МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

ассистент		Гуков С.Ю.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

ПРИЛОЖЕНИЕ ДЛЯ УЧЕТА РЕЙСОВ

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4128		Воробьев В.А.
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Оглавление

ВВЕДЕНИЕ

Цель курсового проектирования как учебной дисциплины: использование полученных знаний и навыков объектно-ориентированного программирования для разработки пользовательской библиотеки классов, предназначенной для программирования приложений пользователя в конкретной предметной области. Библиотека должна учитывать особенности области приложения и располагать необходимой функциональностью для построения интерфейса пользователя, структурирования, записи, чтения данных в различных форматах, обработки, вычисления и визуализации данных. Курсовая работа позволяет:

1. систематизировать теоретические знания и закрепить полученные практические умения по дисциплине «Основы программирования» в соответствии с требованиями к уровню подготовки по направлению 09.03.02 «Информационные системы и технологии»;
2. сформировать умения работы с учебной литературой и иными информационными источниками;
3. развить профессиональную письменную и устную речь;
4. развить системное мышление, творческую инициативу, самостоятельность, организованность и ответственность за принимаемые решения;
5. сформировать навыки планомерной регулярной работы над решением поставленных задач.

АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ

Вариант задания: в приложении пользователь может создать объект класса Самолет, содержащий следующую информацию: пункта назначения; номер рейса; название авиакомпании; дату и время отправления; стоимость полета. Предусмотреть свойства для получения состояния объекта. Описать класс Аэропорт, содержащий массив самолетов. Обеспечить следующие возможности: вывод информации о самолете по указанному номеру рейса; вывод информации о самолетах, отправляющихся после указанного времени, или в указанном направлении, или по совокупности параметров времени и направления

От архитектуры классов требуется:

1) Возможность создания модели данных рейса.
2) Возможность создания отфильтрованной коллекции рейсов. Фильтрация осуществляется:
    a. По месту назначения
    b. По дате вылета
    c. По международному коду
    d. По совокупности вышеуказанных параметров
3) Возможность изменения коллекции:
    a. Удаление рейса
    b. Добавление рейса
4) Возможность получения полей/свойств, характеризующих рейс.

От программного кода библиотеки требуется:

1) Реализация полиморфизма.
2) Реализация наследования.
3) Реализация инкапсуляции.
4) Программный код должен содержать минимум 4 класса.
5) Программный код должен содержать минимум 1 интерфейс.
6) Программный код должен выполнять поставленную задачу для библиотеки.

Также требуется на основе спроектированной библиотеки построить интерфейс и протестировать его. На основе интерфейса показать взаимодействия пользователя с написанной нами библиотекой.

Полученная библиотека должна использоваться для контроля рейсов в аэропорте. С помощью библиотеки пользователь должен с легкостью контролировать процесс контроля рейсов, иметь доступный функционал фильтрации результатов, а также иметь возможность синхронизировать модель данных рейса с элементами контроля в WinForms.

ДИАГРАММА КЛАССОВ

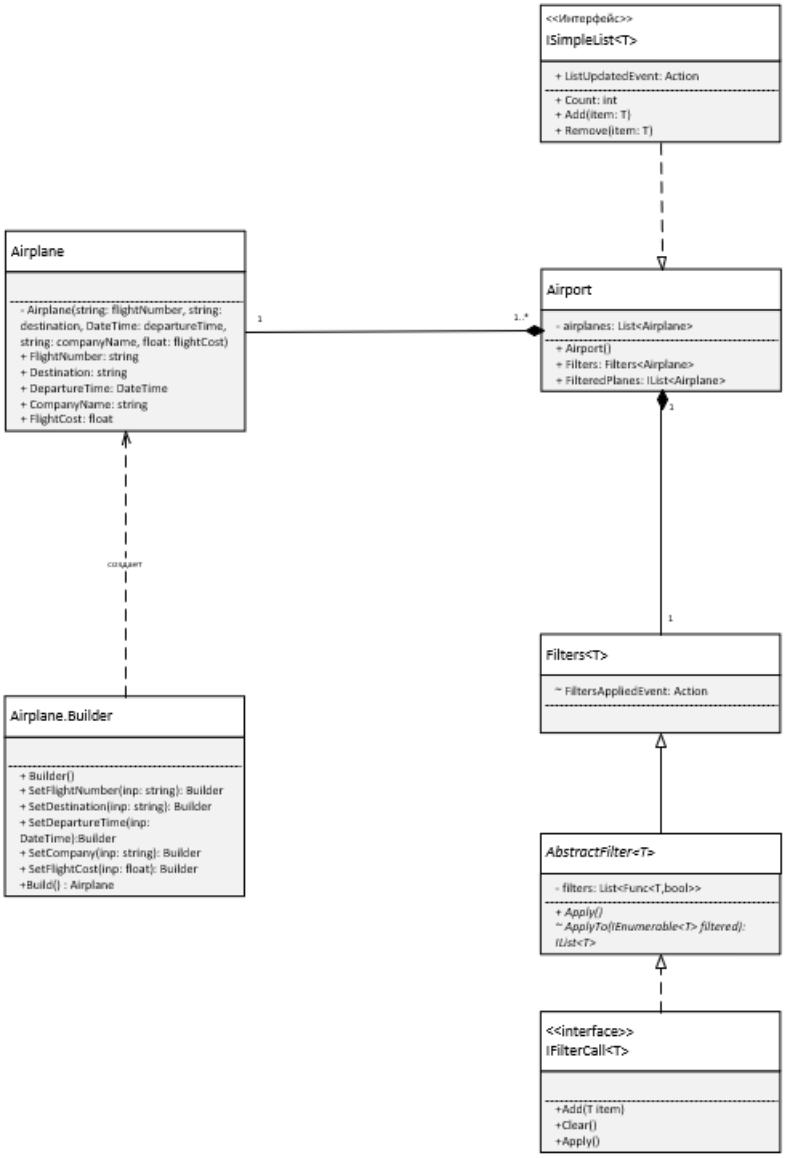


Рисунок 1 – диаграмма классов

На рисунке 1 видно, что:

1) Класс Airplane.Builder находится в зависимости от класса Airplane
2) Класс Airport связан с классом Airplane посредством композиции. Класс Airplane включен в Airport.
3) Filters<T> находится в Airport посредством композиции.
4) Класс Airport реализует интерфейс ISimpleList<T>.
5) Класс Filter<T> наследуется от абстрактного класса AbstractFilter<T>, который реализует интерфейс IFilterCall<T>.

ОПИСАНИЕ ПРОГРАММНОГО КОДА КЛАССОВ

Ниже представлен класс Airplane, который является моделью самолета авиакомпании. Класс Airplane содержит автосвойства: FlightNumber (Номер рейса ИКАО), Destination (Пункт назначения), DepartureTime (Время вылета), CompanyName (Имя компании, отвечающего за самолет), FlightCost (Стоимость полета). Конструктор принимает на вход номер рейса, пункт назначения, время вылета, имя компании, и стоимость полета.

```
public class Airplane
{
    public string FlightNumber { get; init; }
    public string Destination { get; init; }
    public DateTime DepartuteTime { get; init; }
    public string CompanyName { get; init; }
    public float FlightCost { get; init; }

    private Airplane(string flightNumber, string destination, DateTime departuteTime, string companyName, float flightCost)
    {
        FlightNumber = flightNumber;
        Destination = destination;
        DepartuteTime = departuteTime;
        CompanyName = companyName;
        FlightCost = flightCost;
    }

    public class Builder...
}
```

Рисунок 2 – код класса Airplane

Для упрощения создания объекта класса Airplane, был спроектирован Builder. Он позволяет нам легко создавать и отслеживать некорректно введенные поля для создания экземпляра класса Airplane. Класс Builder имеет методы SetFlightNumber, SetDestination, SetDepartureTime, SetCompany, SetFlightCost, которые отвечают за передачу в соответствующие аргументы конструктора Airplane. Метод Build создает объект класса Airplane.

```
public class Builder
{
    private string? _flightNumber;
    private string? _destination = null;
    private DateTime? _departureTime = null;
    private string? _company = null;
    private float? _flightCost = null;

    public Builder SetFlightNumber(string inp)
    {
        if (inp.Trim().Length < 3 || inp.Trim().Length > 8)
        {
            _flightNumber = null;
            throw new ArgumentException();
        }
        _flightNumber = inp.Trim();
        return this;
    }

    public Builder SetDestination(string inp)
    {
        if (inp.Trim() == "")
        {
            _destination = null;
            throw new ArgumentException();
        }
        _destination = inp.Trim();
        return this;
    }

    public Builder SetDepartureTime(DateTime inp)
    {
        _departureTime = inp;
        return this;
    }

    public Builder SetCompany(string inp)
    {
        _company = inp.Trim();
        return this;
    }

    public Builder SetFlightCost(float inp)
    {
        if (inp < 0)
        {
            _flightCost = null;
            throw new ArgumentException();
        }
        _flightCost = inp;
        return this;
    }

    public Airplane Build()
    {
        if (_destination == null || _flightNumber == null ||
            _departureTime == null || _company == null || _flightCost == null)
        {
            throw new ArgumentException();
        }
        return new Airplane(_flightNumber,
            _destination,
            (DateTime)_departureTime,
            _company,
            (float)_flightCost);
    }
}
```

