**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА**

**з дисципліни**

**«Об’єктно-орієнтоване програмування»**

**спеціальність 123 «Комп’ютерна інженерія»**

**Студента групи 201-ТК Любарця Владислава**

Керівник роботи

доктор технічних наук, професор Демиденко М.І.

Полтава – 2024

# Зміст

[Вступ 3](#_Toc168874480)

[РОЗДІЛ 1 6](#_Toc168874481)

[ПОСТАНОВКА ЗАДАЧІ 6](#_Toc168874482)

[1.1 Мова програмування С# 6](#_Toc168874483)

[1.2 Призначення розкладів аеропорту та їх функціонал 7](#_Toc168874484)

[Розділ 2 9](#_Toc168874485)

[Опис функціоналу класів 9](#_Toc168874486)

[2.1 Класи проекту Windows Forms 9](#_Toc168874487)

[Розділ 3 11](#_Toc168874488)

[Дизайн інтерфейсу 11](#_Toc168874489)

[3.1 Головне вікно програми 11](#_Toc168874490)

[3.2 Додавання даних рейсу 11](#_Toc168874491)

[3.3 Головне вікно після додавання рейсу 12](#_Toc168874492)

[3.4 Видалення рейсу 13](#_Toc168874493)

[Розділ 4 14](#_Toc168874494)

[Вихідний код проекту 14](#_Toc168874495)

[4.1 Клас “Flight” 14](#_Toc168874496)

[4.2 Клас Form1Design 15](#_Toc168874497)

[4.3 Клас Form1 16](#_Toc168874498)

[Висновок 20](#_Toc168874499)

[Список використаних джерел 22](#_Toc168874500)

# ВСТУП

Об'єктно-орієнтоване програмування (ООП, іноді об'єктно-орієнтоване

програмування; від англ. Object-oriented programming, OOP) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою.

Попри те, що ця парадигма з'явилась в 1960-х роках, вона не мала широкого застосування до 1990-х, коли розвиток комп'ютерів та комп'ютерних мереж дав змогу писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Сьогодні багато мов програмування або підтримують ООП (PHP, Lua), або ж є цілком об'єкто-орієнтованими (зокрема, Java, C#, C++, Python, Ruby, Swift). Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування Симула в 1960-х роках, одночасно з посиленням дискусій про кризу програмного забезпечення. Через ускладнення апаратного та програмного забезпечення було дуже важко зберегти якість програм.

Об'єкто-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми.

Дана парадигма програмування базується на чотирьох основних принципах: інкапсуляція, наслідування, абстракція та поліморфізм.

Інкапсуляція – це приховування деталей про роботу класів від об'єктів, що їх використовують або надсилають їм повідомлення. Інкапсуляція досягається шляхом вказування, які класи можуть звертатися до членів об'єкта. Як наслідок, кожен об'єкт надає кожному іншому класу певний інтерфейс — члени, доступні іншим класам. Інкапсуляція потрібна для того, аби запобігти використанню користувачами інтерфейсу тих частин реалізації, які, швидше за все, будуть змінюватись. Це дасть змогу полегшити внесення змін без потреби змінювати й користувачів інтерфейсу. [[1]](https://uk.wikipedia.org/wiki/Об%27єктно-орієнтоване_програмування)

Автоматизація процесів управління аеропортами є актуальною темою в контексті розвитку інформаційних технологій та підвищення ефективності транспортних послуг. Програми, подібні до "Аеропорту", можуть бути використані не лише для внутрішнього управління рейсами, але й для надання зручних сервісів пасажирам, таких як онлайн-бронювання квитків та отримання інформації про маршрути.

Аеропорти є важливими інфраструктурними об'єктами, які забезпечують зручність і безпеку пасажирських перевезень. Ефективне управління рейсами на автовокзалах потребує використання сучасних інформаційних технологій для автоматизації процесів, що дозволяє зменшити людський фактор і підвищити точність та оперативність обробки даних.

У цій розрахунково-графічній роботі ми розглядаємо процес розробки програми "Аеропорт" за допомогою мови програмування C# та середовища розробки Windows Forms. Метою цієї програми є автоматизація основних операцій з управління рейсами, таких як додавання нових маршрутів, видалення існуючих та перегляд наявних маршрутів.

