**Лабораторна робота №6.**

**Тема:** Делегати

**Мета:** Одержати практичні навички створення та роботи з делегатами.

**Основний зміст роботи.**

Створити декілька делегатів і використати їх для даних різних методів.

**Задача №1**

В проєкті до Задачі з ЛР5 (**с#**) програмно реалізувати декілька делегатів(2-4), застосовуючи різні способи їх використання. Прокоментувати алгоритм роботи з делегатами в проєкті.

**Порядок виконання задачі:**

* **Додано** делегат стратегії CostStrategy(Podorozh) і набір стратегій TripDelegates (BaseCost, WithAccommodation, BusinessMarkup).
* **Додано** властивість RevenueStrategy у TravelAgency та метод TotalAgencyRevenueEstimate(CostStrategy overrideStrategy = null) для підміни формули на льоту.
* **Додано** мультикаст-логер TripLogger(string) у Form1 з підписниками: у пам’ять, у Debug, у TextBox.
* **Додано** централізовану шину логів AppLog.Write + AppLog.Msg(...), щоб будь-який клас міг логувати.
* **Додано** подію event Action<Accommodation> AccommodationAdded у Podorozh і лямбда-обробник у Form1 (лог + оновлення таблиці).
* **Додано** предикат-фільтр Predicate<Podorozh> \_currentFilter для динамічної фільтрації списку подорожей.
* **Додано** логування ключових дій у домені: композиція (Itinerary.Add/RemoveAccommodation), агрегація (TravelAgency.Add/RemoveTrip, підсумок доходу), оператори у Podorozh (+, >, <, ==, !=, ++, --).
* **Додано** розрахунок «Вартість проживання» без N/A: фактичне з Itinerary або оцінка EstimateAccommodation (за транспортом × дні) з відображенням у гріді.

**Код програми:**

**MainForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using static LR1\_3.TravelAgency;

namespace LR1\_3

{

public partial class Form1 : Form

{

private TravelAgency agency;

private Podorozh currentTrip;

private readonly List<Podorozh> podorozhi = new List<Podorozh>();

private readonly Dictionary<Podorozh, string> tripAgencyMap = new Dictionary<Podorozh, string>();

private readonly Dictionary<string, TravelAgency> agencies = new Dictionary<string, TravelAgency>();

// Делегати/логер/фільтр

private readonly List<string> \_logs = new List<string>();

private TripLogger \_logger;

private Predicate<Podorozh> \_currentFilter;

public Form1()

{

InitializeComponent();

// 1) Агентство

agency = new TravelAgency("Моя агенція");

agencies["Моя агенція"] = agency;

agency.RevenueStrategy = TripDelegates.WithAccommodation; // можна BaseCost/BusinessMarkup

// 2) Логер (мультикаст)

\_logger += LogToMemory;

\_logger += msg => Debug.WriteLine("[FormLog] " + msg);

\_logger += AppendLog; // щоб писало у textBox1 (ти вже маєш AppendLog)

AppLog.Write = msg => \_logger?.Invoke(msg); // ← ВАЖЛИВО: вся доменна логіка тепер може логувати

\_logger?.Invoke("Лог-шина під’єднана.");

// 3) Початковий фільтр

\_currentFilter = p => true;

// 4) Таблиці

dataGridView1.AutoGenerateColumns = false;

dataGridView1.Columns.Add("Misto", "Місто");

dataGridView1.Columns.Add("Vidstan", "Відстань (км)");

dataGridView1.Columns.Add("Dni", "Днів");

dataGridView1.Columns.Add("Transport", "Транспорт");

dataGridView1.Columns.Add("Vartist", "Вартість");

dataGridView1.Columns.Add("Serednya", "Середня ціна/км");

dataGridView1.Columns.Add("TypTuru", "Тип туру");

dataGridView1.Columns.Add("VartistProzhyvannya", "Вартість проживання");

dataGridView1.Columns.Add("ZagalnaVartist", "Загальна вартість");

dataGridView1.Columns.Add("AgencyName", "Агенція");

dataGridView2.AutoGenerateColumns = false;

dataGridView2.Columns.Add("Misto", "Місто");

dataGridView2.Columns.Add("Vidstan", "Відстань (км)");

dataGridView2.Columns.Add("Dni", "Днів");

dataGridView2.Columns.Add("Transport", "Транспорт");

dataGridView2.Columns.Add("Vartist", "Вартість");

dataGridView2.Columns.Add("Serednya", "Середня ціна/км");

dataGridView1.SelectionChanged += DataGridView1\_SelectionChanged;

}

// ==== Допоміжні: лог, оцінка проживання, базова/похідна вартість ====

private void LogToMemory(string message)

{

\_logs.Add($"[{DateTime.Now:HH:mm:ss}] {message}");

}

// Груба оцінка проживання за ніч (коли житла ще немає в Itinerary)

private double EstimateAccommodation(Podorozh trip)

{

double perNight =

trip.Transport.Equals("Літак", StringComparison.OrdinalIgnoreCase) ? 1200 :

trip.Transport.Equals("Поїзд", StringComparison.OrdinalIgnoreCase) ? 800 :

trip.Transport.Equals("Автобус", StringComparison.OrdinalIgnoreCase) ? 600 : 700;

return perNight \* Math.Max(1, trip.Dni);

}

// Базова/похідна вартість:

// - для Туристичної вже включає проживання (VartistProzhyvannya)

// - для інших підтипів повертає їхню спец-формулу (без житла)

private double BaseOrSubclassCost(Podorozh p)

{

if (p is ITurystychnaPodorozh t) return t.RozrakhVartistPodorozh(); // вже з проживанням

if (p is IBusinessPodorozh b) return b.RozrakhVartistPodorozh();

if (p is IExtremePodorozh ex) return ex.RozrakhVartistPodorozh();

if (p is ILongTermPodorozh l) return l.RozrakhVartistPodorozh();

return p.RozrakhVartist();

}

// ==== UI події ====

private void DataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 1)