Рисунок 3 – код класса Builder

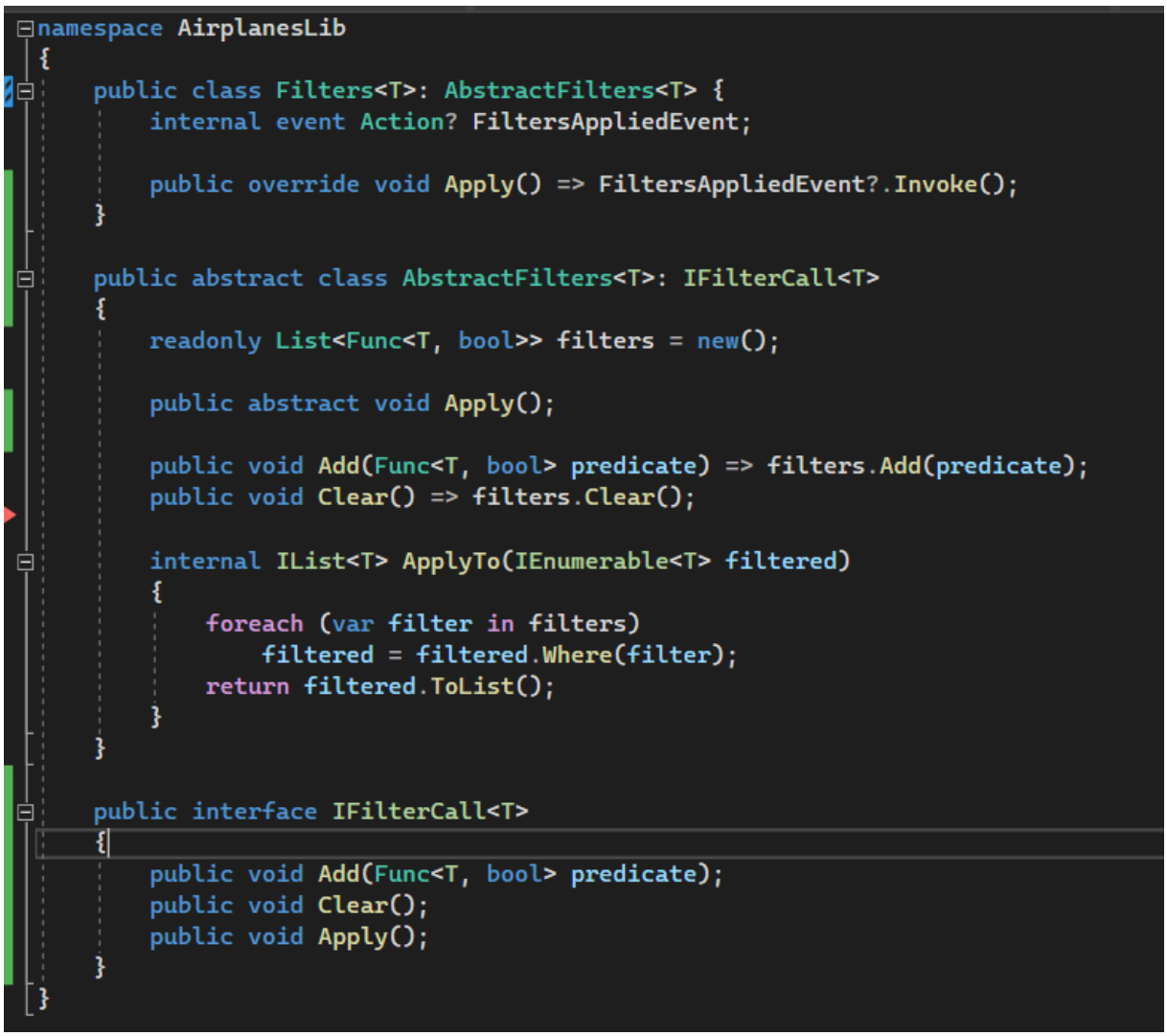
На рис 4 представлен код обобщенного интерфейса ISimpleList<T>. Интерфейс обладает функциональностью простой коллекции, а именно: добавление элемента, удаление конкретного элемента, и получение числа элементов. В интерфейсе объявлено событие, которое должно происходить при изменении коллекции.

```
public interface ISimpleList<T>
{
    public event Action? ListUpdatedEvent;

    public int Count { get; }

    public void Add(T item);
    public void Remove(T item);
}
```

Рисунок 4 – код интерфейса ISimpleList<T>

Для фильтрации элементов некоторой коллекции был спроектирован обобщенный класс Filters<T>. В этом классе объявлено событие FiltersAppliedEvent, вызывающиеся при желании пользователя применить фильтры. Также в классе объявлены методы которые были наследованы от абстрактного класса AbstractFilter<T>:

1) Add – добавление предиката, ответственного за фильтрацию коллекции.
2) Clear – очищение всех фильтров.
3) ApplyTo – применение фильтров к конкретной коллекции, возвращает итератор отфильтрованной коллекции.
4) Apply – метод, который вызывает событие FiltersAppliedEvent.



```
namespace AirplanesLib
{
    public class Filters<T>: AbstractFilters<T> {
        internal event Action? FiltersAppliedEvent;

        public override void Apply() => FiltersAppliedEvent?.Invoke();
    }

    public abstract class AbstractFilters<T>: IFilterCall<T>
    {
        readonly List<Func<T, bool>> filters = new();

        public abstract void Apply();

        public void Add(Func<T, bool> predicate) => filters.Add(predicate);
        public void Clear() => filters.Clear();

        internal IList<T> ApplyTo(IEnumerable<T> filtered)
        {
            foreach (var filter in filters)
                filtered = filtered.Where(filter);
            return filtered.ToList();
        }
    }

    public interface IFilterCall<T>
    {
        public void Add(Func<T, bool> predicate);
        public void Clear();
        public void Apply();
    }
}
```

Рисунок 5 – код класса Filters<T>

Для хранения объектов класса Airplane, предоставления функционала фильтрации и реализации интерфейса ISimpleList<Airplane> был спроектирован класс Airport. Помимо реализации интерфейса ISimpleList<Airplane> ,функционал которого очевиден, в классе представлены свойства:

1) Filters - получение настроек фильтрации
2) FilteredPlanes - возвращение коллекции отфильтрованных элементов.

```
public class Airport : ISimpleList<Airplane>
{
    public event Action? ListUpdatedEvent;

    private List<Airplane> airplanes = new();

    public Airport()
    {
        Filters.FiltersAppliedEvent +=
            () => this.ListUpdatedEvent?.Invoke();
    }

    public Filters<Airplane> Filters { get; init; } = new();

    public IList<Airplane> FilteredPlanes { get => Filters.ApplyTo(airplanes); }

    public int Count => airplanes.Count;

    public void Add(Airplane item)
    {
        airplanes.Add(item);
        ListUpdatedEvent?.Invoke();
    }

    public void Remove(Airplane item)
    {
        airplanes.Remove(item);
        ListUpdatedEvent?.Invoke();
    }
}
```

Рисунок 6 – Программный код класса Airport

ТЕСТИРОВАНИЕ ФУНКЦИОНАЛЬНОСТИ КЛАССОВ

Для тестирования функциональности библиотеки, был спроектирован графический интерфейс с возможностью:

1) Добавления/удаления самолетов из аэропорта;
2) Поиску самолета по регистрационному коду;
3) Фильтрации по месту назначения и дате вылету;
4) Выводу информацию о текущем рейсе самолета.

Для начала протестируем добавление и показ информации о самолете. Для этого создадим самолета и выберем его в левом списке.