Програма має наступні основні функції:

1. Додавання рейсу: користувач вводить номер літака, пункт відправлення, пункт призначення, час відправлення та вартість квитка;
2. Видалення рейсу: видалення рейсу за номером літака;
3. Перегляд всіх рейсів: відображення списку всіх рейсів у зручному для користувача форматі.

Для реалізації програми використовується об'єктно-орієнтований підхід, що дозволяє структурувати дані та функціональність у вигляді окремих класів. Це сприяє зручності розробки та підтримки програмного забезпечення.

Програма складається з наступних компонентів:

1. Клас Flight.cs: описує окремий автобусний маршрут, зберігаючи інформацію про номер автобуса, пункт відправлення, пункт призначення, час відправлення та вартість квитка;
2. Головна форма Form1: реалізує графічний інтерфейс користувача (GUI), що дозволяє взаємодіяти з програмою за допомогою кнопок та списку маршрутів (ListBox).

Розробка програми "Аеропорт" демонструє використання таких ключових концепцій програмування, як об'єктно-орієнтоване програмування, робота з графічним інтерфейсом, обробка подій та взаємодія з користувачем. Виконання цієї роботи дозволяє закріпити теоретичні знання та отримати практичний досвід розробки програмного забезпечення.

**РОЗДІЛ 1**

**ПОСТАНОВКА ЗАДАЧІ**

Розробити програму «Аеропорт». Властивості: номері рейсу, назва літака, пункті відправлення та призначення, час відправлення, час прибуття, вартість квитка. Функції: «додати рейс», «видалити рейс», «перегляд всіх рейсів».

**1.1 Мова програмування С#**

**C#** (вимовляється *Сі-шарп*) — об'єктно-орієнтована мова програмував-ня з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, [Скотом Вілтамутом](https://uk.wikipedia.org/w/index.php?title=%D0%92%D1%96%D0%BB%D1%82%D0%B0%D0%BC%D1%83%D1%82_%D0%A1%D0%BA%D0%BE%D1%82&action=edit&redlink=1) та [Пітером Гольде](https://uk.wikipedia.org/w/index.php?title=%D0%93%D0%BE%D0%BB%D1%8C%D0%B4%D0%B5_%D0%9F%D1%96%D1%82%D0%B5%D1%80&action=edit&redlink=1) під егідою Microsoft Research (належить Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників — мов С++, Object Pascal, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова С#, на відміну від C++, не передбачає множинне успадкування класів.

Станом на вересень 2023 року поточна стабільна версія мови C# 11.0, яка була випущена в 2022 році як частина платформи .NET 7.0.[[2]](https://uk.wikipedia.org/wiki/C_Sharp)

Більшість програм та сервісів для вебу, пов’язаних із продуктами Microsoft, написано на C#. Відомий сайт stackoverflow.com написаний на C# із використанням фреймворку.NET. Подібним методом зроблені сайти Microsoft і Dell.

С# застосовується для розробки десктопних Windows-додатків. Наприклад, Microsoft Visual Studio, Paint.NET написані на C#. Платформа.NET, точніше, Windows Forms лежить у основі Skype, Microsoft Office, Photoshop.

Геймдев-розробники знають про рушій Unity. Він використовується для створення 2D та 3D комп’ютерних ігор. Програмування в Unity дає змогу зосередитися на контенті гри, не торкаючись багатьох технічних деталей. Воно робиться переважно на C#. Ігри Rust, Hearthstone, Fall Guys та інші зроблені в Unity з використанням C#. Крім цього, він дає змогу взаємодіяти з DirectX — набором компонентів для графіки та звуку в іграх.

C# також використовується для розробки мобільних додатків. Є платформи (наприклад, Xamarin), о дають змогу запускати код C# на різних операційних системах, включно з мобільними OS Android та iOS.