{

int idx = dataGridView1.SelectedRows[0].Index;

if (idx >= 0 && idx < podorozhi.Count) currentTrip = podorozhi[idx];

}

}

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

{

// --- Діалог створення ---

var inputForm = new Form

{

Text = "Створити подорож",

Width = 380,

Height = 340,

StartPosition = FormStartPosition.CenterParent,

FormBorderStyle = FormBorderStyle.FixedDialog,

MaximizeBox = false,

MinimizeBox = false

};

var lblMisto = new Label { Text = "Місто:", Left = 12, Top = 20, Width = 140 };

var lblVidstan = new Label { Text = "Відстань (км):", Left = 12, Top = 60, Width = 140 };

var lblDni = new Label { Text = "Кількість днів:", Left = 12, Top = 100, Width = 140 };

var lblTransport = new Label { Text = "Вид транспорту:", Left = 12, Top = 140, Width = 140 };

var lblTypTuru = new Label { Text = "Вид подорожі:", Left = 12, Top = 180, Width = 140 };

var txtMisto = new TextBox { Left = 160, Top = 18, Width = 190 };

var txtVidstan = new TextBox { Left = 160, Top = 58, Width = 190 };

var txtDni = new TextBox { Left = 160, Top = 98, Width = 190 };

var cbTransport = new ComboBox

{

Left = 160,

Top = 138,

Width = 190,

DropDownStyle = ComboBoxStyle.DropDownList

};

cbTransport.Items.AddRange(new[] { "Автобус", "Поїзд", "Літак", "Авто" });

cbTransport.SelectedIndex = 0;

var cbTypTuru = new ComboBox

{

Left = 160,

Top = 178,

Width = 190,

DropDownStyle = ComboBoxStyle.DropDownList

};

cbTypTuru.Items.AddRange(new[] { "Туристична", "Бізнес", "Екстремальна", "Довготривала" });

cbTypTuru.SelectedIndex = 0;

var btnOK = new Button { Text = "OK", Left = 110, Top = 235, Width = 100, DialogResult = DialogResult.OK };

var btnCancel = new Button { Text = "Скасувати", Left = 220, Top = 235, Width = 130, DialogResult = DialogResult.Cancel };

txtVidstan.KeyPress += (s, ke) =>

{

if (!char.IsControl(ke.KeyChar) && !char.IsDigit(ke.KeyChar) && ke.KeyChar != '.') ke.Handled = true;

if (ke.KeyChar == '.' && (s as TextBox).Text.Contains(".")) ke.Handled = true;

};

txtDni.KeyPress += (s, ke) =>

{

if (!char.IsControl(ke.KeyChar) && !char.IsDigit(ke.KeyChar)) ke.Handled = true;

};

inputForm.Controls.AddRange(new Control[]

{

lblMisto, lblVidstan, lblDni, lblTransport, lblTypTuru,

txtMisto, txtVidstan, txtDni, cbTransport, cbTypTuru,

btnOK, btnCancel

});

inputForm.AcceptButton = btnOK;

inputForm.CancelButton = btnCancel;

if (inputForm.ShowDialog() != DialogResult.OK) return;

// --- Валідація ---

string misto = txtMisto.Text.Trim();

if (string.IsNullOrWhiteSpace(misto))

{

MessageBox.Show("Вкажіть місто.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!double.TryParse(txtVidstan.Text, out double vidstan))

{

MessageBox.Show("Невірний формат відстані.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!int.TryParse(txtDni.Text, out int dni))

{

MessageBox.Show("Невірний формат кількості днів.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string transport = cbTransport.SelectedItem?.ToString() ?? "Автобус";

string typTuru = cbTypTuru.SelectedItem?.ToString() ?? "Туристична";

// --- Створення потрібного типу ---

IPodorozh p;

switch (typTuru)

{

case "Туристична":

double vartistProzh = 1000; // стартова (можете спитати у користувача)

p = new TurystychnaPodorozh(misto, vidstan, dni, transport, typTuru, vartistProzh);

break;

case "Бізнес":

p = new BusinessPodorozh(misto, vidstan, dni, transport, "ACME Corp", "Переговори");

break;

case "Екстремальна":

p = new ExtremePodorozh(misto, vidstan, dni, transport, "Високий");

break;

case "Довготривала":

p = new LongTermPodorozh(misto, vidstan, dni, transport, 6);

break;

default:

MessageBox.Show("Невідомий тип подорожі.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Кидаємо у конкретний базовий тип

var trip = (Podorozh)p;

currentTrip = trip;

podorozhi.Add(trip);

// Підписка на подію додавання житла

trip.AccommodationAdded += acc =>

{

\_logger?.Invoke($"До подорожі '{trip.Misto}' додано житло: {acc.Name} — {acc.Nights} ноч., {acc.TotalPrice():0.##} грн");

UpdateGridTotalsForTrip(trip);

};

// --- РОЗРАХУНКИ ДЛЯ РЯДКА ТАБЛИЦІ ---

double accomCost; // для колонки "Вартість проживання"

double totalCostToShow; // для "Загальна вартість"

string realTypTuru = typTuru;

if (p is ITurystychnaPodorozh tTrip)

{

// Туристична: проживання вже є у властивості та враховане у RozrakhVartistPodorozh()

accomCost = tTrip.VartistProzhyvannya;

totalCostToShow = tTrip.RozrakhVartistPodorozh();

realTypTuru = tTrip.TypTuru;

}

else

{

// Інші типи: беремо реальне житло з Itinerary, якщо пусто — даємо оцінку

accomCost = trip.Itinerary.TotalAccommodationCost();

if (accomCost <= 0) accomCost = EstimateAccommodation(trip);

// Їхній «базовий» + житло

double baseOrSubclass = BaseOrSubclassCost(trip);

totalCostToShow = baseOrSubclass + accomCost;

}

// Додаємо рядок у грід

dataGridView1.Rows.Add(

trip.Misto,

trip.Vidstan,

trip.Dni,

trip.Transport,

trip.RozrakhVartist(),

trip.SerednyaTsinaKm().ToString("F2"),

realTypTuru,

accomCost.ToString("F2"), // ← більше не "N/A"

totalCostToShow.ToString("F2"),

tripAgencyMap.TryGetValue(trip, out var agName2) ? agName2 : "—"

