МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННО-СЕТЕВЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| кандидат технических наук, доцент |  |  |  | Н.В. Богословская |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ (ПРОЕКТУ) |
| ДОБАВЛЕНИЕ САМОЛЕТОВ В БАЗУ АЭРОПОРТА |
| по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № |  |  |  |  | Левин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

1. **Описание целей и постановка задачи**

Требуется разработать библиотеку классов для возможности добавления самолетов в базу аэропорта. Объекты классов должны хранить следующую информацию:

* Название самолета
* Номер самолета
* Дату выпуска
* Количество посадочных мест
* Объект мотор
* Объект шасси
* Объект фабрика
* Объект маршрут
* Количество колес
* Размах крыльев
* Тип топлива
* Уровень объема топлива

Объекты классов должны иметь методы для:

* Постройки самолета
* Установки и демонтажа агрегатов
* Заправки

Библиотека может использоваться для добавления самолетов в базу аэропорта, и графического отображения данных о самолетах.

В классах необходимо обеспечить:

* Хранение условно-постоянной информации статическими полями или константами.
* Все необходимые конструкторы с параметрами и конструкторы по умолчанию.

Библиотека будет использована для оформления интерфейсов пользователя информационных систем.

1. Диаграмма классов

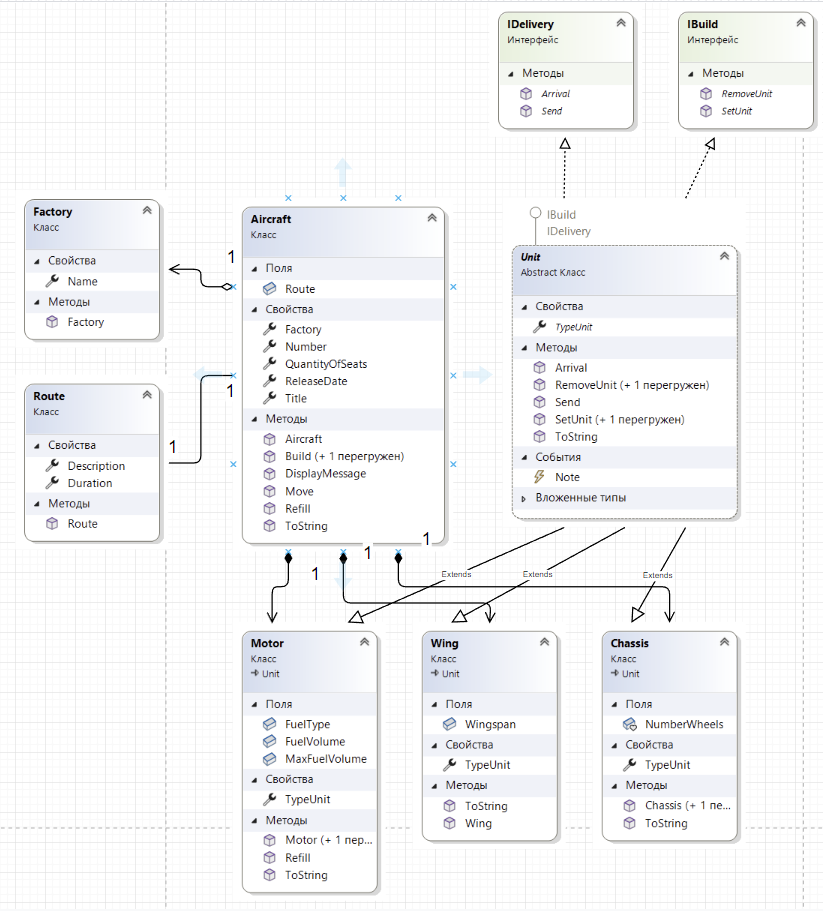


Рисунок 1. Диаграмма классов

1. Описание программного кода классов

* Класс самолет
  + Конструктор служит для определения некоторых свойств класса параметрами конструктора, а также для инициализации мотора, крыла и шасси.
  + Метод ToString скрывает базовый метод, возвращает строку с информацией о самолете
  + Метод Build по очереди вызывает методы установок детали для каждой детали
  + Метод Build с параметром типа даты, вызывает методы установок деталей и передает в каждый метод параметр дату.
  + Метод заправки принимает объем топлива и типа топлива и вызывает одноименный метод мотора с передачей туда этих параметров.
* Класс Unit
  + SetUnit – метод для установки детали, вызывает событие, реализует метод из интерфейса IBuild.
  + SetUnit(date) – перегруженный метод, для случая, если нам нужно вывести дату установки детали
  + RemoveUnit - метод для демонтажа детали, вызывает событие, реализует метод из интерфейса IBuild.
  + RemoveUnit(date) – перегруженный метод, для случая, если нам нужно вывести дату установки детали
  + Методы Send и Arrival методы для отправки и прибытия детали соответственно, реализуют методы из интерфейса IDelivery.
* Класс Motor
  + Переопределенный метод ToString возвращает информацию о моторе.
  + Метод Refill позволяет заправлять баки для мотора, сделана проверка на типа топлива и на превышение объема бака.
* Класс Wing
