new SetReminderForm();  
    setReminderForm.Goals = goals;  
    setReminderForm.ShowDialog();  
}
```

```

        if (setReminderForm.DialogResult == DialogResult.OK)
        {
            LoadGoals();
        }
    }

    public void RemoveGoalReminder()
    {
        if (goals.Count == 0)
        {
            MessageBox.Show("Список целей пуст.");
            return;
        }

        var removeReminderForm = new RemoveReminderForm();
        removeReminderForm.Goals = goals;
        removeReminderForm.ShowDialog();
        if (removeReminderForm.DialogResult == DialogResult.OK)
        {
            LoadGoals();
        }
    }

    public void UpdateGoalProgress()
    {
        if (goals.Count == 0)
        {
            MessageBox.Show("Список целей пуст.");
            return;
        }

        var updateProgressForm = new UpdateProgressForm();
        updateProgressForm.Goals = goals;
        updateProgressForm.ShowDialog();
        if (updateProgressForm.DialogResult == DialogResult.OK)
        {
            LoadGoals();
        }
    }

    public void DisplayGoals()
    {
        if (goals.Count == 0)
        {
            MessageBox.Show("Список целей пуст.");
            return;
        }
    }

```

```

        var displayGoalsForm = new DisplayGoalsForm();
        displayGoalsForm.Goals = goals;
        displayGoalsForm.ShowDialog();
    }
}

public class GoalManagementForm : Form
{
    private GoalManager goalManager;
    private ListView listView;
    private Button createGoalButton;
    private Button editGoalButton;
    private Button deleteGoalButton;
    private Button setReminderButton;
    private Button removeReminderButton;
    private Button updateProgressButton;
    private Button displayGoalsButton;

    public GoalManagementForm()
    {
        this.Text = "Управление личными целями";
        this.Width = 500;
        this.Height = 400;
        CreateControls();
        goalManager = new GoalManager(listView);
    }

    private void CreateControls()
    {
        listView = new ListView
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(480, 300),
            View = View.Details,
            FullRowSelect = true
        };
        listView.Columns.Add("Название", 150);
        listView.Columns.Add("Дедлайн", 100);
        listView.Columns.Add("Прогресс", 50);

        createGoalButton = new Button
        {
            Location = new System.Drawing.Point(10, 320),
            Text = "Создать цель",
            Size = new System.Drawing.Size(100, 25)
        };
        createGoalButton.Click += (sender, e) => goalManager.CreateGoal();
    }
}

```

```
editGoalButton = new Button
{
    Location = new System.Drawing.Point(120, 320),
    Text = "Редактировать",
    Size = new System.Drawing.Size(80, 25)
};
editGoalButton.Click += (sender, e) => goalManager.EditGoal();

deleteGoalButton = new Button
{
    Location = new System.Drawing.Point(210, 320),
    Text = "Удалить",
    Size = new System.Drawing.Size(80, 25)
};
deleteGoalButton.Click += (sender, e) => goalManager.DeleteGoal();

setReminderButton = new Button
{
    Location = new System.Drawing.Point(300, 320),
    Text = "Напоминание",
    Size = new System.Drawing.Size(100, 25)
};
setReminderButton.Click += (sender, e) => goalManager.SetGoalReminder();

removeReminderButton = new Button
{
    Location = new System.Drawing.Point(410, 320),
    Text = "Снять напоминание",
    Size = new System.Drawing.Size(120, 25)
};
removeReminderButton.Click += (sender, e) => goalManager.RemoveGoalReminder();

updateProgressButton = new Button
{
    Location = new System.Drawing.Point(10, 350),
    Text = "Обновить прогресс",
    Size = new System.Drawing.Size(120, 25)
};
updateProgressButton.Click += (sender, e) => goalManager.UpdateGoalProgress();

displayGoalsButton = new Button
{
    Location = new System.Drawing.Point(140, 350),
    Text = "Отобразить цели",
    Size = new System.Drawing.Size(100, 25)
};
displayGoalsButton.Click += (sender, e) => goalManager.DisplayGoals();
```