);

\_logger?.Invoke($"Створено подорож: {trip.Info()}");

MessageBox.Show("Подорож створено і додано в таблицю.", "OK",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

if (dataGridView1.SelectedRows.Count != 1)

{

MessageBox.Show("Будь ласка, виберіть одну подорож для зміни типу!");

return;

}

int index = dataGridView1.SelectedRows[0].Index;

if (index < 0 || index >= podorozhi.Count)

{

MessageBox.Show("Некоректний вибір рядка.");

return;

}

var selectedTrip = podorozhi[index];

string newType = ShowTripTypePicker();

if (string.IsNullOrEmpty(newType)) return;

if (selectedTrip is TurystychnaPodorozh tur) tur.ZminytyTypTuru(newType);

else if (selectedTrip is BusinessPodorozh biz) biz.ZminytyTypTuru(newType);

else if (selectedTrip is ExtremePodorozh ext) ext.ZminytyTypTuru(newType);

else if (selectedTrip is LongTermPodorozh lng) lng.ZminytyTypTuru(newType);

if (dataGridView1.Columns.Contains("TypTuru"))

dataGridView1.Rows[index].Cells["TypTuru"].Value = newType;

MessageBox.Show("Тип туру змінено!");

}

private string ShowTripTypePicker()

{

var f = new Form

{

Text = "Вибір типу туру",

Width = 320,

Height = 170,

StartPosition = FormStartPosition.CenterParent,

FormBorderStyle = FormBorderStyle.FixedDialog,

MaximizeBox = false,

MinimizeBox = false

};

var lbl = new Label { Text = "Оберіть тип:", Left = 12, Top = 20, Width = 100 };

var cb = new ComboBox

{

Left = 120,

Top = 16,

Width = 170,

DropDownStyle = ComboBoxStyle.DropDownList

};

cb.Items.AddRange(new[] { "Туристична", "Бізнес", "Екстремальна", "Довготривала" });

cb.SelectedIndex = 0;

var ok = new Button { Text = "OK", Left = 70, Top = 70, Width = 80, DialogResult = DialogResult.OK };

var cancel = new Button { Text = "Скасувати", Left = 160, Top = 70, Width = 100, DialogResult = DialogResult.Cancel };

f.Controls.AddRange(new Control[] { lbl, cb, ok, cancel });

f.AcceptButton = ok;

f.CancelButton = cancel;

return f.ShowDialog() == DialogResult.OK ? cb.SelectedItem?.ToString() : null;

}

private void УвімкнутиФільтрЛітак3ПлюсДнів\_Click(object sender, EventArgs e)

{

\_currentFilter = p => p.Transport.Equals("Літак", StringComparison.OrdinalIgnoreCase) && p.Dni >= 3;

\_logger?.Invoke("Увімкнено фільтр: Літак і 3+ днів.");

}

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

{

var filtered = podorozhi.Where(p => \_currentFilter == null || \_currentFilter(p)).ToList();

Form2 formSearch = new Form2(filtered);

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

{

dataGridView2.Rows.Clear();

foreach (var p in formSearch.SearchResults)

{

dataGridView2.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport,

p.RozrakhVartist(),

p.SerednyaTsinaKm().ToString("F2")

);

}

}

}

private void RefreshRow(int index)

{

if (index < 0 || index >= podorozhi.Count) return;

Podorozh p = podorozhi[index];

dataGridView1.Rows[index].Cells[0].Value = p.Misto;

dataGridView1.Rows[index].Cells[1].Value = p.Vidstan;

dataGridView1.Rows[index].Cells[2].Value = p.Dni;

dataGridView1.Rows[index].Cells[3].Value = p.Transport;

dataGridView1.Rows[index].Cells[4].Value = p.RozrakhVartist();

dataGridView1.Rows[index].Cells[5].Value = p.SerednyaTsinaKm().ToString("F2");

}

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

{

if (dataGridView1.SelectedRows.Count == 2)

{

int index1 = dataGridView1.SelectedRows[0].Index;

int index2 = dataGridView1.SelectedRows[1].Index;

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

Podorozh combined = p1 + p2;

podorozhi.Add(combined);

dataGridView1.Rows.Add(

combined.Misto,

combined.Vidstan,

combined.Dni,

combined.Transport,

combined.RozrakhVartist(),

combined.SerednyaTsinaKm().ToString("F2"),

"—",

EstimateAccommodation(combined).ToString("F2"), // оцінка проживання

(BaseOrSubclassCost(combined) + EstimateAccommodation(combined)).ToString("F2"),

tripAgencyMap.TryGetValue(combined, out var agName3) ? agName3 : "—"

);

MessageBox.Show("Подорожі успішно об’єднано!");

}

else

{

MessageBox.Show("Виберіть рівно 2 подорожі для об’єднання!");

}

}

private void toolStripMenuItem4\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 2)

{

int index1 = dataGridView1.SelectedRows[0].Index;

int index2 = dataGridView1.SelectedRows[1].Index;

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

if (p1 > p2) MessageBox.Show($"{p1.Misto} дорожча за {p2.Misto}");

else if (p1 < p2) MessageBox.Show($"{p2.Misto} дорожча за {p1.Misto}");

else MessageBox.Show("Подорожі мають однакову вартість!");

}

else

{

MessageBox.Show("Виберіть рівно 2 подорожі для порівняння!");

}

}

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

{

if (dataGridView1.SelectedRows.Count == 2)

{

int index1 = dataGridView1.SelectedRows[0].Index;

int index2 = dataGridView1.SelectedRows[1].Index;

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

if (p1 == p2) MessageBox.Show("Подорожі однакові (місто + транспорт)");

else MessageBox.Show("Подорожі різні");

}

else

{

MessageBox.Show("Виберіть рівно 2 подорожі для перевірки!");

}

}

private void toolStripMenuItem2\_Click(object sender, EventArgs e)