У 2018 до програмного середовища.NET додали бібліотеку ML.NET, яка дає змогу користуватися моделями машинного навчання. Вона поступається бібліотекам на Python за кількістю документації, але зручна та часом використовується в продакшені.[[3]](https://robotdreams.cc/uk/blog/284-s-chto-eto-za-yazyk-i-gde-ego-ispolzuyut)

**1.2 Призначення розкладів аеропорту та їх функціонал**

Розклади аеропорту є важливою частиною управління пасажирськими перевезеннями. Вони виконують кілька ключових функцій, які забезпечують ефективну організацію транспортних послуг і зручність для пасажирів.

**Призначення розкладів автовокзалу**

1. Організація пасажирських перевезень: Розклади допомагають структуровано організувати пасажирські перевезення. Вони визначають час відправлення та прибуття літаків, що дозволяє пасажирам планувати свої поїздки і забезпечує регулярність руху;
2. Забезпечення регулярності та точності: Регулярні розклади гарантують, що літаки відправляються та прибувають в певний час. Це дозволяє пасажирам бути впевненими в точності часу своєї подорожі і знижує ймовірність затримок;
3. Оптимізація використання ресурсів: Розклади допомагають оптимально використовувати транспортні ресурси. Вони дозволяють планувати кількість літаків на маршруті в залежності від попиту, що допомагає знизити витрати і підвищити ефективність роботи автовокзалу;
4. Забезпечення безпеки: Заздалегідь визначені розклади допомагають забезпечити безпеку пасажирів. Вони знижують хаос і зупинки в нерегулярних місцях, що може бути небезпечно.

**Функціонал розкладів аеропорта**

1. Планування рейсів:

* Визначення точного часу відправлення та прибуття;
* Встановлення періодичності рейсів (щоденно, щотижнево, у певні дні).

1. Інформаційне забезпечення пасажирів:

* Надання актуальної інформації про розклад руху через електронні табло, інформаційні кіоски та онлайн-сервіси;
* Оновлення розкладу в режимі реального часу для відображення затримок або змін,

1. Продаж квитків:

* Інтеграція з системами продажу квитків для забезпечення доступу пасажирів до інформації про доступні рейси та можливість бронювання квитків.

1. Моніторинг та аналітика:

* Використання даних розкладу для моніторингу ефективності роботи маршрутів;
* Аналіз пасажиропотоку для оптимізації розкладу і підвищення ефективності перевезень.

1. Координація з іншими видами транспорту:

* Узгодження розкладів з іншими видами транспорту (залізничні вокзали, аеропорти) для забезпечення зручних пересадок.[[4]](https://chatgpt.com/)

**РОЗДІЛ 2**

**ОПИС ФУНКЦІОНАЛУ КЛАСІВ**

Дана розрахунково графчіна робота включає в себе проект в середовищі розробки Visual Studio 2022 за допомогою Windows Forms (.NET Framework 4.8) під назвою “RGR”.

**2.1 Класи проекту Windows Forms**

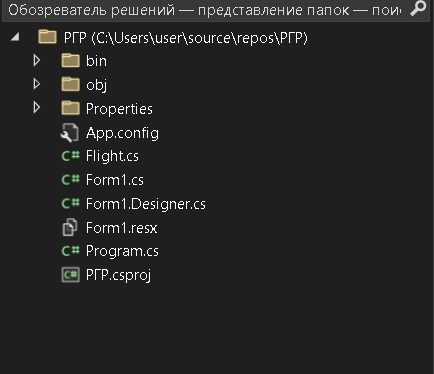


Рисунок 2.1 – Структура проекту Windows Forms

Програма складається з головного вікна та основної панелі, на якій в залежності від обраної вкладки з’являється відповідна форма для управління розкладами літаків. Основні форми зберігаються у папці Forms. Програма також містить допоміжні класи для роботи з рейсами літаків, дизайну інтерфейсу та управління головним вікном.

**Класи і їх функціонал:**

* + - * 1. Клас Flight.cs: Клас моделі, що представляє рейс літака. Містить властивості для зберігання номера літака, пункту відправлення, пункту призначення, часу відправлення та вартості квитка;
        2. Клас Form1.cs: Форма головного вікна програми. Містить основну панель, на якій в залежності від обраної вкладки з’являється відповідна форма для управління розкладами рейсів. Описує події натискання бокових вкладок (кнопок);
        3. Клас Form1Design.cs: Допоміжний клас для опису дизайну інтерфейсу користувача головного вікна. Містить налаштування стилів, кольорів, шрифтів та інших елементів дизайну;
        4. Клас Program.cs: Точка входу програми. Відповідає за запуск головного вікна програми та ініціалізацію основних компонентів.