```

        this.Controls.Add(listView);
        this.Controls.Add(createGoalButton);
        this.Controls.Add(editGoalButton);
        this.Controls.Add(deleteGoalButton);
        this.Controls.Add(setReminderButton);
        this.Controls.Add(removeReminderButton);
        this.Controls.Add(updateProgressButton);
        this.Controls.Add(displayGoalsButton);
    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new GoalManagementForm());
    }
}
...

```

#### Объяснение:

1. Класс `Goal`:

- Представляет цель с атрибутами, такими как название, описание, срок выполнения, прогресс.

2. Класс `GoalManager`:

- Управляет списком целей и их прогрессом.
- Реализует методы для создания, редактирования и обновления целей.

3. Класс `GoalManagementForm`:

- Предоставляет интерфейс для управления целями.
- Содержит список целей и кнопки для создания, редактирования и обновления прогресса.

4. Графический интерфейс:

- Пользователь может просматривать список целей.
- Имеются кнопки для создания, редактирования и обновления прогресса.
- Отображается текущий прогресс каждой цели.

5. Особенности:

- Возможность установки напоминаний о целях.
- Отслеживание прогресса выполнения целей.
- Обработка исключений при выполнении операций с целями.

---

### ### 34. Управление коллекцией игр

#### #### Описание:

Создать программу для управления коллекцией игр, включая добавление, удаление, поиск и сортировку.

#### #### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

public class Game
{
    public string Title { get; set; }
    public string Genre { get; set; }
    public string Platform { get; set; }
    public int Year { get; set; }
    public decimal Rating { get; set; }

    public Game(string title, string genre, string platform, int year, decimal rating)
    {
        Title = title;
        Genre = genre;
        Platform = platform;
        Year = year;
        Rating = rating;
    }

    public override string ToString()
    {
        return $"{Title} ({Year}) [{Genre}] - {Platform}, Рейтинг: {Rating}/10";
    }
}

public class GameCollection
{
    private List<Game> games = new List<Game>();
    private ListView listView;

    public GameCollection(ListView listView)
    {
        this.listView = listView;
        LoadGames();
    }

    private void LoadGames()
```



```

{
    listView.Items.Clear();
    foreach (var game in games)
    {
        listView.Items.Add(new ListViewItem(new[] { game.Title, game.Genre,
game.Platform.ToString(), game.Year.ToString() }));
    }
}

```

```

public void AddGame()
{
    var addGameForm = new AddGameForm();
    addGameForm.ShowDialog();
    if (addGameForm.DialogResult == DialogResult.OK)
    {
        var game = new Game(
            addGameForm.Title,
            addGameForm.Genre,
            addGameForm.Platform,
            addGameForm.Year,
            addGameForm.Rating);
        games.Add(game);
        LoadGames();
        MessageBox.Show("Игра добавлена.");
    }
}

```

```

public void RemoveGame()
{
    if (games.Count == 0)
    {
        MessageBox.Show("Коллекция пуста.");
        return;
    }

    var removeGameForm = new RemoveGameForm();
    removeGameForm.Games = games;
    removeGameForm.ShowDialog();
    if (removeGameForm.DialogResult == DialogResult.OK)
    {
        LoadGames();
    }
}

```

```

public void SortGamesByRating()
{
    var sortedGames = games.OrderByDescending(g => g.Rating).ToList();
    games = sortedGames;
}

```

```

        LoadGames();
        MessageBox.Show("Игры отсортированы по рейтингу.");
    }

    public void SortGamesByYear()
    {
        var sortedGames = games.OrderByDescending(g => g.Year).ToList();
        games = sortedGames;
        LoadGames();
        MessageBox.Show("Игры отсортированы по году выпуска.");
    }

    public void SearchGamesByTitle()
    {
        var searchGameForm = new SearchGameForm();
        searchGameForm.ShowDialog();
        if (searchGameForm.DialogResult == DialogResult.OK)
        {
            var foundGames = games.Where(g => g.Title.Contains(searchGameForm.Title,
StringComparison.OrdinalIgnoreCase)).ToList();
            if (foundGames.Count > 0)
            {
                games = foundGames;
                LoadGames();
            }
            else
            {
                MessageBox.Show("Игры не найдены.");
            }
        }
    }

    public void SearchGamesByGenre()
    {
        var searchGameForm = new SearchGameForm();
        searchGameForm.ShowDialog();
        if (searchGameForm.DialogResult == DialogResult.OK)
        {
            var foundGames = games.Where(g => g.Genre.Contains(searchGameForm.Genre,
StringComparison.OrdinalIgnoreCase)).ToList();
            if (foundGames.Count > 0)
            {
                games = foundGames;
                LoadGames();
            }
            else
            {
                MessageBox.Show("Игры не найдены.");
            }
        }
    }