{

int index = -1;

if (dataGridView1.SelectedRows.Count > 0)

index = dataGridView1.SelectedRows[0].Index;

else if (dataGridView1.SelectedCells.Count > 0)

index = dataGridView1.SelectedCells[0].RowIndex;

if (index >= 0 && index < podorozhi.Count)

{

podorozhi[index]++;

RefreshRow(index);

UpdateGridTotalsForTrip(podorozhi[index]);

}

else

{

MessageBox.Show("Виберіть одну подорож!");

}

}

private void toolStripMenuItem3\_Click(object sender, EventArgs e)

{

int index = -1;

if (dataGridView1.SelectedRows.Count > 0)

index = dataGridView1.SelectedRows[0].Index;

else if (dataGridView1.SelectedCells.Count > 0)

index = dataGridView1.SelectedCells[0].RowIndex;

if (index >= 0 && index < podorozhi.Count)

{

podorozhi[index]--;

RefreshRow(index);

UpdateGridTotalsForTrip(podorozhi[index]);

}

else

{

MessageBox.Show("Виберіть одну подорож!");

}

}

// Додати житло до поточної/виділеної

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

{

Podorozh targetTrip = null;

if (dataGridView1.SelectedRows.Count == 1)

{

int idx = dataGridView1.SelectedRows[0].Index;

if (idx >= 0 && idx < podorozhi.Count) targetTrip = podorozhi[idx];

}

if (targetTrip == null) targetTrip = currentTrip;

if (targetTrip == null)

{

MessageBox.Show("Оберіть подорож у таблиці або створіть поточну подорож.", "Помилка");

return;

}

string hotel = Prompt.ShowDialog("Введіть назву житла:", "Житло");

if (hotel == null) return;

string nightsStr = Prompt.ShowDialog("Кількість ночей:", "Житло");

if (nightsStr == null) return;

string priceStr = Prompt.ShowDialog("Ціна за ніч:", "Житло");

if (priceStr == null) return;

if (!int.TryParse(nightsStr, out int nights) || !double.TryParse(priceStr, out double price))

{

MessageBox.Show("Невірний формат чисел!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

targetTrip.AddAccommodation(hotel, nights, price);

UpdateGridTotalsForTrip(targetTrip);

\_logger?.Invoke($"Перераховано підсумки для '{targetTrip.Misto}': {targetTrip.TotalCostWithAccommodation():0.##} грн (разом)");

MessageBox.Show("Житло додано!", "OK");

}

private void AppendLog(string message)

{

string line = $"[{DateTime.Now:HH:mm:ss}] {message}{Environment.NewLine}";

if (textBox1.InvokeRequired)

{

textBox1.BeginInvoke(new Action<string>(AppendLog), message);

return;

}

// обмежуємо розмір, щоб не роздувався до безкінечності

if (textBox1.TextLength > 50000)

textBox1.Text = textBox1.Text.Substring(textBox1.TextLength - 40000);

textBox1.AppendText(line);

textBox1.SelectionStart = textBox1.TextLength;

textBox1.ScrollToCaret();

}

// Додати подорож до агенції

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

{

if (currentTrip == null)

{

MessageBox.Show("Спочатку створіть подорож!");

return;

}

string agencyName = Prompt.ShowDialog("Введіть назву агенції:", "Агенція");

if (string.IsNullOrWhiteSpace(agencyName)) return;

if (!agencies.TryGetValue(agencyName, out var ag))

{

ag = new TravelAgency(agencyName);

agencies[agencyName] = ag;

}

ag.AddTrip(currentTrip);

AppLog.Msg($"[Агрегація] '{currentTrip.Misto}' додано до агенції '{agencyName}'");

tripAgencyMap[currentTrip] = agencyName;

int idxRow = FindRowIndexForTrip(currentTrip);

if (idxRow >= 0 && dataGridView1.Columns.Contains("AgencyName"))

dataGridView1.Rows[idxRow].Cells["AgencyName"].Value = agencyName;

MessageBox.Show("Подорож додано до агенції!");

}

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

{

var sb = new StringBuilder();

sb.AppendLine("Подорожі за агенціями:");

foreach (var kv in agencies)

{

string name = kv.Key;

var ag = kv.Value;

sb.AppendLine($"— {name}:");

foreach (var trip in ag.Trips)

{

sb.AppendLine($" {trip.Misto} — {trip.TotalCostWithAccommodation():0.##} грн");

}

}

MessageBox.Show(sb.ToString());

}

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

{

double totalBusiness = agencies.Values.Sum(a => a.TotalAgencyRevenueEstimate(TripDelegates.BusinessMarkup));

double totalCurrent = agencies.Values.Sum(a => a.TotalAgencyRevenueEstimate());

MessageBox.Show(

$"За поточною стратегією: {totalCurrent:0.##} грн\n" +

$"З бізнес-націнкою: {totalBusiness:0.##} грн"

);

\_logger?.Invoke("Пораховано дохід агентств (дві стратегії).");

}

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

{

if (currentTrip == null)

{

MessageBox.Show("Спочатку створіть подорож!");

return;

}

MessageBox.Show(currentTrip.Itinerary.ToString(), "Житло подорожі");

}

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

{

if (currentTrip == null)

{

MessageBox.Show("Спочатку створіть подорож!");

return;

}

double total = currentTrip.TotalCostWithAccommodation();

MessageBox.Show($"Повна вартість подорожі (з житлом): {total:0.##} грн");

}

// ==== Підрахунки та оновлення рядка ====

private int FindRowIndexForTrip(Podorozh trip) => podorozhi.IndexOf(trip);

private void UpdateGridTotalsForTrip(Podorozh trip)

{

int idx = FindRowIndexForTrip(trip);

if (idx < 0 || idx >= dataGridView1.Rows.Count) return;

// 1) Проживання — реальне з Itinerary або оцінка

double accom = trip.Itinerary.TotalAccommodationCost();

if (accom <= 0) accom = EstimateAccommodation(trip);

// 2) База/похідна

double baseOrSubclass = BaseOrSubclassCost(trip);