  + Переопределенный метод ToString возвращает информацию о крыле.
* Класс Chassis
  + Переопределенный метод ToString возвращает информацию о шасси.
* Класс Factory
  + Хранит в себе название фабрики
* Класс AddAircraft
  + В нем хранится массив фабрик, инициализирующийся при загрузке формы. По клику на кнопку «добавить» мы можем добавить самолет и закрыть форму, можем нажать отмена и самолет не добавится. По клику на кнопку «создать и собрать самолет» мы может создать самолет и собрать самолет с помощью метода и конструктора самолета. По клику на кнопку «добавить маршрут» мы можем добавить маршрут к нашему самолету.
* Класс Form1
  + Основная форма в которой находится список добавленных самолетов, текстовое поле для вывода информации о самолете, кнопка «добавить самолет» для добавления самолета и открытия дополнительной формы, и кнопка «заправить на» для заправки на объем равный значению числового поля, находящегося рядом с кнопкой.

1. Тестирование функциональности классов
2. Пример использования библиотеки для построения интерфейса прикладного решения
3. Руководство пользователя-программиста при использовании созданной библиотеки как ресурса
4. Назначение и условия применения

Разработанная библиотека классов предназначена для создания самолетов и добавления их в базу аэропорта и может быть внедрена в информационные системы. В функционал библиотеки входит сборка самолета, заправка самолета, добавление маршрута самолета. Для работы с библиотекой необходим компилятор Microsoft Visual Studio 2019 c платформой .Net Framework.

## Характеристика

## Временные характеристики

Время готовности библиотеки классов – время загрузки проекта библиотеки в среде разработки Microsoft Visual Studio.

## Обращение к библиотеке

Процедуры загрузки и вызова разработанной библиотеки классов обеспечивают автоматическую загрузку формы для ввода данных пользователем. Какие-либо действия оператора в ходе загрузки, вызова и запуска библиотеки не предусмотрены.

## Входные и выходные данные

## Входная информация – входные данные, введенные пользователем на форме в текстовые поля. Кодирование данных не применяется.

## Выходная информация – выходные данные, выводимые в специальное текстовое поле на форме. Применяется переопределенный метод ToString для вывода обработанных данных.

1. Заключение

Была разработана библиотека классов для редактирования сборки самолетов и добавления их в базу аэропорта. Классы библиотеки содержат все необходимые члены (поля, конструкторы, свойства, методы).

Библиотека включает в себя класс самолета, детали самолета. Также библиотека содержит два интерфейса, реализуемый в одном классе. Методы классов реализуют необходимые функции библиотеки: вывод информации в текстовое поле, запись и чтение данных.

Функционал библиотеки успешно протестирован. Все поставленные задачи реализованы.