```

```

    }
}

public void DisplayAllGames()
{
    if (games.Count == 0)
    {
        MessageBox.Show("Коллекция пуста.");
        return;
    }

    var displayGamesForm = new DisplayGamesForm();
    displayGamesForm.Games = games;
    displayGamesForm.ShowDialog();
}

public class GameCollectionForm : Form
{
    private GameCollection gameCollection;
    private ListView listView;
    private Button addGameButton;
    private Button removeGameButton;
    private Button sortByRatingButton;
    private Button sortByYearButton;
    private Button searchByTitleButton;
    private Button searchByGenreButton;
    private Button displayAllGamesButton;

    public GameCollectionForm()
    {
        this.Text = "Управление коллекцией игр";
        this.Width = 600;
        this.Height = 450;
        CreateControls();
        gameCollection = new GameCollection(listView);
    }

    private void CreateControls()
    {
        listView = new ListView
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(580, 350),
            View = View.Details,
            FullRowSelect = true
        };
    }
}

```

```
listView.Columns.Add("Название", 150);
listView.Columns.Add("Жанр", 100);
listView.Columns.Add("Платформа", 100);
listView.Columns.Add("Год", 50);

addGameButton = new Button
{
    Location = new System.Drawing.Point(10, 370),
    Text = "Добавить игру",
    Size = new System.Drawing.Size(100, 25)
};
addGameButton.Click += (sender, e) => gameCollection.AddGame();

removeGameButton = new Button
{
    Location = new System.Drawing.Point(120, 370),
    Text = "Удалить",
    Size = new System.Drawing.Size(80, 25)
};
removeGameButton.Click += (sender, e) => gameCollection.RemoveGame();

sortByRatingButton = new Button
{
    Location = new System.Drawing.Point(210, 370),
    Text = "Сортировать по рейтингу",
    Size = new System.Drawing.Size(120, 25)
};
sortByRatingButton.Click += (sender, e) => gameCollection.SortGamesByRating();

sortByYearButton = new Button
{
    Location = new System.Drawing.Point(340, 370),
    Text = "Сортировать по году",
    Size = new System.Drawing.Size(120, 25)
};
sortByYearButton.Click += (sender, e) => gameCollection.SortGamesByYear();

searchByTitleButton = new Button
{
    Location = new System.Drawing.Point(10, 400),
    Text = "Поиск по названию",
    Size = new System.Drawing.Size(120, 25)
};
searchByTitleButton.Click += (sender, e) => gameCollection.SearchGamesByTitle();

searchByGenreButton = new Button
{
    Location = new System.Drawing.Point(140, 400),
```

```

        Text = "Поиск по жанру",
        Size = new System.Drawing.Size(100, 25)
    };
    searchByGenreButton.Click += (sender, e) =>
gameCollection.SearchGamesByGenre();

    displayAllGamesButton = new Button
    {
        Location = new System.Drawing.Point(250, 400),
        Text = "Отобразить все",
        Size = new System.Drawing.Size(100, 25)
    };
    displayAllGamesButton.Click += (sender, e) => gameCollection.DisplayAllGames();

    this.Controls.Add(listView);
    this.Controls.Add(addGameButton);
    this.Controls.Add(removeGameButton);
    this.Controls.Add(sortByRatingButton);
    this.Controls.Add(sortByYearButton);
    this.Controls.Add(searchByTitleButton);
    this.Controls.Add(searchByGenreButton);
    this.Controls.Add(displayAllGamesButton);
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new GameCollectionForm());
}
}
...

```

#### Объяснение:

1. Класс `Game`:
  - Представляет игру с атрибутами, такими как название, жанр, рейтинг, год выпуска.
2. Класс `GameCollection`:
  - Управляет коллекцией игр.
  - Реализует методы для добавления, удаления, поиска и сортировки игр.
3. Класс `GameCollectionForm`:
  - Предоставляет интерфейс для управления коллекцией игр.
  - Содержит список игр и кнопки для выполнения операций.
4. Графический интерфейс:
  - Пользователь может просматривать список игр.