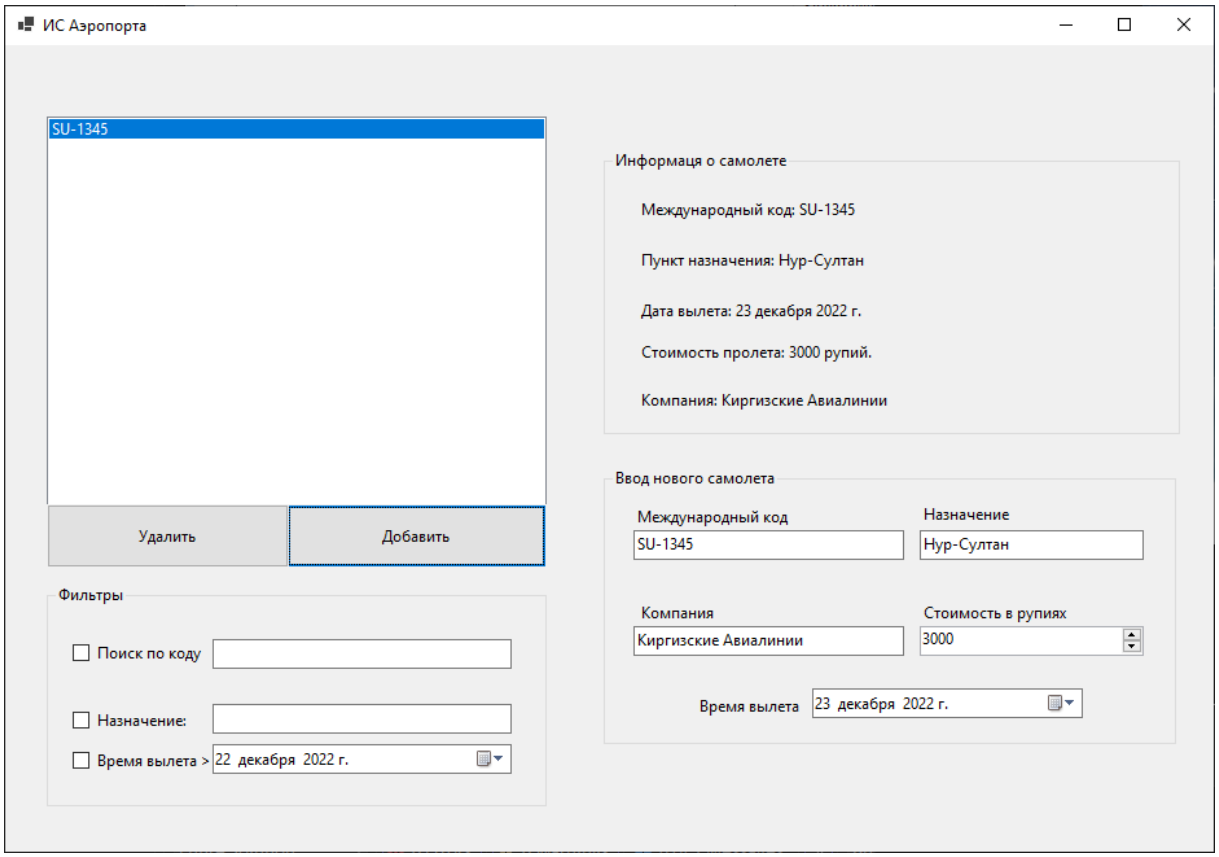


Рисунок 7 – тест добавления самолета

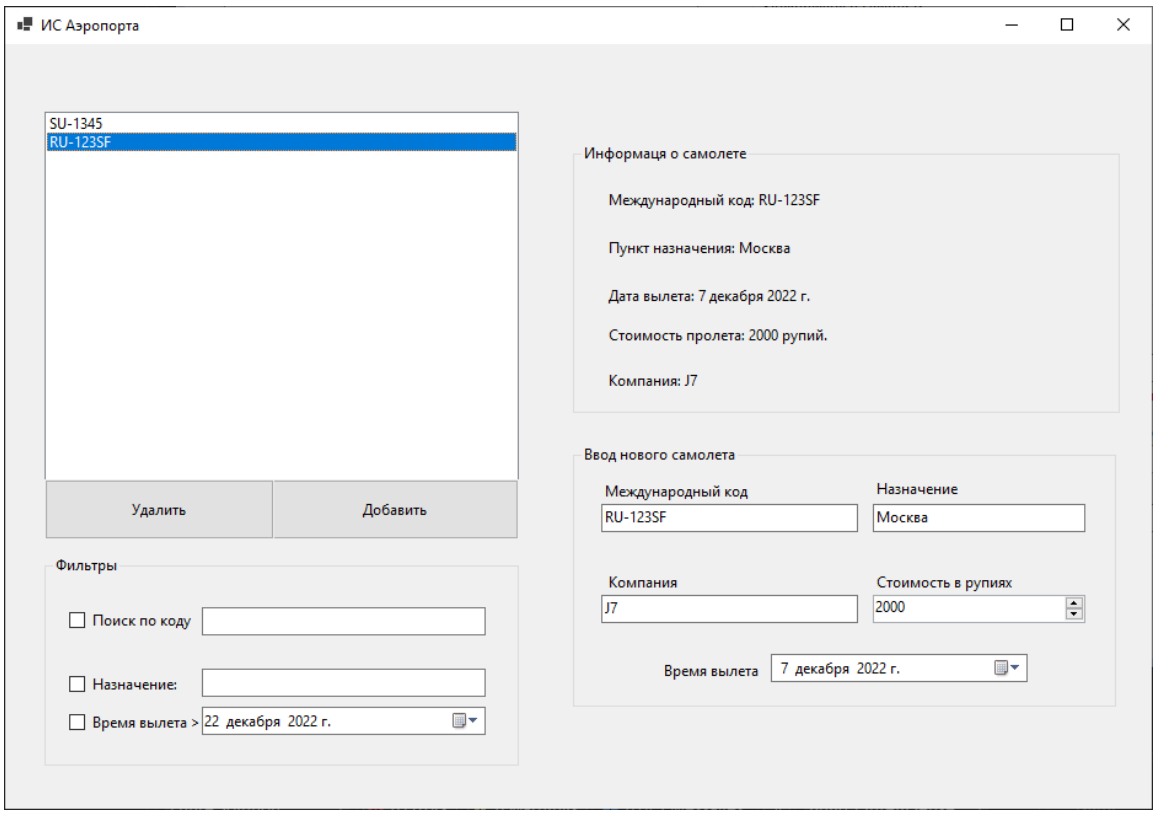


Рисунок 8 – тест показа самолета.

Протестируем фильтрацию по назначению. Для этого в разделе “Фильтры” выберем нужный нами параметр фильтрации.

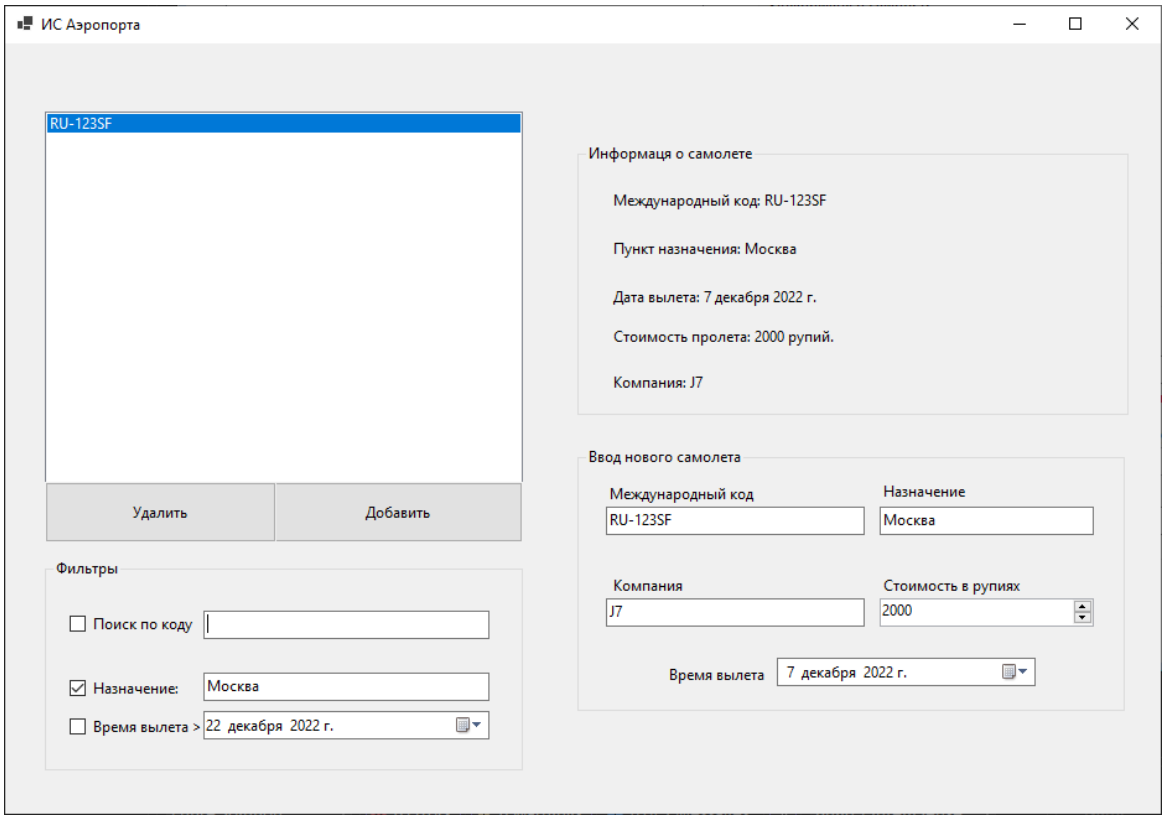


Рисунок 9 – тест фильтрации по назначению

Протестируем фильтрацию по времени вылета.

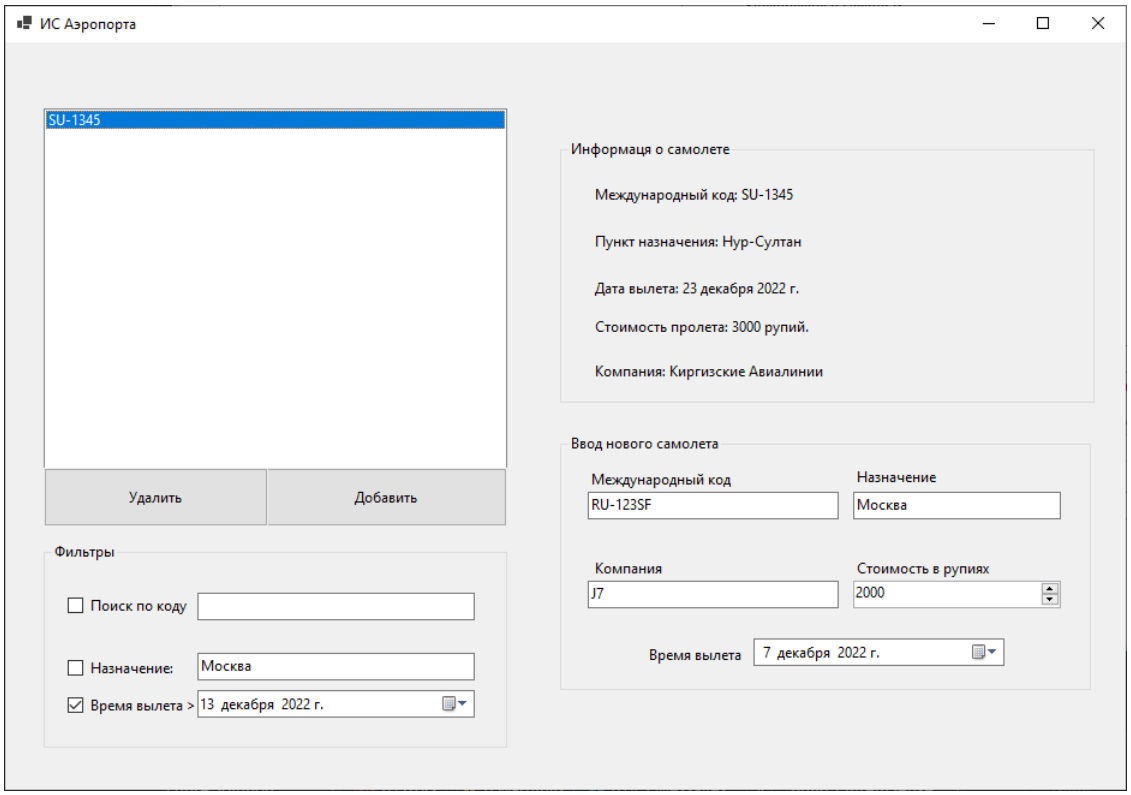


Рисунок 10 – тест фильтрации по времени вылета

Протестируем фильтрацию по назначению и времени вылета.

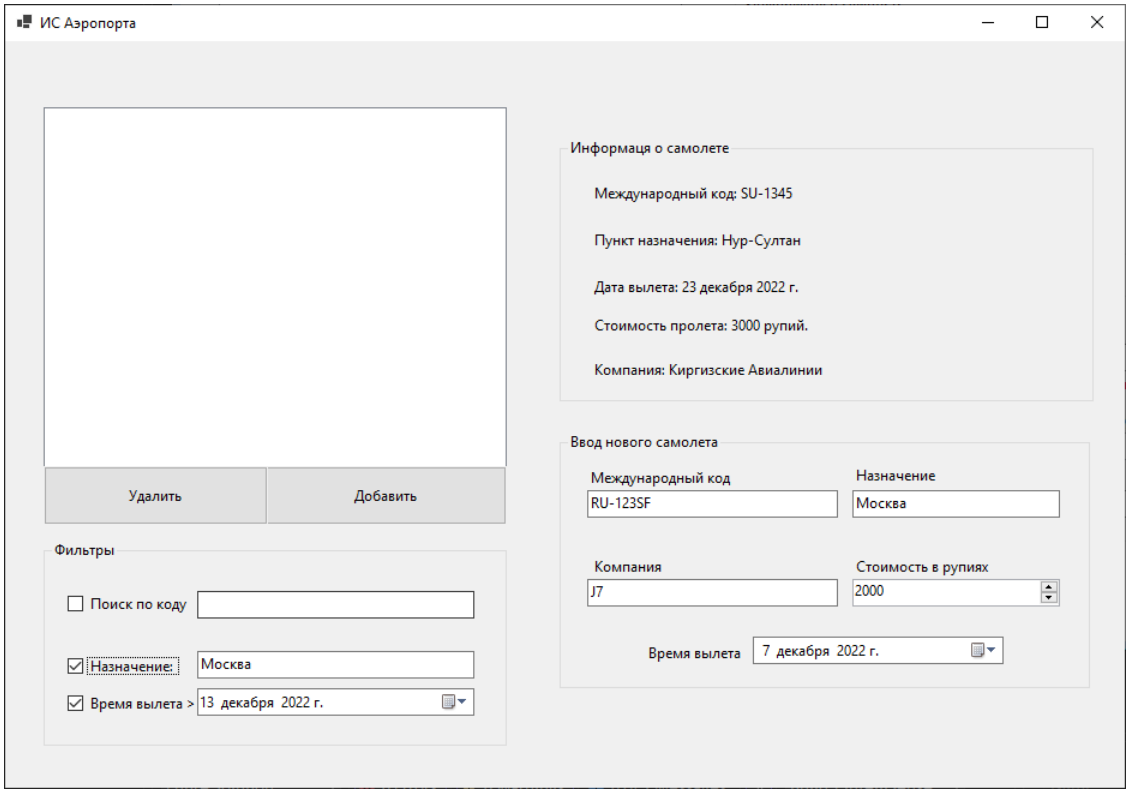


Рисунок 11 – тест фильтрации по назначению и времени вылета

Протестируем поиск по коду.