**РОЗДІЛ 3**

**ДИЗАЙН ІНТЕРФЕЙСУ**

**3.1 Головне вікно програми**

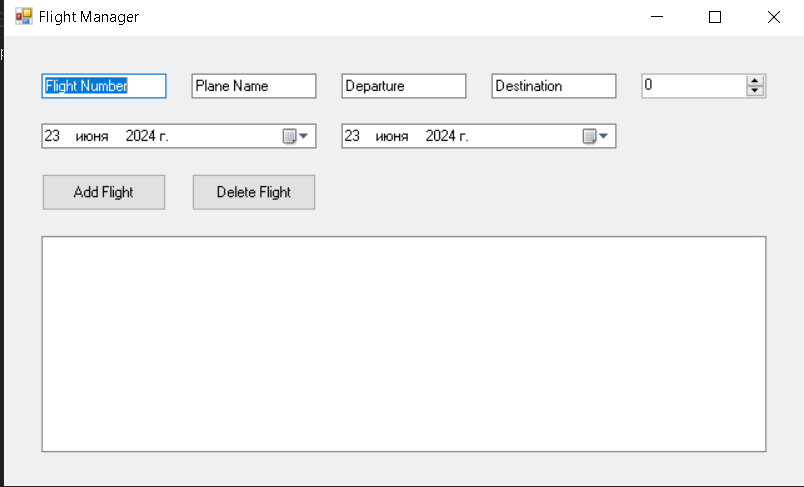


Рисунок 3.1 – Головне вікно програми

**3.2 Додавання даних рейсу**



Рисунок 3.2 – Номер автобуса

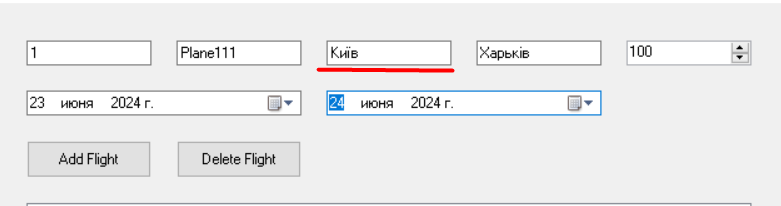


Рисунок 3.3 – Місце відправлення

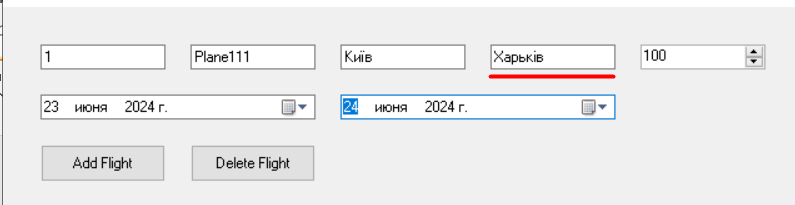


Рисунок 3.4 – Місце прибуття

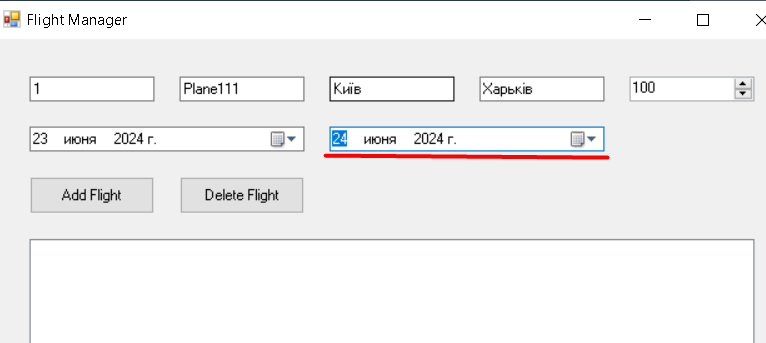


Рисунок 3.5 – Час прибуття

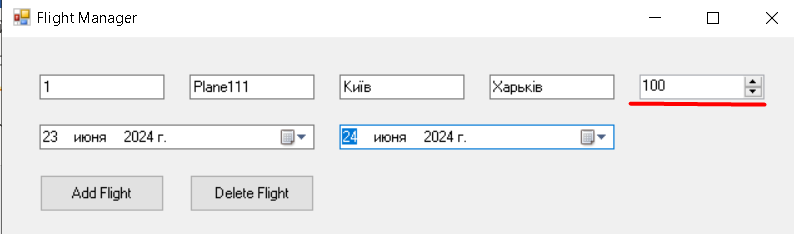


Рисунок 3.6 – Ціна квитка

**3.3 Головне вікно після додавання рейсу**

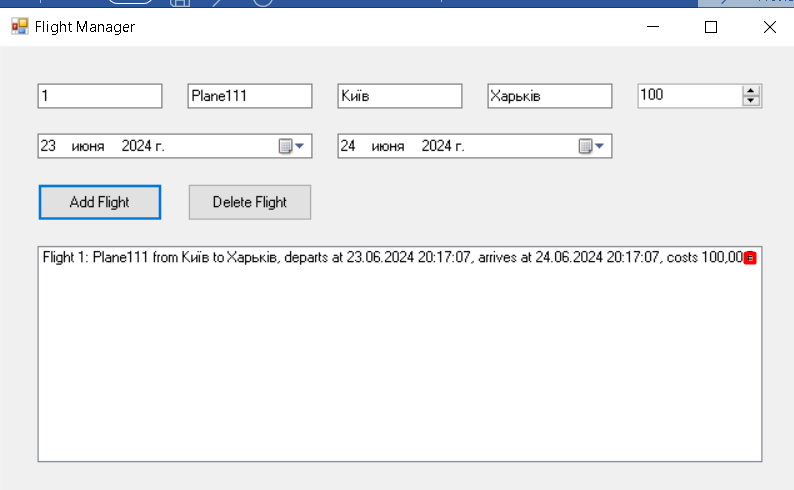


Рисунок 3.7 – Після додавання рейсу

**3.4 Видалення рейсу**

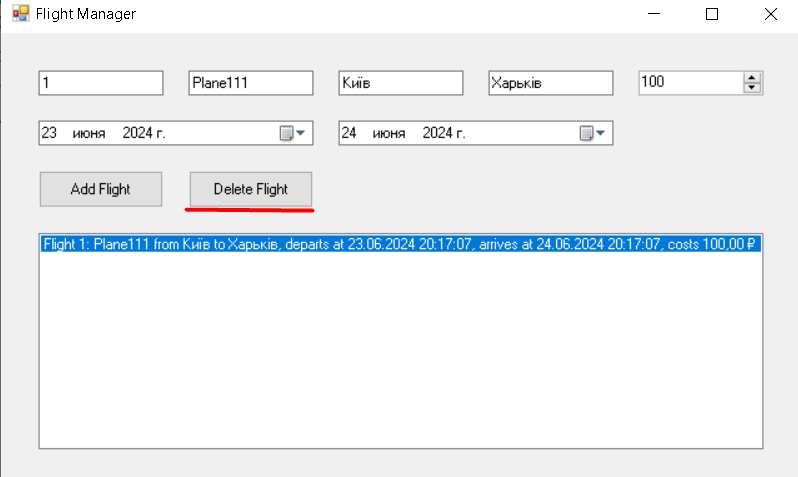


Рисунок 3.8 – Видалення рейсу

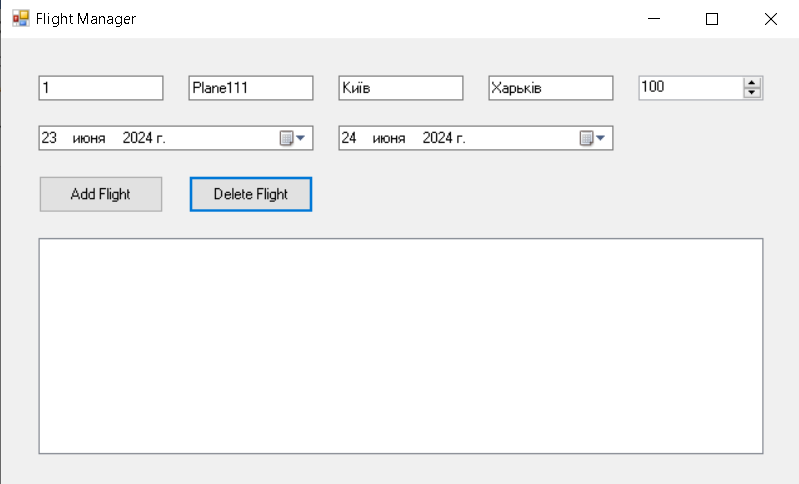


Рисунок 3.9 – Результат видалення рейсу

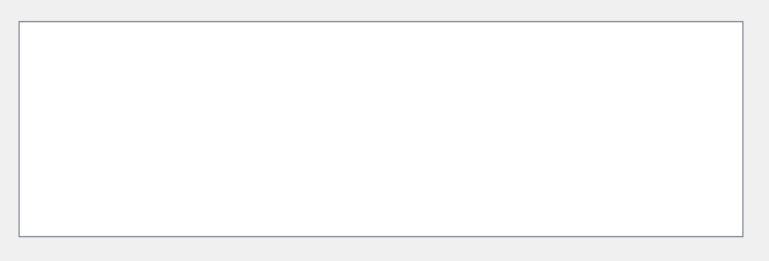


Рисунок 3.10 – Показ всіх рейсів

**РОЗДІЛ 4**

**ВИХІДНИЙ КОД ПРОЕКТУ**

**4.1 Клас “Flight”**

Клас BusRoute представляє окремий рейс. Він містить інформацію про рейс, таку як номер літака, пункт відправлення, пункт призначення, час відправлення та вартість квитка.

**Код:**

using System;

namespace РГР

{

public class Flight

{

public string FlightNumber { get; set; }

public string PlaneName { get; set; }

public string Departure { get; set; }