// 3) Загальна: для туристичної база вже включає проживання

double total = (trip is ITurystychnaPodorozh)

? baseOrSubclass

: baseOrSubclass + accom;

if (dataGridView1.Columns.Contains("VartistProzhyvannya"))

dataGridView1.Rows[idx].Cells["VartistProzhyvannya"].Value = accom.ToString("F2");

if (dataGridView1.Columns.Contains("ZagalnaVartist"))

dataGridView1.Rows[idx].Cells["ZagalnaVartist"].Value = total.ToString("F2");

}

// ==== Збереження/завантаження ====

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

{

var dlg = new SaveFileDialog

{

Filter = "Текстові файли|\*.txt",

Title = "Збереження даних"

};

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

{

using (var sw = new System.IO.StreamWriter(dlg.FileName, false, Encoding.UTF8))

{

foreach (var p in podorozhi)

{

sw.WriteLine($"{p.Misto};{p.Vidstan};{p.Dni};{p.Transport};{p.RozrakhVartist()};{p.SerednyaTsinaKm():F2}");

}

}

AppLog.Msg($"[IO] Збереження у файл: {dlg.FileName} ({podorozhi.Count} записів)");

MessageBox.Show("Дані збережено!");

}

}

private void вивестиToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

var dlg = new OpenFileDialog

{

Filter = "Текстові файли|\*.txt",

Title = "Відкрити файл з даними"

};

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

{

podorozhi.Clear();

dataGridView1.Rows.Clear();

using (var sr = new System.IO.StreamReader(dlg.FileName, Encoding.UTF8))

{

string line;

while ((line = sr.ReadLine()) != null)

{

var parts = line.Split(';');

if (parts.Length >= 3)

{

string misto = parts[0];

double vidstan = double.Parse(parts[1]);

int dni = int.Parse(parts[2]);

string transport = parts.Length > 3 ? parts[3] : "Не вказано";

var p = new Podorozh(misto, vidstan, dni, transport);

podorozhi.Add(p);

// одразу порахуємо й покажемо оцінку проживання та загальну

double accom = p.Itinerary.TotalAccommodationCost();

if (accom <= 0) accom = EstimateAccommodation(p);

double total = BaseOrSubclassCost(p) + accom; // BaseOrSubclassCost тут == p.RozrakhVartist()

dataGridView1.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport,

p.RozrakhVartist(),

p.SerednyaTsinaKm().ToString("F2"),

"—",

accom.ToString("F2"),

total.ToString("F2"),

"—"

);

}

}

}

AppLog.Msg($"[IO] Завантаження з файлу: {dlg.FileName}");

AppLog.Msg($"[IO] Завантажено подорожей: {podorozhi.Count}");

MessageBox.Show("Дані завантажено!");

}

}

}

}

**Search Form.cs:**

//form2search

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace LR1\_3

{

public partial class Form2 : Form

{

private List<Podorozh> \_list;

public List<Podorozh> SearchResults { get; private set; }

public Form2(List<Podorozh> list)

{

InitializeComponent();

\_list = list;

comboBox1.Items.Clear();

comboBox1.Items.Add("Місто");

comboBox1.Items.Add("Транспорт");

comboBox1.Items.Add("Дні");

comboBox1.Items.Add("Тип туру"); // Додаємо новий критерій для пошуку по типу подорожі

comboBox1.Items.Add("Місто + Транспорт + Дні"); // для комбінованого пошуку

comboBox1.SelectedIndex = 0; // за замовчуванням перший елемент

comboBox2.Items.Clear();

comboBox2.Items.Add("Будь-який"); // для всіх

foreach (var t in \_list.Select(p => p.Transport).Distinct())

{

comboBox2.Items.Add(t);

}

comboBox2.SelectedIndex = 0; // за замовчуванням “Будь-який”

}

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

{

string crit = comboBox1.SelectedItem?.ToString(); // критерій пошуку

string mistoText = textBox1.Text.Trim();

string transportText = comboBox2.SelectedItem?.ToString();

string dniText = textBox2.Text.Trim();

var results = \_list.AsEnumerable();

if (!string.IsNullOrEmpty(crit))

{

switch (crit)

{

case "Місто":

if (string.IsNullOrEmpty(mistoText))

{

MessageBox.Show("Будь ласка, введіть місто для пошуку!");

return;

}

results = results.Where(p => p.Misto.IndexOf(mistoText, StringComparison.OrdinalIgnoreCase) >= 0);

break;

case "Транспорт":

if (string.IsNullOrEmpty(transportText))

{

MessageBox.Show("Будь ласка, введіть транспорт для пошуку!");

return;

}

results = results.Where(p => p.Transport.IndexOf(transportText, StringComparison.OrdinalIgnoreCase) >= 0);

break;

case "Дні":

if (string.IsNullOrEmpty(dniText))

{

MessageBox.Show("Будь ласка, введіть кількість днів для пошуку!");

return;

}

if (!int.TryParse(dniText, out int dni))

{

MessageBox.Show("Кількість днів повинна бути числом!");

return;

}

results = results.Where(p => p.Dni == dni);

break;

case "Тип туру": // Додаємо умову для пошуку по типу подорожі

if (string.IsNullOrEmpty(mistoText))

{

MessageBox.Show("Будь ласка, введіть тип туру для пошуку!");

return;

}

results = results.Where(p => (p is TurystychnaPodorozh trip) &&trip.TypTuru.IndexOf(mistoText, StringComparison.OrdinalIgnoreCase) >= 0);

break;

case "Місто + Транспорт + Дні":

if (string.IsNullOrEmpty(mistoText) || string.IsNullOrEmpty(transportText) || string.IsNullOrEmpty(dniText))

{

MessageBox.Show("Будь ласка, заповніть усі поля для комбінованого пошуку!");

return;

}

if (!int.TryParse(dniText, out int dniCombined))

{

MessageBox.Show("Кількість днів повинна бути числом!");

return;