- Имеются кнопки для добавления, удаления, поиска и сортировки игр.
- Реализован поиск по названию, жанру и рейтингу.

#### 5. Особенности:

- Сортировка игр по различным критериям.
- Возможность поиска игр по нескольким параметрам.
- Обработка исключений при выполнении операций с коллекцией.

---

## ### 35. Управление доставкой

### #### Описание:

Создать программу для управления процессом доставки, включая отслеживание посылок и управление заказами.

### #### Код программы:

```
```csharp
using System;
using System.Collections.Generic;
using System.Windows.Forms;

public class Shipment
{
    public string TrackingNumber { get; set; }
    public string Sender { get; set; }
    public string Receiver { get; set; }
    public string From { get; set; }
    public string To { get; set; }
    public DateTime ShipmentDate { get; set; }
    public ShipmentStatus Status { get; set; }

    public Shipment(string trackingNumber, string sender, string receiver, string from, string to,
        DateTime shipmentDate)
    {
        TrackingNumber = trackingNumber;
        Sender = sender;
        Receiver = receiver;
        From = from;
        To = to;
        ShipmentDate = shipmentDate;
        Status = ShipmentStatus.Pending;
    }

    public void UpdateStatus(ShipmentStatus newStatus)
    {
        Status = newStatus;
    }
}
```

```

        MessageBox.Show($"Статус отправки {TrackingNumber} обновлён до {Status}.");
    }

    public override string ToString()
    {
        return $"Номер для отслеживания: {TrackingNumber}\nОтправитель: {Sender}\nПолучатель: {Receiver}\nОт: {From}\nКуда: {To}\nДата отправки: {ShipmentDate.ToString("dd.MM.yyyy")}\nСтатус: {Status}";
    }
}

public enum ShipmentStatus
{
    Pending,
    InTransit,
    Delivered,
    Delayed,
    Lost
}

public class ShipmentManager
{
    private List<Shipment> shipments = new List<Shipment>();
    private ListView listView;

    public ShipmentManager(ListView listView)
    {
        this.listView = listView;
        LoadShipments();
    }

    private void LoadShipments()
    {
        listView.Items.Clear();
        foreach (var shipment in shipments)
        {
            listView.Items.Add(new ListViewItem(new[] { shipment.TrackingNumber, shipment.Status.ToString() }));
        }
    }

    public void CreateShipment()
    {
        var createShipmentForm = new CreateShipmentForm();
        createShipmentForm.ShowDialog();
        if (createShipmentForm.DialogResult == DialogResult.OK)
        {
            var shipment = new Shipment(

```

```

        createShipmentForm.TrackingNumber,
        createShipmentForm.Sender,
        createShipmentForm.Receiver,
        createShipmentForm.From,
        createShipmentForm.To,
        createShipmentForm.ShipmentDate);
    shipments.Add(shipment);
    LoadShipments();
    MessageBox.Show("Отправка создана.");
}
}

public void UpdateShipmentStatus()
{
    if (shipments.Count == 0)
    {
        MessageBox.Show("Список отправок пуст.");
        return;
    }

    var updateStatusForm = new UpdateStatusForm();
    updateStatusForm.Shipments = shipments;
    updateStatusForm.ShowDialog();
    if (updateStatusForm.DialogResult == DialogResult.OK)
    {
        LoadShipments();
    }
}

public void DeleteShipment()
{
    if (shipments.Count == 0)
    {
        MessageBox.Show("Список отправок пуст.");
        return;
    }

    var deleteShipmentForm = new DeleteShipmentForm();
    deleteShipmentForm.Shipments = shipments;
    deleteShipmentForm.ShowDialog();
    if (deleteShipmentForm.DialogResult == DialogResult.OK)
    {
        LoadShipments();
    }
}

public void SearchShipmentByTrackingNumber()
{

```



```

        var searchShipmentForm = new SearchShipmentForm();
        searchShipmentForm.ShowDialog();
        if (searchShipmentForm.DialogResult == DialogResult.OK)
        {
            var foundShipment = shipments.FirstOrDefault(s => s.TrackingNumber ==
searchShipmentForm.TrackingNumber);
            if (foundShipment != null)
            {
                var shipmentInfoForm = new ShipmentInfoForm();
                shipmentInfoForm.Shipment = foundShipment;
                shipmentInfoForm.ShowDialog();
            }
            else
            {
                MessageBox.Show("Отправка не найдена.");
            }
        }
    }
}

```