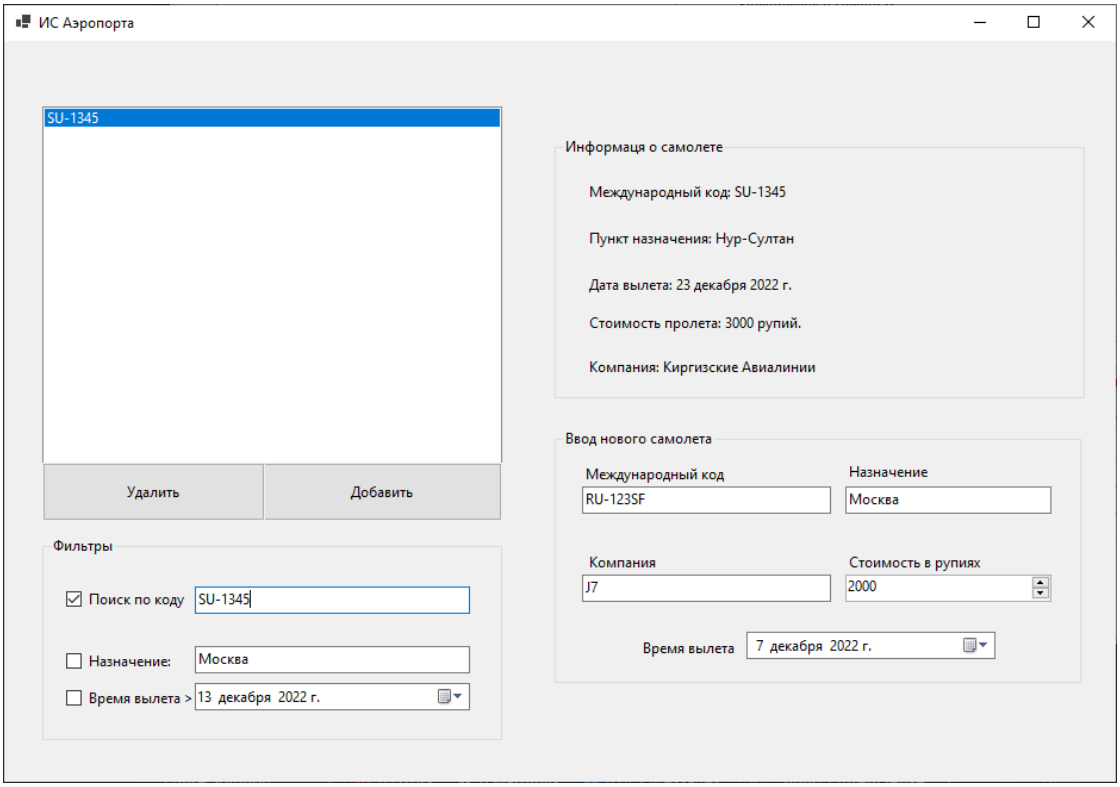


Рисунок 12 – тест фильтрации поиска по коду

Протестируем удаление самолета.

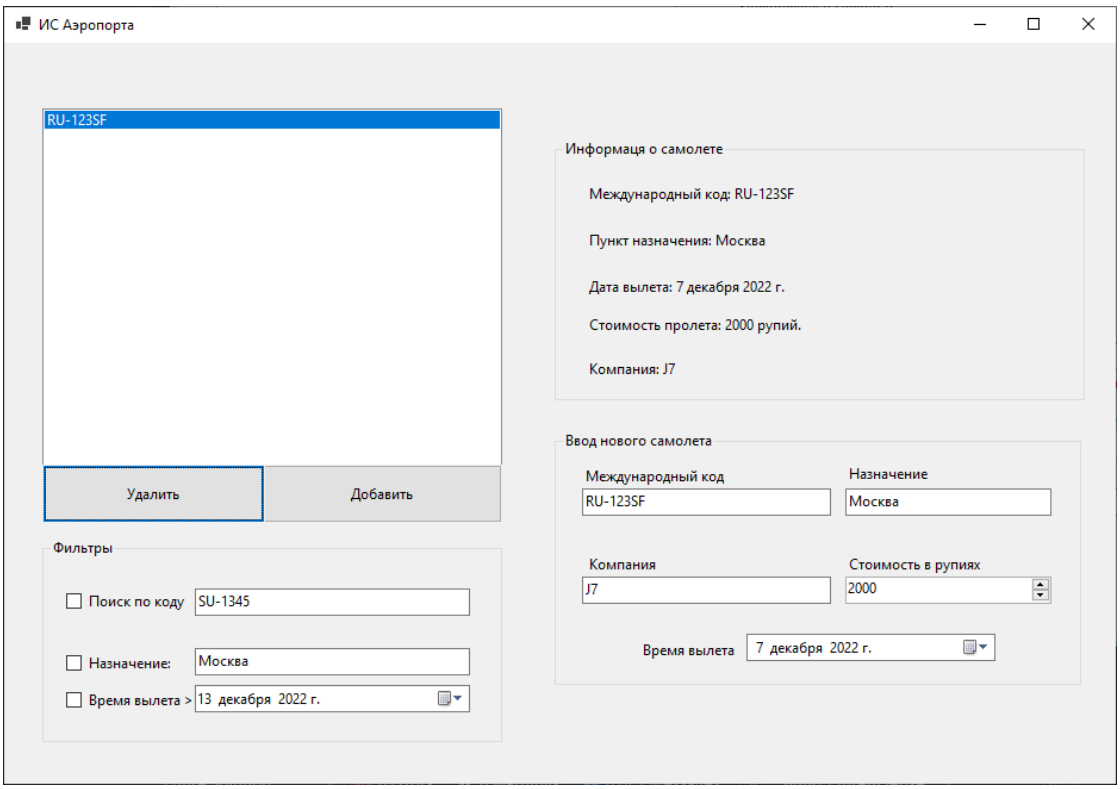


Рисунок 13 – тест фильтрации поиска по коду

ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ В ПРИЛОЖЕНИИ:

Протестируем разработанную библиотеку в разработанном нами интерфейсе. Для начала пользователь вводит некоторое количества самолетов в текущий аэропорт. Что бы это сделать пользователь вводит в окошко ввод нового самолета требуемые данные и нажимает на кнопку “Добавить”. Если ввод был некорректный показывается ошибка.