public string Destination { get; set; }

public DateTime DepartureTime { get; set; }

public DateTime ArrivalTime { get; set; }

public decimal TicketPrice { get; set; }

public Flight(string flightNumber, string planeName, string departure, string destination, DateTime departureTime, DateTime arrivalTime, decimal ticketPrice)

{

FlightNumber = flightNumber;

PlaneName = planeName;

Departure = departure;

Destination = destination;

DepartureTime = departureTime;

ArrivalTime = arrivalTime;

TicketPrice = ticketPrice;

}

public override string ToString()

{

return $"Flight {FlightNumber}: {PlaneName} from {Departure} to {Destination}, departs at {DepartureTime}, arrives at {ArrivalTime}, costs {TicketPrice:C}";

}

}

}

**Призначення команд:**

**using System;** - Це директива, яка дозволяє використовувати класи та типи з простору імен System, наприклад, DateTime.

**namespace РГР** - Визначає простір імен РГР, щоб згрупувати пов'язані класи та інтерфейси.

**public class Flight** - Оголошує публічний клас Flight, який буде доступний з інших частин програми.

**public string FlightNumber { get; set; }** - Автоматична властивість для зберігання номера рейсу.

**public string PlaneName { get; set; }** - Автоматична властивість для зберігання назви літака.

**public string Departure { get; set; }** - Автоматична властивість для зберігання місця вильоту.

**public string Destination { get; set; }** - Автоматична властивість для зберігання місця призначення.

**public DateTime DepartureTime { get; set; }** - Автоматична властивість для зберігання часу вильоту.

**public DateTime ArrivalTime { get; set; }** - Автоматична властивість для зберігання часу прибуття.

**public decimal TicketPrice { get; set; }** - Автоматична властивість для зберігання вартості квитка.

**public Flight(string flightNumber, string planeName, string departure, string destination, DateTime departureTime, DateTime arrivalTime, decimal ticketPrice)** - Конструктор класу Flight, який ініціалізує всі властивості при створенні об'єкта.

**FlightNumber = flightNumber;** - Ініціалізація властивості FlightNumber значенням аргументу конструктора flightNumber.

**PlaneName = planeName;** - Ініціалізація властивості PlaneName значенням аргументу конструктора planeName.

**Departure = departure;** - Ініціалізація властивості Departure значенням аргументу конструктора departure.

**Destination = destination;** - Ініціалізація властивості Destination значенням аргументу конструктора destination.

**DepartureTime = departureTime;** - Ініціалізація властивості DepartureTime значенням аргументу конструктора departureTime.