}

results = results.Where(p =>p.Misto.IndexOf(mistoText, StringComparison.OrdinalIgnoreCase) >= 0 &&p.Transport.Equals(transportText, StringComparison.OrdinalIgnoreCase) &&p.Dni == dniCombined);

break;

}

}

var finalResults = results.ToList();

if (finalResults.Count == 0)

{

MessageBox.Show("Нічого не знайдено!");

return;

}

double totalCost = finalResults.Sum(p => p.RozrakhVartist());

double avgDays = finalResults.Average(p => p.Dni);

double minDistance = finalResults.Min(p => p.Vidstan);

double maxDistance = finalResults.Max(p => p.Vidstan);

MessageBox.Show($"Знайдено {finalResults.Count} подорожей\n" +$"Сумарна вартість: {totalCost}\n" +$"Середня кількість днів: {avgDays:F1}\n" +$"Відстань: від {minDistance} до {maxDistance} км"

);

SearchResults = finalResults;

this.DialogResult = DialogResult.OK;

this.Close();

}

}

}

**Delegates.cs:**

using System;

namespace LR1\_3

{

/// <summary>

/// 1) Власний делегат-стратегія для розрахунку вартості подорожі.

/// Дозволяє «підміняти» формулу без зміни класів подорожей.

/// </summary>

public delegate double CostStrategy(Podorozh trip);

/// <summary>

/// 2) Делегат для логування. Демонструємо мультикаст (ланцюжок викликів).

/// </summary>

public delegate void TripLogger(string message);

public static class TripDelegates

{

// Готові стратегії (через method group і лямбда-вираз):

public static readonly CostStrategy BaseCost = t => t.RozrakhVartist();

public static readonly CostStrategy WithAccommodation = t => t.TotalCostWithAccommodation();

public static readonly CostStrategy BusinessMarkup = t => t.RozrakhVartist() + 500;

}

}

**Accommodation.cs:**

using System;

namespace LR1\_3

{

public class Accommodation

{

public string Name { get; private set; }

public int Nights { get; private set; }

public double PricePerNight { get; private set; }

public Accommodation(string name, int nights, double pricePerNight)

{

if (string.IsNullOrWhiteSpace(name)) throw new ArgumentException("Name required");

if (nights < 1) throw new ArgumentOutOfRangeException(nameof(nights));

if (pricePerNight < 0) throw new ArgumentOutOfRangeException(nameof(pricePerNight));

Name = name;

Nights = nights;

PricePerNight = pricePerNight;

}

public double TotalPrice() => Nights \* PricePerNight;

public override string ToString()

{

return $"{Name} — {Nights} night(s), {TotalPrice():0.##} грн ({PricePerNight:0.##}/night)";

}

}

}

**BusinessPodorozh.cs:**

using LR1\_3;

public class BusinessPodorozh : Podorozh, IBusinessPodorozh

{

public string CompanyName { get; set; }

public string PurposeOfTrip { get; set; }

public BusinessPodorozh(string misto, double vidstan, int dni, string transport, string companyName, string purposeOfTrip)

: base(misto, vidstan, dni, transport)

{

CompanyName = companyName;

PurposeOfTrip = purposeOfTrip;

}

public double RozrakhVartistPodorozh()

{

double businessCosts = 500; // Фіксовані витрати для бізнес-подорожей

return RozrakhVartist() + businessCosts;

}

public override string Info()

{

return $"{base.Info()}, Компанія: {CompanyName}, Мета поїздки: {PurposeOfTrip}, Загальна вартість: {RozrakhVartistPodorozh()} грн";

}

// Метод для зміни типу туру

public void ZminytyTypTuru(string newType)

{

// Додайте логіку для зміни типу туру, якщо це потрібно

}

}

**ExtremePodorozh.cs:**

//ExtremePodorozh.cs

using LR1\_3;

public class ExtremePodorozh : Podorozh, IExtremePodorozh

{

public string DangerLevel { get; set; }

public ExtremePodorozh(string misto, double vidstan, int dni, string transport, string dangerLevel)

: base(misto, vidstan, dni, transport)

{

DangerLevel = dangerLevel;

}

public double RozrakhVartistPodorozh()

{

double dangerSurcharge = 1000; // Додаткові витрати на екстремальність подорожі

return RozrakhVartist() + dangerSurcharge;

}

public override string Info()

{

return $"{base.Info()}, Рівень небезпеки: {DangerLevel}, Загальна вартість: {RozrakhVartistPodorozh()} грн";

}

// Метод для зміни типу туру

public void ZminytyTypTuru(string newType)

{

// Додайте логіку для зміни типу туру, якщо це потрібно

}

}

**IBusinessPodorozh.cs:**

//IBusinessPodorozh.cs

namespace LR1\_3

{

public interface IBusinessPodorozh : IPodorozh

{

// Унікальні властивості

string CompanyName { get; set; }

string PurposeOfTrip { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**IExtremePodorozh.cs:**

//IExtremePodorozh.cs

namespace LR1\_3

{

public interface IExtremePodorozh : IPodorozh

{

// Унікальна властивість

string DangerLevel { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**IlongTermPodorozh.cs:**

//ILongTermPodorozh.cs

namespace LR1\_3

{

public interface ILongTermPodorozh : IPodorozh

{

// Унікальна властивість

int Months { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**IPodorozh.cs:**

//IPodorozh.cs

namespace LR1\_3

{

public interface IPodorozh

{

string Misto { get; set; }

double Vidstan { get; set; }

int Dni { get; set; }

string Transport { get; set; }

string Info(); //Метод для отримання інформації про подорож

double RozrakhVartist(); //Метод для розрахунку вартості подорожі

double SerednyaTsinaKm(); //Метод для розрахунку середньої ціни за км

}

}

**Itinerary.cs:**

// Itinerary.cs

using System;

using System.Collections.Generic;

using System.Linq;

using static LR1\_3.TravelAgency;

namespace LR1\_3

{

// Це об'єкт, який належить конкретній подорожі.