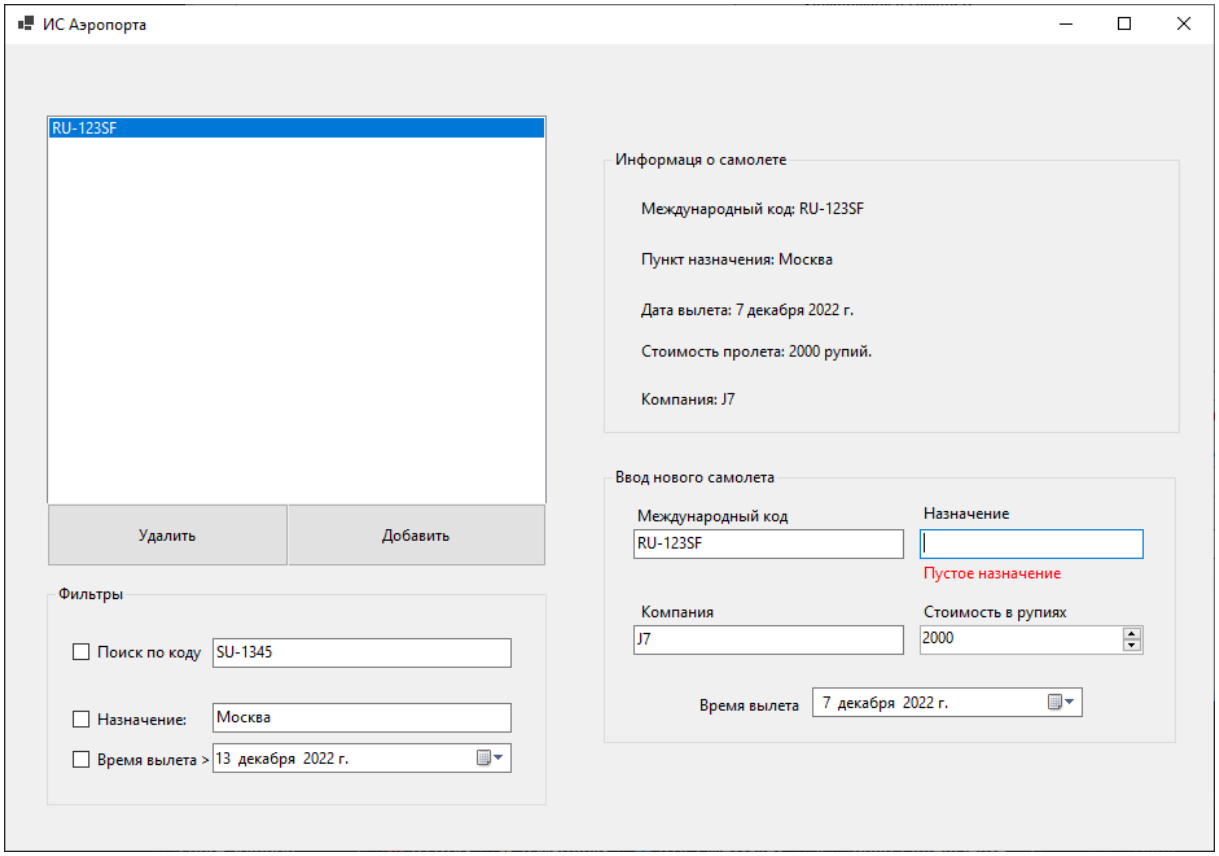


Рисунок 14 – показ ошибки при вводе пользователя

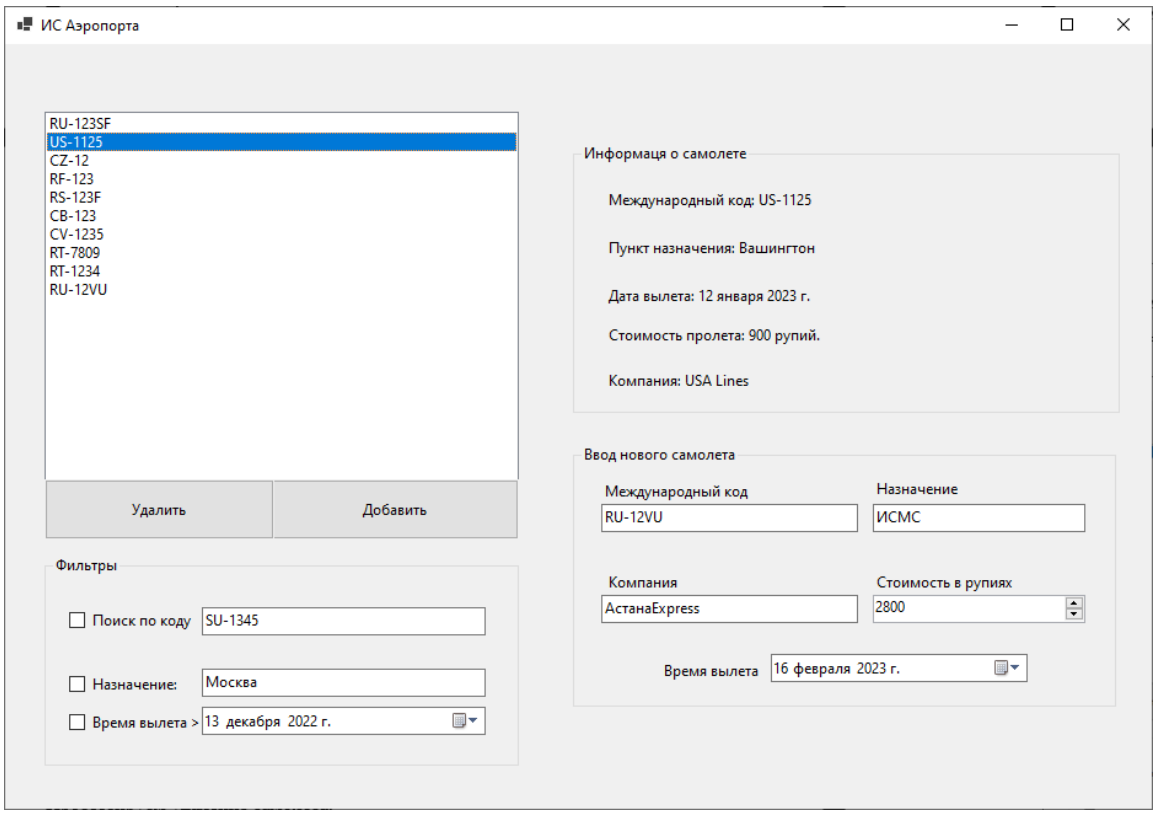


Рисунок 14 – Пользователь добавил рейсы

Затем пользователь может проверить информацию о требуемом самолете просто выбрав в правом списке требуемый рейс. Информация отобразится в окошке “Информация о самолете”.

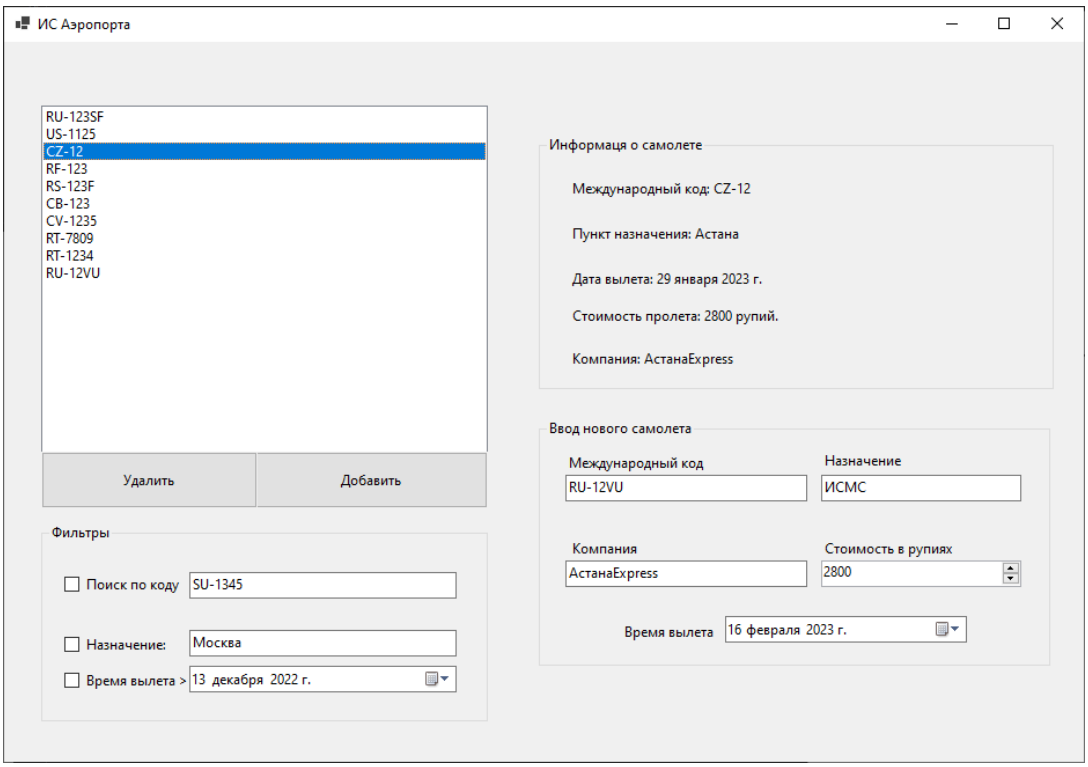


Рисунок 15 – Пользователь выбрал рейс

Пользователь может захотеть отфильтровать текущие рейса. Для этого пользователь в окошко “Фильтры” указывает требуемые параметры фильтрации.

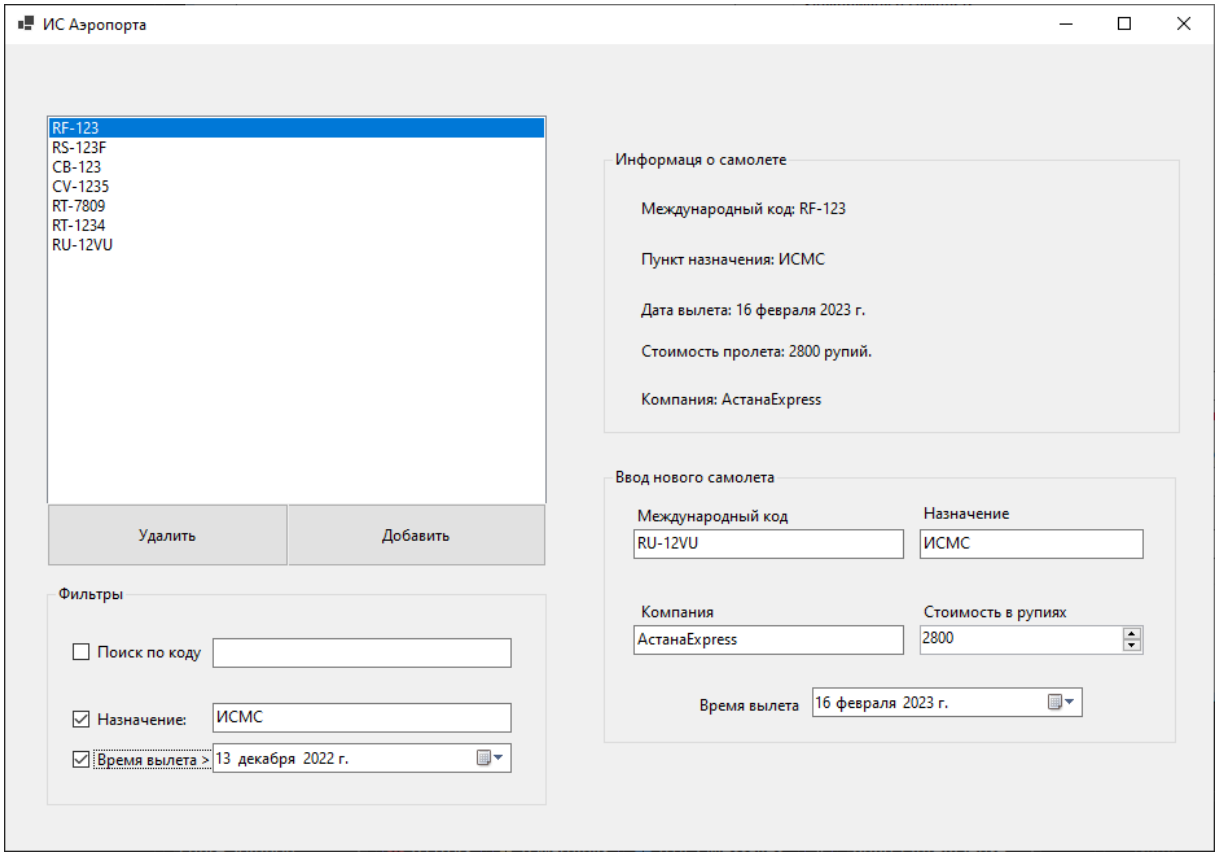


Рисунок 16 – Пользователь использовал фильтрацию по назначению и времени вылета

Также пользователь имеет возможность найти конкретный рейс по его коду. Для этого достаточно в окошке “Фильтры” выбрать “Поиск по коду” и ввести в поле требуемый код.