**ArrivalTime = arrivalTime;** - Ініціалізація властивості ArrivalTime значенням аргументу конструктора arrivalTime.

**TicketPrice = ticketPrice;** - Ініціалізація властивості TicketPrice значенням аргументу конструктора ticketPrice.

**public override string ToString()** - Перевизначення методу ToString для класу Flight, щоб повертати рядок, що представляє об'єкт у зручному для читання вигляді.

**return $"Flight {FlightNumber}: {PlaneName} from {Departure} to {Destination}, departs at {DepartureTime}, arrives at {ArrivalTime}, costs {TicketPrice:C}";** - Повертає рядок, який описує рейс, використовуючи інтерполяцію рядків для включення значень властивостей об'єкта.

**4.3 Клас Form1**

Клас Form1 відповідає за інтерфейс користувача та взаємодію з користувачем. Він містить елементи управління, такі як кнопки та список, і обробляє події, що виникають при взаємодії користувача з цими елементами.

**Код:**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace РГР

{

public partial class Form1 : Form

{

private List<Flight> flights;

public Form1()

{

InitializeComponent();

flights = new List<Flight>();

}

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

{

string flightNumber = txtFlightNumber.Text;

string planeName = txtPlaneName.Text;

string departure = txtDeparture.Text;

string destination = txtDestination.Text;

DateTime departureTime = dtpDepartureTime.Value;

DateTime arrivalTime = dtpArrivalTime.Value;

decimal ticketPrice = nudTicketPrice.Value;

Flight flight = new Flight(flightNumber, planeName, departure, destination, departureTime, arrivalTime, ticketPrice);

flights.Add(flight);

UpdateFlightList();

}

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

{

if (lstFlights.SelectedItem is Flight selectedFlight)

{

flights.Remove(selectedFlight);

UpdateFlightList();

}

}

private void UpdateFlightList()

{

lstFlights.Items.Clear();

foreach (var flight in flights)

{

lstFlights.Items.Add(flight);

}

}

}

}

**Призначення команд:**

**using System;** - Підключення простору імен для базових класів .NET, наприклад, DateTime.

**using System.Collections.Generic;** - Підключення простору імен для використання колекцій, таких як List.

**using System.Windows.Forms;** - Підключення простору імен для створення додатків Windows Forms.

**namespace РГР** - Визначення простору імен РГР для організації коду.

**public partial class Form1 : Form** - Оголошення часткового класу Form1, який успадковує клас Form з Windows Forms.

**private List<Flight> flights;** - Оголошення приватного поля flights, яке зберігає список рейсів.

**public Form1()** - Конструктор класу Form1, який ініціалізує компоненти форми і створює новий список рейсів.

**InitializeComponent();** - Метод, який викликається для ініціалізації компонентів форми (генерується автоматично дизайнером форми).

**flights = new List<Flight>();** - Ініціалізація списку рейсів.

**private void btnAddFlight\_Click(object sender, EventArgs e)** - Метод, який викликається при натисканні кнопки "Add Flight".

**string flightNumber = txtFlightNumber.Text;** - Отримання номера рейсу з текстового поля txtFlightNumber.

**string planeName = txtPlaneName.Text;** - Отримання назви літака з текстового поля txtPlaneName.

**string departure = txtDeparture.Text;** - Отримання місця вильоту з текстового поля txtDeparture.

**string destination = txtDestination.Text;** - Отримання місця призначення з текстового поля txtDestination.

**DateTime departureTime = dtpDepartureTime.Value;** - Отримання часу вильоту з елемента вибору дати dtpDepartureTime.

**DateTime arrivalTime = dtpArrivalTime.Value;** - Отримання часу прибуття з елемента вибору дати dtpArrivalTime.

**decimal ticketPrice = nudTicketPrice.Value;** - Отримання вартості квитка з числового елемента nudTicketPrice.

**Flight flight = new Flight(flightNumber, planeName, departure, destination, departureTime, arrivalTime, ticketPrice);** - Створення нового об'єкта Flight з отриманих даних.