1. Библиографический список
2. Microsoft: Метод Object.ToString, URL: <https://docs.microsoft.com/ru-ru/dotnet/api/system.object.tostring?view=netframework-4.8>
3. Metanit: Полное руководство по языку программирования С# 9.0 и платформе .NET 5, URL: <https://metanit.com/sharp/tutorial/>
4. Metanit: Отношения между классами и объектами, URL: <https://metanit.com/sharp/patterns/1.2.php>
5. Приложения

## Приложение 1 – Код класса Aircraft

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Library

{

public class Aircraft

{

public string Title { get; set; }

//композиция

public Motor Motor { get; set; }

public Wing Wing { get; set; }

public Chassis Chassis { get; set; }

//ассоциация, у самолета может и не быть аэропорта, а может быть

public Route Route;

//агрегация

public Factory Factory { get; set; }

public int Number { get; set; }

public byte QuantityOfSeats { get; set; }

public DateTime ReleaseDate { get; set; }

public Aircraft(Factory factory, string fuelType, decimal wingSpan, string title, int number, byte quantityOfSeats, DateTime releaseDate)

{

Title = title;

//композиция, детали создаются вместе с самолетом

Motor = new Motor(fuelType, 100, 1000);

Wing = new Wing(wingSpan);

Chassis = new Chassis(6);

//агрегация, потому что фабрика это внешний объект, который мы указываем, они равноправны

Factory = factory;

Number = number;

QuantityOfSeats = quantityOfSeats;

ReleaseDate = releaseDate;

Motor.Note += DisplayMessage;

Wing.Note += DisplayMessage;

Chassis.Note += DisplayMessage;

}

public void DisplayMessage(string message)

{

MessageBox.Show(message);

}

public new string ToString()

{

return String.Format(

"Самолет с номером {0}.\n" +

"Название самолета: {1} \n" +

"Количество посадочных мест: {2} \n" +

"Дата выпуска: {3} \n" +

"Мотор: {4} \n" +

"Шасси: {5} \n" +

"Крыло: {6} \n",

Number,

Title,

QuantityOfSeats,

ReleaseDate.ToShortDateString(),

Motor.ToString(),

Chassis.ToString(),

Wing.ToString()

);

}

//перегруженные методы

public void Build()

{

Motor.SetUnit();

Wing.SetUnit();

Chassis.SetUnit();

}

public void Build(DateTime date)

{

Motor.SetUnit(date);

Wing.SetUnit(date);

Chassis.SetUnit(date);

}

public void Refill(decimal volume, string fuelType)

{

Motor.Refill(volume, fuelType);

}

}

}

## Приложение 2 – Код класса Unit

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Library

{

public abstract class Unit : IBuild, IDelivery

{

public abstract string TypeUnit { get;}

public delegate void UnitHandler(string message);

public event UnitHandler Note;

//метод для установки детали

public void SetUnit()

{

Note?.Invoke("Set " + TypeUnit);

}

public void RemoveUnit()

{

Note?.Invoke("Remove " + TypeUnit);

}

//перегруженный метод для установки детали

public void SetUnit(DateTime dateTime)

{

Note?.Invoke(String.Format("Set {0}, Date: {1}", TypeUnit, dateTime.ToShortDateString()));

}

public void RemoveUnit(DateTime dateTime)

{

Note?.Invoke(String.Format("Remove {0}, Date: {1}", TypeUnit, dateTime.ToShortDateString()));

}

//метод для отправки детали

public void Send()

{

Note?.Invoke("Send " + TypeUnit);

}

//метод для прибытия детали

public void Arrival()

{

Note?.Invoke("Arrival " + TypeUnit);

}

public override string ToString()

{

return TypeUnit;

}

}

}

## Приложение 3 – Код класса Motor

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Library

{

public class Motor : Unit

{

public string FuelType;

public decimal FuelVolume;

public decimal MaxFuelVolume;

//переопределение

public override string TypeUnit

{

get => "Мотор";

}

public Motor(string fuel, decimal fuelVolume, decimal maxFuelVolume)

{

FuelType = fuel;

FuelVolume = fuelVolume;

MaxFuelVolume = maxFuelVolume;

}

public Motor()

{

}

//переопредление метода

public override string ToString()

{

return

String.Format(

"Тип топлива: {0}\n" +

"Макс. объем топлива: {1}\n" +

"Объем топлива: {2}\n",

FuelType,

MaxFuelVolume,

FuelVolume

);