public class Itinerary

{

// Список житла — повна власність Itinerary (composition)

private readonly List<Accommodation> accommodations = new List<Accommodation>();

public IReadOnlyList<Accommodation> Accommodations => accommodations.AsReadOnly();

public double TotalAccommodationCost() => accommodations.Sum(a => a.TotalPrice());

public override string ToString()

{

if (!accommodations.Any()) return "No accommodations.";

return string.Join(Environment.NewLine, accommodations.Select(a => a.ToString()));

}

public void AddAccommodation(Accommodation acc)

{

if (acc == null) throw new ArgumentNullException(nameof(acc));

accommodations.Add(acc);

AppLog.Msg($"[Композиція] Додано житло: '{acc.Name}', ночей: {acc.Nights}, {acc.TotalPrice():0.##} грн");

}

public bool RemoveAccommodation(Accommodation acc)

{

bool ok = accommodations.Remove(acc);

if (ok) AppLog.Msg($"[Композиція] Видалено житло: '{acc?.Name}'");

return ok;

}

}

}

**IturystychnaPodorozh.cs:**

//ITurystychnaPodorozh.cs

namespace LR1\_3

{

public interface ITurystychnaPodorozh : IPodorozh

{

// Унікальні властивості туристичної подорожі

string TypTuru { get; set; }

double VartistProzhyvannya { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**LongTermPodorozh.cs:**

//LongTermPodorozh.cs

using LR1\_3;

public class LongTermPodorozh : Podorozh, ILongTermPodorozh

{

public int Months { get; set; }

public LongTermPodorozh(string misto, double vidstan, int dni, string transport, int months)

: base(misto, vidstan, dni, transport)

{

Months = months;

}

public double RozrakhVartistPodorozh()

{

double longTermSurcharge = 3000; // Додаткові витрати на довготривалі подорожі

return RozrakhVartist() + longTermSurcharge;

}

public override string Info()

{

return $"{base.Info()}, Тривалість: {Months} місяців, Загальна вартість: {RozrakhVartistPodorozh()} грн";

}

// Метод для зміни типу туру

public void ZminytyTypTuru(string newType)

{

// Додайте логіку для зміни типу туру, якщо це потрібно

}

}

**Podorozh.cs:**

//Podorozh.cs

using LR1\_3;

using System;

using static LR1\_3.TravelAgency;

public class Podorozh : IPodorozh

{

public string Misto { get; set; }

public double Vidstan { get; set; }

public int Dni { get; set; }

public string Transport { get; set; }

private readonly Itinerary itinerary = new Itinerary();

public Itinerary Itinerary => itinerary;

/\*3) Подія на стандартному делегаті Action<T> — «додали житло».

На неї зручно підписатися з форми лямбдою.\*/

public event Action<Accommodation> AccommodationAdded;

// Помічник для додавання житла

public void AddAccommodation(string hotelName, int nights, double pricePerNight)

{

var acc = new Accommodation(hotelName, nights, pricePerNight);

itinerary.AddAccommodation(acc);

// підняти подію

AccommodationAdded?.Invoke(acc);

}

public virtual string Info()

{

return $"Місто: {Misto}, Відстань: {Vidstan} км, Дні: {Dni}, Транспорт: {Transport}, Вартість: {RozrakhVartist():0.##} грн";

}

public Podorozh(string misto, double vidstan, int dni, string transport)

{

Misto = misto;

Vidstan = vidstan;

Dni = dni;

Transport = transport;

}

public double RozrakhVartist()

{

double tsinaZaKm = 5;

double tsinaZaDen = 200;

double v = (Vidstan \* tsinaZaKm) + (Dni \* tsinaZaDen);

return v;

}

public double SerednyaTsinaKm()

{

double s = RozrakhVartist() / Vidstan;

return s;

}

public double TotalCostWithAccommodation()

{

double baseCost = 0;

try

{

var m = this.GetType().GetMethod("RozrakhVartistPodorozh") ?? this.GetType().GetMethod("RozrakhVartist");

if (m != null) baseCost = Convert.ToDouble(m.Invoke(this, null));

}

catch { /\* ignore \*/ }

double accom = itinerary.TotalAccommodationCost();

double total = baseCost + accom;

AppLog.Msg($"[Розрахунок] Разом (з житлом) '{Misto}': база {baseCost:0.##} + житло {accom:0.##} = {total:0.##} грн");

return total;

}

// Оператори:

public static Podorozh operator +(Podorozh a, Podorozh b)

{

var res = new Podorozh(

a.Misto + "-" + b.Misto,

a.Vidstan + b.Vidstan,

a.Dni + b.Dni,

a.Transport

);

AppLog.Msg($"[Оператор '+'] Об'єднання '{a.Misto}' + '{b.Misto}' → '{res.Misto}' ({res.Vidstan} км, {res.Dni} дн.)");

return res;

}

public static bool operator >(Podorozh a, Podorozh b)

{

bool r = a.RozrakhVartist() > b.RozrakhVartist();

AppLog.Msg($"[Оператор '>'] '{a.Misto}' {(r ? ">" : "<=")} '{b.Misto}' за вартістю");

return r;

}

public static bool operator <(Podorozh a, Podorozh b)

{

bool r = a.RozrakhVartist() < b.RozrakhVartist();

AppLog.Msg($"[Оператор '<'] '{a.Misto}' {(r ? "<" : ">=")} '{b.Misto}' за вартістю");

return r;

}

public static bool operator ==(Podorozh a, Podorozh b)

{

if (ReferenceEquals(a, null) && ReferenceEquals(b, null)) { AppLog.Msg("[Оператор '=='] null == null → true"); return true; }

if (ReferenceEquals(a, null) || ReferenceEquals(b, null)) { AppLog.Msg("[Оператор '=='] один null → false"); return false; }

bool eq = a.Misto == b.Misto && a.Transport == b.Transport;

AppLog.Msg($"[Оператор '=='] '{a.Misto}/{a.Transport}' {(eq ? "==" : "!=")} '{b.Misto}/{b.Transport}'");

return eq;

}

public static bool operator !=(Podorozh a, Podorozh b)

{

bool ne = !(a == b);

// тут лог уже зроблено в '=='

return ne;