**flights.Add(flight);** - Додавання нового рейсу до списку flights.

**UpdateFlightList();** - Виклик методу для оновлення списку рейсів на формі.

**private void btnDeleteFlight\_Click(object sender, EventArgs e)** - Метод, який викликається при натисканні кнопки "Delete Flight".

**if (lstFlights.SelectedItem is Flight selectedFlight)** - Перевірка, чи вибраний елемент у списку є об'єктом Flight.

**flights.Remove(selectedFlight);** - Видалення вибраного рейсу зі списку flights.

**UpdateFlightList();** - Виклик методу для оновлення списку рейсів на формі.

**private void UpdateFlightList()** - Метод для оновлення списку рейсів на формі.

**lstFlights.Items.Clear();** - Очищення списку елементів на формі.

**foreach (var flight in flights)** - Цикл для проходження по кожному рейсу в списку flights.

**lstFlights.Items.Add(flight);** - Додавання рейсу до списку елементів на формі lstFlights.

**ВИСНОВОК**

У даній курсовій роботі було розроблено програму для управління розкладами рейсів арепорту. Основною метою проекту було створення зручного та ефективного інтерфейсу для додавання, видалення та перегляду рейсів літаків. Програма реалізована з використанням мови програмування C# та середовища розробки Visual Studio. У процесі розробки було застосовано об'єктно-орієнтований підхід, що забезпечило високу модульність, розширюваність та зручність у підтримці коду.

Програма складається з кількох основних компонентів, кожен з яких виконує свою конкретну функцію

Програма забезпечує зручний інтерфейс для користувача, дозволяючи легко додавати нові рейси, видаляти існуючі та переглядати всі рейси у зручному форматі. Інтерфейс програми є інтуїтивно зрозумілим, що дозволяє користувачам швидко освоїтись з роботою програми. Завдяки застосуванню об'єктно-орієнтованого підходу, програма має високу модульність та розширюваність, що дозволяє легко додавати нові функції та форми у майбутньому.

У процесі розробки програми було опрацьовано кілька важливих аспектів, таких як:

1. **Створення класів для моделювання даних**: реалізація класу Flight, який містить усі необхідні властивості для представлення рейсу літака;
2. **Розробка графічного інтерфейсу користувача**: створення форм для додавання, видалення та перегляду рейсів, а також забезпечення уніфікованого дизайну інтерфейсу;
3. **Обробка подій та взаємодія між формами**: реалізація подій натискання кнопок для перемикання між формами, додавання нових рейсів, видалення існуючих та відображення списку рейсів.

Ця курсова робота дала змогу закріпити теоретичні знання з програмування на C# та розробки програм з графічним інтерфейсом, а також набути практичного досвіду у створенні комплексних програмних рішень. Проект став гарним прикладом застосування об'єктно-орієнтованих принципів у розробці програмного забезпечення, що підвищило якість та підтримуваність коду.

У майбутньому програму можна розширити, додавши нові функціональні можливості, такі як:

1. Пошук рейсів за різними критеріями (номер літака, пункт відправлення, пункт призначення тощо);
2. Редагування існуючих рейсів;
3. Імпорт та експорт даних про рейси у різних форматах;
4. Інтеграція з базою даних для зберігання великої кількості рейсів та забезпечення швидкого доступу до даних.

Реалізація цих можливостей дозволить зробити програму ще більш корисною та зручною для користувачів, а також забезпечить її гнучкість та масштабованість.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Об’єктно-орієнтовне програмування. URL: <https://uk.wikipedia.org/wiki/Об%27єктно-орієнтоване_програмування> (дата звернення: 12.06.2024)
2. C SHARP. URL: <https://uk.wikipedia.org/wiki/C_Sharp> (дата звернення: 13.06.2024)
3. C# Що це за мова та де її використовують. URL: <https://robotdreams.cc/uk/blog/284-s-chto-eto-za-yazyk-i-gde-ego-ispolzuyut> (дата звернення: 13.06.2024)
4. ChatGPT. URL: <https://chatgpt.com/> (дата звернення: 17.06.2024)