}

public void Refill(decimal volume, string fuelType)

{

//если тип топлива совпадает

if (FuelType == fuelType)

{

//прибавляем

FuelVolume += volume;

if (FuelVolume > MaxFuelVolume)

{

//если заполнили

FuelVolume = MaxFuelVolume;

}

}

}

}

}

## Приложение 4 – Код класса Chassis

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Library

{

public class Chassis : Unit

{

internal int NumberWheels;

public override string TypeUnit

{

get => "Шасси";

}

public Chassis()

{

}

public Chassis(int numberWheels)

{

NumberWheels = numberWheels;

}

public override string ToString()

{

return

String.Format(

"Количество колес: {0}\n",

NumberWheels

);

}

}

}

## Приложение 5 – Код класса Wing

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms.VisualStyles;

namespace Library

{

public class Wing : Unit

{

public override string TypeUnit

{

get => "Крыло";

}

public decimal Wingspan;

public Wing(decimal wingspan)

{

Wingspan = wingspan;

}

public override string ToString()

{

return

String.Format(

"Размах крыла: {0}\n",

Wingspan

);

}

}

}

## Приложение 6 – Код интерфейса IBuild

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Library

{

internal interface IBuild

{

//метод для установки детали

void SetUnit();

//метод для снятия детали

void RemoveUnit();

}

}

## Приложение 7 – Код интерфейса IDelivery

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Library

{

internal interface IDelivery

{

void Send();

void Arrival();

}

}

## Приложение 8 – Код класса Factory

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Library

{

public class Factory

{

public Factory(string name)

{

Name = name;

}

public string Name { get; set; }

}

}

## Приложение 9 – Код класса AddAircraft

using Library;

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 kurs

{

public partial class AddAircraft : Form

{

private BindingList<Factory> factories;

public Aircraft aircraft;

public AddAircraft()

{

InitializeComponent();

}

private void button6\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedIndex > -1)

{

aircraft = new Aircraft(

factories[listBox1.SelectedIndex],

textBox1.Text,

numericUpDown4.Value,

textBox5.Text,

Decimal.ToInt32(numericUpDown1.Value),

Decimal.ToByte(numericUpDown3.Value),

dateTimePicker1.Value

);

if (checkBox1.Checked)

{

aircraft.Build(DateTime.Today);

}

else

aircraft.Build();

}

else

{

MessageBox.Show("Добавьте фабрику");

}

}

private void button7\_Click(object sender, EventArgs e)

{

if (aircraft != null)

{

aircraft.Route = new Route(numericUpDown2.Value, richTextBox1.Text);

}

else

{

MessageBox.Show("Создайте самолет");

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (aircraft != null)

{

DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Создайте самолет");

}

}

private void Form2\_Load(object sender, EventArgs e)

{

factories = new BindingList<Factory>();

factories.Add(new Factory("Фабрика 1"));

factories.Add(new Factory("Фабрика 2"));

listBox1.DataSource = factories;

listBox1.DisplayMember = "Name";

}

private void button1\_Click(object sender, EventArgs e)

{

DialogResult = DialogResult.Cancel;

this.Close();

}

}

}

## Приложение 10 – Код класса Form1

using Library;

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 kurs

{

public partial class Form1 : Form

{

private BindingList<Aircraft> aircrafts = new BindingList<Aircraft>();

public Form1()

{

InitializeComponent();

}

private void button3\_Click(object sender, EventArgs e)

{

AddAircraft form = new AddAircraft();

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

{

aircrafts.Add(form.aircraft);

richTextBox2.Text = aircrafts[listBox1.SelectedIndex].ToString();

}

}

private void Form1\_Load(object sender, EventArgs e)

{

listBox1.DataSource = aircrafts;

listBox1.DisplayMember = "Number";

}

private void button2\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedIndex > -1)

{

aircrafts[listBox1.SelectedIndex].Refill(100, "Керосин");

richTextBox2.Text = aircrafts[listBox1.SelectedIndex].ToString();

}

}

}

}

## 