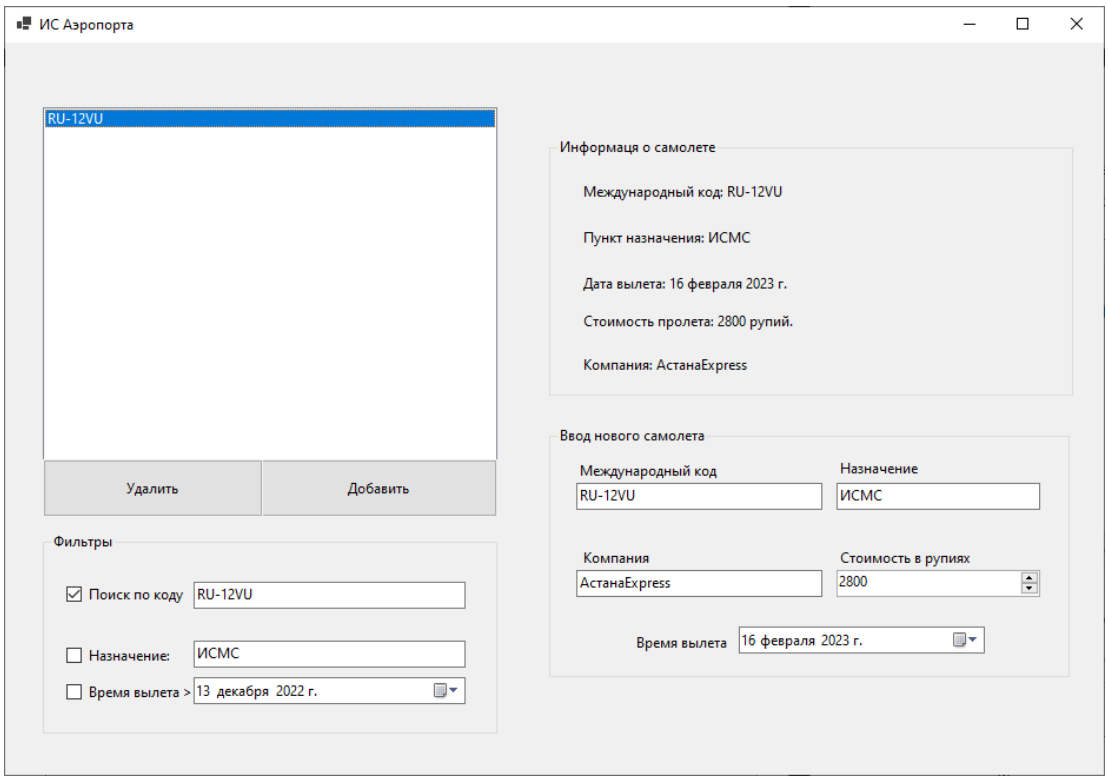


Рисунок 17 – Пользователь использовал поиск по коду

Если же пользователю потребуется удалить конкретный рейс, то ему достаточно выбрать в списке требуемый рейс и нажать кнопку “Удалить”.

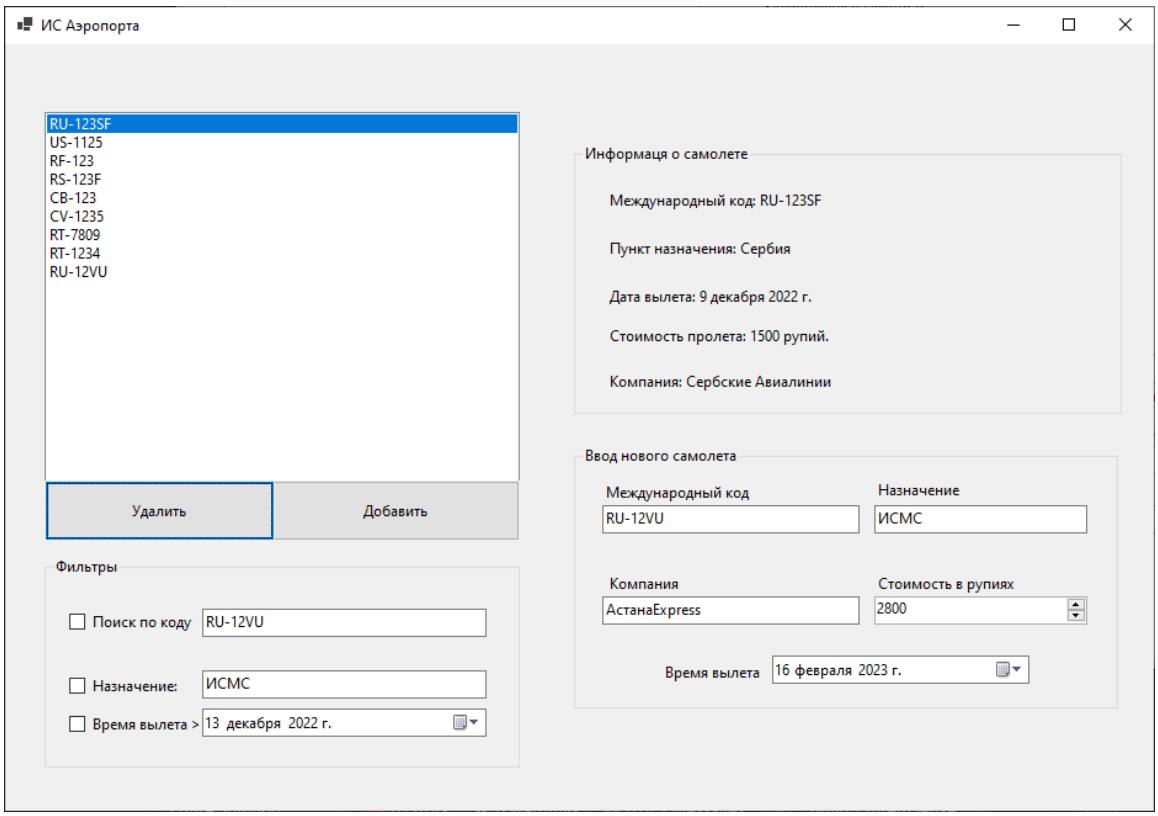


Рисунок 18 – Пользователь удалил рейс из аэропорта

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для работы с библиотекой необходим компилятор C#, встроенный в среду разработки Microsoft Visual Studio. После установки среды программирования, необходимо установить .Net 6.0 SDK. Библиотека не имеет других зависимостей и готова к использованию.

Библиотека классов ориентирована на создание ИС учета рейсов. Основные элементы разработанной библиотеки классов:

1) Класс Airplane. Класс представляет из себя модель данных рейса. Для создания объекта класса Airplane следует использовать Airplane.Builder. В методах установки полей Airplane в Builder могут выбрасываться исключения, с помощью которого пользователь может отслеживать некорректный ввод пользователя.
2) Класс Airport. Класс представляет из себя коллекцию всех рейсов в текущей системе.
    a. Для получения элементов коллекции следует использовать свойство FilteredPlanes.
    b. Для изменения коллекции следует использовать методы интерфейса ISimpleList.
    c. Класс позволяет прослушивать событие изменения коллекции, благодаря чему пользователь с легкостью может синхронизировать класс Airport с интерфейсом.
    d. Если пользователю потребуется фильтрация коллекции, то используя свойство Filters, он сможет задать собственные настройки фильтрации посредством методов класса Filters.

ЗАКЛЮЧЕНИЕ

Код проекта выложен и доступен на удаленном репозитории на GitHub`e: https://github.com/vladcto/SUAI_homework/tree/3_course_project/3_semester/OP/course_project

В результате выполнения курсового проекта была разработана библиотека классов, для учета рейсов. Классы библиотеки содержат все необходимые члены (поля, конструкторы, свойства, методы).

В данной библиотеке реализованы основные принципы объектно-ориентированного программирования: полиморфизм (представлен использованием обобщенных типов), инкапсуляция (представлен методами и свойствами), наследование (представлен реализацией интерфейсов).

Функционал библиотеки успешно протестирован. Все поставленные задачи реализованы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. OMG. Unified Modeling Language (OMG UML) Version 2.5.1 (дата посещения 17.12.22): https://www.omg.org/spec/UML/2.5.1/PDF
2. Microsoft Docs. Interface (справочник по C#) (дата посещения 17.12.22): https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/interface
3. События C# (справочник по C#) (дата посещения 18.12.22): https://metanit.com/sharp/tutorial/3.14.php
4. «C#. Карманный справочник» сост.: Д. Албахари, Б. Албахари, изд. «Вильямс», 2018г.
5. «Программирование на C# для начинаюших» сост.: А.Н. Васильев, изд. «Бомбора», 2022г.
6. «Head First. Изучаем C#» сост.: Э. Стиллмен, Д.Грин, изд. «Питер», 2022г.

ПРИЛОЖЕНИЕ

Код классов:

```
namespace AirplanesLib
{
    public class Airplane
    {
        public string FlightNumber { get; init; }
        public string Destination { get; init; }
        public DateTime DepartuteTime { get; init; }
        public string CompanyName { get; init; }
        public float FlightCost { get; init; }

        private Airplane(string flightNumber, string destination, DateTime
departuteTime, string companyName, float flightCost)
        {
            FlightNumber = flightNumber;
            Destination = destination;
            DepartuteTime = departuteTime;
            CompanyName = companyName;
            FlightCost = flightCost;
        }

        public class Builder
        {
            private string? _flightNumber;
            private string? _destination = null;
            private DateTime? _departureTime = null;
            private string? _company = null;
            private float? _flightCost = null;

            public Builder SetFlightNumber(string inp)
            {
                //Не совпадает формату ИКАО
                if (inp.Trim().Length < 3 || inp.Trim().Length > 8)
                {
                    _flightNumber = null;
                    throw new ArgumentException();
                }
                _flightNumber = inp.Trim();
                return this;
            }

            public Builder SetDestination(string inp)
            {
                if (inp.Trim() == "")
                {
                    _destination = null;
                    throw new ArgumentException();
                }
                _destination = inp.Trim();
                return this;
            }

            public Builder SetDepartureTime(DateTime inp)
            {
                _departureTime = inp;
                return this;
            }

            public Builder SetCompany(string inp)
            {
                _company = inp.Trim();
                return this;
```

```
            }

            public Builder SetFlightCost(float inp)
            {
                if (inp < 0)
                {
                    _flightCost = null;
                    throw new ArgumentException();
                }
                _flightCost = inp;
                return this;
            }

            public Airplane Build()
            {
                if (_destination == null || _flightNumber == null ||
                    _departureTime == null || _company == null || _flightCost ==
null)
                {
                    throw new ArgumentException();
                }
                return new Airplane(_flightNumber,
                    _destination,
                    (DateTime)_departureTime,
                    _company,
                    (float)_flightCost);
            }
        }
    }
}

namespace AirplanesLib
{
    public class Airport : ISimpleList<Airplane>
    {
        public event Action? ListUpdatedEvent;

        private List<Airplane> airplanes = new();

        public Airport()
        {
            Filters.FiltersAppliedEvent +=
                () => this.ListUpdatedEvent?.Invoke();
        }

        public Filters<Airplane> Filters { get; init; } = new();

        public IList<Airplane> FilteredPlanes { get =>
Filters.ApplyTo(airplanes); }

        public int Count => airplanes.Count;

        public void Add(Airplane item)
        {
            airplanes.Add(item);
            ListUpdatedEvent?.Invoke();
        }

        public void Remove(Airplane item)
        {
            airplanes.Remove(item);
            ListUpdatedEvent?.Invoke();
        }
    }
}
```

```
namespace AirplanesLib
{
    public class Filters<T>: AbstractFilters<T> {
        internal event Action? FiltersAppliedEvent;

        public override void Apply() => FiltersAppliedEvent?.Invoke();
    }

    public abstract class AbstractFilters<T>: IFilterCall<T>
    {
        readonly List<Func<T, bool>> filters = new();

        public abstract void Apply();

        public void Add(Func<T, bool> predicate) => filters.Add(predicate);
        public void Clear() => filters.Clear();

        internal IList<T> ApplyTo(IEnumerable<T> filtered)
        {
            foreach (var filter in filters)
                filtered = filtered.Where(filter);
            return filtered.ToList();
        }
    }

    public interface IFilterCall<T>
    {
        public void Add(Func<T, bool> predicate);
        public void Clear();
        public void Apply();
    }
}


namespace AirplanesLib
{
    public interface ISimpleList<T>
    {
        public event Action? ListUpdatedEvent;

        public int Count { get; }

        public void Add(T item);
        public void Remove(T item);
    }
}
```