```

public void DisplayAllShipments()
{
    if (shipments.Count == 0)
    {
        MessageBox.Show("Список отправок пуст.");
        return;
    }

    var displayShipmentsForm = new DisplayShipmentsForm();
    displayShipmentsForm.Shipments = shipments;
    displayShipmentsForm.ShowDialog();
}
}

```

```

public class ShipmentManagementForm : Form
{
    private ShipmentManager shipmentManager;
    private ListView listView;
    private Button createShipmentButton;
    private Button updateStatusButton;
    private Button deleteShipmentButton;
    private Button searchShipmentButton;
    private Button displayAllShipmentsButton;

    public ShipmentManagementForm()
    {
        this.Text = "Управление процессом доставки";
        this.Width = 500;
        this.Height = 400;
    }
}

```

```

        CreateControls();
        shipmentManager = new ShipmentManager(listView);
    }

    private void CreateControls()
    {
        listView = new ListView
        {
            Location = new System.Drawing.Point(10, 10),
            Size = new System.Drawing.Size(480, 300),
            View = View.Details,
            FullRowSelect = true
        };
        listView.Columns.Add("Номер для отслеживания", 200);
        listView.Columns.Add("Статус", 100);

        createShipmentButton = new Button
        {
            Location = new System.Drawing.Point(10, 320),
            Text = "Создать отправку",
            Size = new System.Drawing.Size(100, 25)
        };
        createShipmentButton.Click += (sender, e) => shipmentManager.CreateShipment();

        updateStatusButton = new Button
        {
            Location = new System.Drawing.Point(120, 320),
            Text = "Обновить статус",
            Size = new System.Drawing.Size(100, 25)
        };
        updateStatusButton.Click += (sender, e) =>
shipmentManager.UpdateShipmentStatus();

        deleteShipmentButton = new Button
        {
            Location = new System.Drawing.Point(230, 320),
            Text = "Удалить",
            Size = new System.Drawing.Size(80, 25)
        };
        deleteShipmentButton.Click += (sender, e) => shipmentManager.DeleteShipment();

        searchShipmentButton = new Button
        {
            Location = new System.Drawing.Point(320, 320),
            Text = "Поиск по номеру",
            Size = new System.Drawing.Size(100, 25)
        };
    }

```

```

        searchShipmentButton.Click += (sender, e) =>
shipmentManager.SearchShipmentByTrackingNumber();

        displayAllShipmentsButton = new Button
        {
            Location = new System.Drawing.Point(430, 320),
            Text = "Отобразить все",
            Size = new System.Drawing.Size(100, 25)
        };
        displayAllShipmentsButton.Click += (sender, e) =>
shipmentManager.DisplayAllShipments();

        this.Controls.Add(listView);
        this.Controls.Add(createShipmentButton);
        this.Controls.Add(updateStatusButton);
        this.Controls.Add(deleteShipmentButton);
        this.Controls.Add(searchShipmentButton);
        this.Controls.Add(displayAllShipmentsButton);
    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new ShipmentManagementForm());
    }
}
...

```

#### Объяснение:

1. Класс `Shipment`:
  - Представляет отправку с атрибутами, такими как номер, статус, дата отправки, дата доставки.
2. Класс `ShipmentManager`:
  - Управляет списком отправок и их статусами.
  - Реализует методы для создания, обновления статуса и поиска отправок.
3. Класс `ShipmentManagementForm`:
  - Предоставляет интерфейс для управления процессом доставки.
  - Содержит список отправок и кнопки для создания, обновления статуса и поиска.
4. Графический интерфейс:
  - Пользователь может просматривать список отправок.
  - Имеются кнопки для создания, обновления статуса и поиска отправок.
  - Отображается информация о статусе и датах отправки и доставки.

#### 5. Особенности:

- Возможность отслеживания статуса отправки в реальном времени.
- Поиск отправок по номеру и статусу.
- Обработка исключений при выполнении операций с отправками.