}

public static Podorozh operator ++(Podorozh a)

{

if (a == null) return a;

a.Dni += 1;

AppLog.Msg($"[Оператор '++'] '{a.Misto}': дні → {a.Dni}");

return a;

}

public static Podorozh operator --(Podorozh a)

{

if (a == null) return a;

int old = a.Dni;

a.Dni = a.Dni > 1 ? a.Dni - 1 : 1;

AppLog.Msg($"[Оператор '--'] '{a.Misto}': дні {old} → {a.Dni}");

return a;

}

public override int GetHashCode()

{

return (Misto + Transport).GetHashCode();

}

}

**Promt.cs:**

using System.Windows.Forms;

namespace LR1\_3

{

public static class Prompt

{

public static string ShowDialog(string text, string caption)

{

Form prompt = new Form()

{

Width = 420,

Height = 150,

FormBorderStyle = FormBorderStyle.FixedDialog,

Text = caption,

StartPosition = FormStartPosition.CenterParent,

MinimizeBox = false,

MaximizeBox = false

};

Label textLabel = new Label() { Left = 10, Top = 10, Width = 380, Text = text };

TextBox textBox = new TextBox() { Left = 10, Top = 35, Width = 380 };

Button confirmation = new Button() { Text = "OK", Left = 220, Width = 80, Top = 70, DialogResult = DialogResult.OK };

Button cancel = new Button() { Text = "Cancel", Left = 310, Width = 80, Top = 70, DialogResult = DialogResult.Cancel };

confirmation.Anchor = AnchorStyles.Bottom | AnchorStyles.Right;

cancel.Anchor = AnchorStyles.Bottom | AnchorStyles.Right;

prompt.Controls.Add(textBox);

prompt.Controls.Add(confirmation);

prompt.Controls.Add(cancel);

prompt.Controls.Add(textLabel);

prompt.AcceptButton = confirmation;

prompt.CancelButton = cancel;

var dialogResult = prompt.ShowDialog();

if (dialogResult == DialogResult.OK)

return textBox.Text;

else

return null;

}

}

}

**TravelAgency.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

namespace LR1\_3

{

// Аггрегація: агентство зберігає список подорожей, але подорож може існувати поза агентством.

public class TravelAgency

{

private readonly List<Podorozh> trips = new List<Podorozh>();

public string Name { get; set; }

public TravelAgency(string name)

{

Name = string.IsNullOrWhiteSpace(name) ? "Unnamed Agency" : name;

}

public IReadOnlyList<Podorozh> Trips => trips.AsReadOnly();

public void AddTrip(Podorozh trip)

{

if (trip == null) throw new ArgumentNullException(nameof(trip));

if (!trips.Contains(trip))

{

trips.Add(trip);

AppLog.Msg($"[Агрегація] Агенція '{Name}' отримала подорож: {trip.Misto} ({trip.Dni} дн., {trip.Transport})");

}

}

public bool RemoveTrip(Podorozh trip)

{

bool ok = trips.Remove(trip);

if (ok) AppLog.Msg($"[Агрегація] З агенції '{Name}' прибрано подорож: {trip?.Misto}");

return ok;

}

public double TotalAgencyRevenueEstimate(CostStrategy overrideStrategy = null)

{

var strategy = overrideStrategy ?? RevenueStrategy ?? TripDelegates.BaseCost;

double sum = 0;

foreach (var t in trips)

{

try { sum += strategy(t); }

catch

{

try

{

var m = t.GetType().GetMethod("RozrakhVartistPodorozh") ?? t.GetType().GetMethod("RozrakhVartist");

if (m != null) sum += Convert.ToDouble(m.Invoke(t, null));

}

catch { }

}

}

AppLog.Msg($"[Агрегація] Оцінка доходу агенції '{Name}': {sum:0.##} грн (за стратегією: {strategy.Method.Name})");

return sum;

}

public IEnumerable<Podorozh> GetTripsByCity(string city)

{

if (string.IsNullOrWhiteSpace(city)) return Enumerable.Empty<Podorozh>();

return trips.Where(t => t.Misto?.IndexOf(city, StringComparison.OrdinalIgnoreCase) >= 0);

}

public CostStrategy RevenueStrategy { get; set; } = TripDelegates.BaseCost;

public static class AppLog

{

public static System.Action<string> Write;

public static void Msg(string message)

=> Write?.Invoke(message);

}

}

}

**TurystychnaPodorozh.cs:**

//TurystychnaPodorozh.cs

using LR1\_3;

public class TurystychnaPodorozh : Podorozh, ITurystychnaPodorozh

{

public string TypTuru { get; set; }

public double VartistProzhyvannya { get; set; }

public TurystychnaPodorozh(string misto, double vidstan, int dni, string transport, string typTuru, double vartistProzhyvannya)

: base(misto, vidstan, dni, transport)

{

TypTuru = typTuru;

VartistProzhyvannya = vartistProzhyvannya;

}

public double RozrakhVartistPodorozh()

{

return RozrakhVartist() + VartistProzhyvannya;

}

public override string Info()

{

return $"{base.Info()}, Тип туру: {TypTuru}, Вартість проживання: {VartistProzhyvannya} грн, Загальна вартість: {RozrakhVartistPodorozh()} грн";

}

// Метод для зміни типу туру

public void ZminytyTypTuru(string newType)

{

TypTuru = newType;

}

}

**Результат програми:**

|  |  |
| --- | --- |
|  | Під час запуску програми бачимо внизу, що в нас є логування дій, тобто воно пише для користувача що додано і т.д. |
|  | Додали дані для праці з ними. Додамо агенцію для першої подорожі.  Порахуємо дохід усіх агенцій. |
|  | Рахує за поточною стратегією(підміняє формулу делегатом) та з бізнес націнкою(додаткова сума або процент). Додамо житло до подорожі. |
|  | Логування одразу нам висвітило розрахунок і підведено підсумок, це також делегат. |

**Висновок:** Під час виконання лабораторної роботи, я одержав практичні навички створення та роботи з делегатами.