Код формы:

```
using AirplanesLib;

namespace CourseProject
{
    public partial class MainForm : Form
    {
        Airport airport = new();

        public MainForm()
        {
            InitializeComponent();
            airportListBox.DisplayMember = nameof(Airplane.FlightNumber);
            airportListBox.SelectedIndexChanged +=
```

```
                (sender, e) =>
DisplayAirplane((Airplane)airportListBox.SelectedItem);
            airport.ListUpdatedEvent += () =>
            {
                airportListBox.DataSource = airport.FilteredPlanes;
                airportListBox.Refresh();
            };
        }

        static void ValidateAction(Action validateOperation, Control errorLabel)
        {
            errorLabel.Hide();
            try
            {
                validateOperation();
            }
            catch (ArgumentException) { errorLabel.Show(); };
        }

        private void AddPlane(object sender, EventArgs e)
        {
            Airplane.Builder builder = new();
            ValidateAction(
                () => builder.SetFlightNumber(flightNumberInput.Text),
flightNumberError);
            ValidateAction(
                () => builder.SetDestination(destinationInp.Text),
destinationError);
            builder.SetCompany(companyInput.Text);
            builder.SetFlightCost((int)flightCostInp.Value);
            builder.SetDepartureTime(departureInput.Value);
            try
            {
                airport.Add(builder.Build());
            }
            catch (ArgumentException) { };

        }

        private void RemovePlane(object sender, EventArgs e)
        {
            if (airportListBox.SelectedIndex == -1)
            {
                MessageBox.Show("Нечего удалять.", "Ошибка удаления!",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            airport.Remove((Airplane)airportListBox.SelectedItem);
        }

        private void DisplayAirplane(Airplane airplane)
        {
            companyNameLabel.Text = $"Компания: {airplane.CompanyName}";
            flightNumberLabel.Text = $"Международный код:
{airplane.FlightNumber}";
            flightCostLabel.Text = $"Стоимость пролета: {airplane.FlightCost}
рупий.";
            departureTimeLabel.Text = $"Дата вылета:
{airplane.DepartuteTime.ToLongDateString()}";
            destinationLabel.Text = $"Пункт назначения: {airplane.Destination}";
        }

        private void ChangedFilters(object sender, EventArgs e)
        {
            airport.Filters.Clear();
```

```
            if (flightNumberCheck.Checked)
                airport.Filters.Add((e) => e.FlightNumber ==
flightNumberFilter.Text);
            if (filterDateCheckBox.Checked)
                airport.Filters.Add((e) => timeFilterInput.Value <=
e.DepartuteTime);
            if (filterDestinationCheckBox.Checked)
                airport.Filters.Add((e) => e.Destination ==
destinationFilterInp.Text);
            airport.Filters.Apply();
        }

        private void FlightNumberChanged(object sender, EventArgs e)
        {
            ValidateAction(() =>
                new Airplane.Builder().SetFlightNumber(flightNumberInput.Text),
                flightNumberError);
        }

        private void DestinationChanged(object sender, EventArgs e)
        {
            ValidateAction(() =>
                new Airplane.Builder().SetDestination(destinationInp.Text),
                destinationError);
        }
    }
}
```

Код дизайна формы:

```
namespace CourseProject

{

  partial class MainForm

  {

    /// <summary>

    ///  Required designer variable.

    /// </summary>

    private System.ComponentModel.IContainer components = null;



    /// <summary>

    ///  Clean up any resources being used.
```

```
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        ///  Required method for Designer support - do not modify
        ///  the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.flightCostLabel = new System.Windows.Forms.Label();
            this.companyNameLabel = new System.Windows.Forms.Label();
```

```
this.departureTimeLabel = new System.Windows.Forms.Label();
this.destinationLabel = new System.Windows.Forms.Label();
this.flightNumberLabel = new System.Windows.Forms.Label();
this.groupBox2 = new System.Windows.Forms.GroupBox();
this.flightNumberFilter = new System.Windows.Forms.TextBox();
this.timeFilterInput = new System.Windows.Forms.DateTimePicker();
this.destinationFilterInp = new System.Windows.Forms.TextBox();
this.filterDateCheckBox = new System.Windows.Forms.CheckBox();
this.filterDestinationCheckBox = new System.Windows.Forms.CheckBox();
this.flightNumberCheck = new System.Windows.Forms.CheckBox();
this.groupBox3 = new System.Windows.Forms.GroupBox();
this.label5 = new System.Windows.Forms.Label();
this.label4 = new System.Windows.Forms.Label();
this.label1 = new System.Windows.Forms.Label();
this.label3 = new System.Windows.Forms.Label();
this.label2 = new System.Windows.Forms.Label();
this.departureInput = new System.Windows.Forms.DateTimePicker();
this.flightCostInp = new System.Windows.Forms.NumericUpDown();
this.destinationError = new System.Windows.Forms.Label();
this.destinationInp = new System.Windows.Forms.TextBox();
this.companyInput = new System.Windows.Forms.TextBox();
this.flightNumberError = new System.Windows.Forms.Label();
```

```
this.flightNumberInput = new System.Windows.Forms.TextBox();
this.addBtn = new System.Windows.Forms.Button();
this.removeBtn = new System.Windows.Forms.Button();
this.airportListBox = new System.Windows.Forms.ListBox();
this.groupBox1.SuspendLayout();
this.groupBox2.SuspendLayout();
this.groupBox3.SuspendLayout();
((System.ComponentModel.ISupportInitialize)(this.flightCostInp)).BeginInit();
this.SuspendLayout();
//
// groupBox1
//
this.groupBox1.Controls.Add(this.flightCostLabel);
this.groupBox1.Controls.Add(this.companyNameLabel);
this.groupBox1.Controls.Add(this.departureTimeLabel);
this.groupBox1.Controls.Add(this.destinationLabel);
this.groupBox1.Controls.Add(this.flightNumberLabel);
this.groupBox1.Location = new System.Drawing.Point(462, 81);
this.groupBox1.Name = "groupBox1";
this.groupBox1.Size = new System.Drawing.Size(445, 220);
this.groupBox1.TabIndex = 2;
this.groupBox1.TabStop = false;
```

```
this.groupBox1.Text = "Информаця о самолете";
//
// flightCostLabel
//
this.flightCostLabel.AutoSize = true;
this.flightCostLabel.Location = new System.Drawing.Point(26, 148);
this.flightCostLabel.Name = "flightCostLabel";
this.flightCostLabel.Size = new System.Drawing.Size(12, 15);
this.flightCostLabel.TabIndex = 4;
this.flightCostLabel.Text = "-";
//
// companyNameLabel
//
this.companyNameLabel.AutoSize = true;
this.companyNameLabel.Location = new System.Drawing.Point(26, 185);
this.companyNameLabel.Name = "companyNameLabel";
this.companyNameLabel.Size = new System.Drawing.Size(12, 15);
this.companyNameLabel.TabIndex = 3;
this.companyNameLabel.Text = "-";
//
// departureTimeLabel
//
```

```
this.departureTimeLabel.AutoSize = true;
this.departureTimeLabel.Location = new System.Drawing.Point(26, 116);
this.departureTimeLabel.Name = "departureTimeLabel";
this.departureTimeLabel.Size = new System.Drawing.Size(12, 15);
this.departureTimeLabel.TabIndex = 2;
this.departureTimeLabel.Text = "-";
//
// destinationLabel
//
this.destinationLabel.AutoSize = true;
this.destinationLabel.Location = new System.Drawing.Point(26, 77);
this.destinationLabel.Name = "destinationLabel";
this.destinationLabel.Size = new System.Drawing.Size(12, 15);
this.destinationLabel.TabIndex = 1;
this.destinationLabel.Text = "-";
//
// flightNumberLabel
//
this.flightNumberLabel.AutoSize = true;
this.flightNumberLabel.Location = new System.Drawing.Point(26, 38);
this.flightNumberLabel.Name = "flightNumberLabel";
this.flightNumberLabel.Size = new System.Drawing.Size(12, 15);
```

```
this.flightNumberLabel.TabIndex = 0;
this.flightNumberLabel.Text = "-";
//
// groupBox2
//
this.groupBox2.Controls.Add(this.flightNumberFilter);
this.groupBox2.Controls.Add(this.timeFilterInput);
this.groupBox2.Controls.Add(this.destinationFilterInp);
this.groupBox2.Controls.Add(this.filterDateCheckBox);
this.groupBox2.Controls.Add(this.filterDestinationCheckBox);
this.groupBox2.Controls.Add(this.flightNumberCheck);
this.groupBox2.Location = new System.Drawing.Point(32, 416);
this.groupBox2.Name = "groupBox2";
this.groupBox2.Size = new System.Drawing.Size(386, 172);
this.groupBox2.TabIndex = 3;
this.groupBox2.TabStop = false;
this.groupBox2.Text = "Фильтры";
//
// flightNumberFilter
//
this.flightNumberFilter.Location = new System.Drawing.Point(128, 42);
this.flightNumberFilter.Name = "flightNumberFilter";
```

```
            this.flightNumberFilter.Size = new System.Drawing.Size(231, 23);
            this.flightNumberFilter.TabIndex = 5;
            this.flightNumberFilter.Leave += new
System.EventHandler(this.ChangedFilters);
            //
            // timeFilterInput
            //
            this.timeFilterInput.Location = new System.Drawing.Point(128, 123);
            this.timeFilterInput.Name = "timeFilterInput";
            this.timeFilterInput.Size = new System.Drawing.Size(231, 23);
            this.timeFilterInput.TabIndex = 4;
            this.timeFilterInput.ValueChanged += new
System.EventHandler(this.ChangedFilters);
            this.timeFilterInput.Leave += new System.EventHandler(this.ChangedFilters);
            //
            // destinationFilterInp
            //
            this.destinationFilterInp.Location = new System.Drawing.Point(128, 92);
            this.destinationFilterInp.Name = "destinationFilterInp";
            this.destinationFilterInp.Size = new System.Drawing.Size(231, 23);
            this.destinationFilterInp.TabIndex = 3;
            this.destinationFilterInp.Leave += new
System.EventHandler(this.ChangedFilters);
```

```
            //
            // filterDateCheckBox
            //
            this.filterDateCheckBox.AutoSize = true;
            this.filterDateCheckBox.Location = new System.Drawing.Point(20, 127);
            this.filterDateCheckBox.Name = "filterDateCheckBox";
            this.filterDateCheckBox.Size = new System.Drawing.Size(114, 19);
            this.filterDateCheckBox.TabIndex = 2;
            this.filterDateCheckBox.Text = "Время вылета >";
            this.filterDateCheckBox.UseVisualStyleBackColor = true;
            this.filterDateCheckBox.CheckedChanged += new
System.EventHandler(this.ChangedFilters);
            //
            // filterDestinationCheckBox
            //
            this.filterDestinationCheckBox.AutoSize = true;
            this.filterDestinationCheckBox.Location = new System.Drawing.Point(20, 96);
            this.filterDestinationCheckBox.Name = "filterDestinationCheckBox";
            this.filterDestinationCheckBox.Size = new System.Drawing.Size(98, 19);
            this.filterDestinationCheckBox.TabIndex = 1;
            this.filterDestinationCheckBox.Text = "Назначение: ";
            this.filterDestinationCheckBox.UseVisualStyleBackColor = true;
```

```
            this.filterDestinationCheckBox.CheckedChanged += new
System.EventHandler(this.ChangedFilters);
            //
            // flightNumberCheck
            //
            this.flightNumberCheck.AutoSize = true;
            this.flightNumberCheck.Location = new System.Drawing.Point(20, 44);
            this.flightNumberCheck.Name = "flightNumberCheck";
            this.flightNumberCheck.Size = new System.Drawing.Size(106, 19);
            this.flightNumberCheck.TabIndex = 0;
            this.flightNumberCheck.Text = "Поиск по коду";
            this.flightNumberCheck.UseVisualStyleBackColor = true;
            //
            // groupBox3
            //
            this.groupBox3.Controls.Add(this.label5);
            this.groupBox3.Controls.Add(this.label4);
            this.groupBox3.Controls.Add(this.label1);
            this.groupBox3.Controls.Add(this.label3);
            this.groupBox3.Controls.Add(this.label2);
            this.groupBox3.Controls.Add(this.departureInput);
            this.groupBox3.Controls.Add(this.flightCostInp);
            this.groupBox3.Controls.Add(this.destinationError);
```

```
this.groupBox3.Controls.Add(this.destinationInp);
this.groupBox3.Controls.Add(this.companyInput);
this.groupBox3.Controls.Add(this.flightNumberError);
this.groupBox3.Controls.Add(this.flightNumberInput);
this.groupBox3.Location = new System.Drawing.Point(462, 326);
this.groupBox3.Name = "groupBox3";
this.groupBox3.Size = new System.Drawing.Size(442, 214);
this.groupBox3.TabIndex = 4;
this.groupBox3.TabStop = false;
this.groupBox3.Text = "Ввод нового самолета";
//
// label5
//
this.label5.AutoSize = true;
this.label5.Location = new System.Drawing.Point(244, 28);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(73, 15);
this.label5.TabIndex = 15;
this.label5.Text = "Назначение";
//
// label4
//
```

```
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(23, 30);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(123, 15);
this.label4.TabIndex = 14;
this.label4.Text = "Международный код";
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(26, 104);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(63, 15);
this.label1.TabIndex = 13;
this.label1.Text = "Компания";
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(71, 176);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(84, 15);
```

```
this.label3.TabIndex = 12;
this.label3.Text = "Время вылета";
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(244, 104);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(118, 15);
this.label2.TabIndex = 11;
this.label2.Text = "Стоимость в рупиях";
//
// departureInput
//
this.departureInput.Location = new System.Drawing.Point(161, 170);
this.departureInput.Name = "departureInput";
this.departureInput.Size = new System.Drawing.Size(209, 23);
this.departureInput.TabIndex = 10;
//
// flightCostInp
//
this.flightCostInp.Increment = new decimal(new int[] {
```

```
100,
0,
0,
0});
this.flightCostInp.Location = new System.Drawing.Point(244, 122);
this.flightCostInp.Maximum = new decimal(new int[] {
100000,
0,
0,
0});
this.flightCostInp.Name = "flightCostInp";
this.flightCostInp.Size = new System.Drawing.Size(173, 23);
this.flightCostInp.TabIndex = 8;
//
// destinationError
//
this.destinationError.AutoSize = true;
this.destinationError.ForeColor = System.Drawing.Color.Red;
this.destinationError.Location = new System.Drawing.Point(244, 74);
this.destinationError.Name = "destinationError";
this.destinationError.Size = new System.Drawing.Size(113, 15);
this.destinationError.TabIndex = 7;
```

```
            this.destinationError.Text = "Пустое назначение";
            this.destinationError.Visible = false;
            //
            // destinationInp
            //
            this.destinationInp.Location = new System.Drawing.Point(244, 48);
            this.destinationInp.Name = "destinationInp";
            this.destinationInp.PlaceholderText = "Назначение";
            this.destinationInp.Size = new System.Drawing.Size(173, 23);
            this.destinationInp.TabIndex = 6;
            this.destinationInp.TextChanged += new
System.EventHandler(this.DestinationChanged);
            //
            // companyInput
            //
            this.companyInput.Location = new System.Drawing.Point(23, 122);
            this.companyInput.Name = "companyInput";
            this.companyInput.PlaceholderText = "Компания";
            this.companyInput.Size = new System.Drawing.Size(209, 23);
            this.companyInput.TabIndex = 3;
            //
            // flightNumberError
            //
```

```
            this.flightNumberError.AutoSize = true;
            this.flightNumberError.ForeColor = System.Drawing.Color.Red;
            this.flightNumberError.Location = new System.Drawing.Point(26, 74);
            this.flightNumberError.Name = "flightNumberError";
            this.flightNumberError.Size = new System.Drawing.Size(138, 15);
            this.flightNumberError.TabIndex = 2;
            this.flightNumberError.Text = "Неверный формат кода";
            this.flightNumberError.Visible = false;
            //
            // flightNumberInput
            //
            this.flightNumberInput.Location = new System.Drawing.Point(23, 48);
            this.flightNumberInput.Name = "flightNumberInput";
            this.flightNumberInput.PlaceholderText = "Международный код";
            this.flightNumberInput.Size = new System.Drawing.Size(209, 23);
            this.flightNumberInput.TabIndex = 1;
            this.flightNumberInput.TextChanged += new
System.EventHandler(this.FlightNumberChanged);
            //
            // addBtn
            //
            this.addBtn.Location = new System.Drawing.Point(217, 354);
            this.addBtn.Name = "addBtn";
```

```
this.addBtn.Size = new System.Drawing.Size(201, 49);
this.addBtn.TabIndex = 5;
this.addBtn.Text = "Добавить";
this.addBtn.UseVisualStyleBackColor = true;
this.addBtn.Click += new System.EventHandler(this.AddPlane);
//
// removeBtn
//
this.removeBtn.Location = new System.Drawing.Point(32, 354);
this.removeBtn.Name = "removeBtn";
this.removeBtn.Size = new System.Drawing.Size(187, 49);
this.removeBtn.TabIndex = 6;
this.removeBtn.Text = "Удалить";
this.removeBtn.UseVisualStyleBackColor = true;
this.removeBtn.Click += new System.EventHandler(this.RemovePlane);
//
// airportListBox
//
this.airportListBox.FormattingEnabled = true;
this.airportListBox.ItemHeight = 15;
this.airportListBox.Location = new System.Drawing.Point(32, 55);
this.airportListBox.Name = "airportListBox";
```

```
this.airportListBox.Size = new System.Drawing.Size(386, 304);
this.airportListBox.TabIndex = 7;
//
// MainForm
//
this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 15F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(933, 622);
this.Controls.Add(this.removeBtn);
this.Controls.Add(this.groupBox3);
this.Controls.Add(this.groupBox2);
this.Controls.Add(this.groupBox1);
this.Controls.Add(this.addBtn);
this.Controls.Add(this.airportListBox);
this.Name = "MainForm";
this.Text = "ИС Аэропорта";
this.groupBox1.ResumeLayout(false);
this.groupBox1.PerformLayout();
this.groupBox2.ResumeLayout(false);
this.groupBox2.PerformLayout();
this.groupBox3.ResumeLayout(false);
this.groupBox3.PerformLayout();
```

```
    ((System.ComponentModel.ISupportInitialize)(this.flightCostInp)).EndInit();
    this.ResumeLayout(false);

}

#endregion
private GroupBox groupBox1;
private Label flightCostLabel;
private Label companyNameLabel;
private Label departureTimeLabel;
private Label destinationLabel;
private Label flightNumberLabel;
private GroupBox groupBox2;
private CheckBox filterDateCheckBox;
private CheckBox filterDestinationCheckBox;
private CheckBox flightNumberCheck;
private GroupBox groupBox3;
private TextBox companyInput;
private Label flightNumberError;
private TextBox flightNumberInput;
private DateTimePicker timeFilterInput;
private TextBox destinationFilterInp;
```

```
        private DateTimePicker departureInput;
        private NumericUpDown flightCostInp;
        private Label destinationError;
        private TextBox destinationInp;
        private Button addBtn;
        private Button removeBtn;
        private TextBox flightNumberFilter;
        private ListBox airportListBox;
        private Label label2;
        private Label label3;
        private Label label1;
        private Label label5;
        private Label label4;
    }
}
```