Министерство образования Российской Федерации Пензенский государственный университет

Кафедра «Математическое обеспечение и применение ЭВМ»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «ООП» на тему «Поведенческие паттерны»

Выполнил:

студент группы 23ВП1

Рейимов М.

Принял:

к.т.н. Афонин А.Ю.

Пенза 2025

### Название

Поведенческие паттерны.

### Цель работы

Научиться использовать поведенческие паттерны.

### Лабораторное задание

Реализовать поведенческий паттерн программирования «Стратегия».

Предметная область – фирма грузоперевозок.

### Листинг

### Файл TransportCompany.cs:

using System;

using System.Collections.Generic;

using System.Drawing.Printing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab1

{

public interface IDeliveryType

{

string GetDeliveryType();

}

public class SeaDelivery : IDeliveryType

{

public string GetDeliveryType()

{

return "По воде";

}

}

public class RoadDelivery : IDeliveryType

{

public string GetDeliveryType()

{

return "По земле";

}

}

public class AirDelivery : IDeliveryType

{

public string GetDeliveryType()

{

return "По воздуху";

}

}

public interface IEfficiencyStrategy

{

float CalculateEfficiency(TransportCompany company);

string Description();

}

public class MassTimeEfficiency : IEfficiencyStrategy

{

public float CalculateEfficiency(TransportCompany company)

{

if (company.averageDeliveryTime == 0)

return 0;

return (float)Math.Round(company.transportedMass / company.averageDeliveryTime, 2);

}

public string Description()

{

return "По массе и времени";

}

}

public class AgePriceEfficiency : IEfficiencyStrategy

{

public float CalculateEfficiency(TransportCompany company)

{

int currentYear = DateTime.Now.Year;

int age = currentYear - company.yearFounded;

if (company.pricePerKilometer == 0)

return 0;

return (float)Math.Round(age / (float)company.pricePerKilometer, 2);

}

public string Description()

{

return "По возрасту и цене";

}

}

public class BalancedEfficiency : IEfficiencyStrategy

{

public float CalculateEfficiency(TransportCompany company)

{

float massEff = company.transportedMass / (company.averageDeliveryTime + 1);

float priceEff = 100f / (company.pricePerKilometer + 1);

return (float)Math.Round(0.6f \* massEff + 0.4f \* priceEff, 2);

}

public string Description()

{

return "Сбалансированная";

}

}

public abstract class TransportCompany

{

public string name { get; set; }

public int pricePerKilometer { get; set; }

public float averageDeliveryTime { get; set; }

public int yearFounded { get; set; }

public float transportedMass { get; set; }

public float rating { get; set; }

public string phoneNumber { get; set; }

public float efficiency { get; set; }

public IDeliveryType deliveryType { get; set; }

public IEfficiencyStrategy efficiencyStrategy { get; set; }

public static int countObj = 0;

public TransportCompany()

{

name = "";

pricePerKilometer = 0;

averageDeliveryTime = 0;

yearFounded = 2000;

transportedMass = 0;

rating = 0;

phoneNumber = "";

countObj++;

deliveryType = null;

efficiencyStrategy = null;

}

public TransportCompany(string name) : this()

{

this.name = name;

}

public TransportCompany(string name, int pricePerKilometer) : this(name)

{

this.pricePerKilometer = pricePerKilometer;

}

public TransportCompany(string name, int pricePerKilometer, float averageDeliveryTime,

int yearFounded, float transportedMass, float rating, string phoneNumber, IDeliveryType deliveryType) : this(name, pricePerKilometer)

{

this.averageDeliveryTime = averageDeliveryTime;

this.yearFounded = yearFounded;

this.transportedMass = transportedMass;

this.rating = rating;

this.phoneNumber = phoneNumber;

this.deliveryType = deliveryType;

this.efficiencyStrategy = new BalancedEfficiency();

}

public virtual string GetDeliveryInfo()

{

return deliveryType.GetDeliveryType();

}

public virtual void RecalculateEfficiency()

{

efficiency = efficiencyStrategy.CalculateEfficiency(this);

}

}

public class ExpressCompany : TransportCompany

{

public ExpressCompany(string name,

int pricePerKilometer,

float averageDeliveryTime,

int yearFounded,

float transportedMass,

float rating,

string phoneNumber,

IDeliveryType deliveryType) : base(name, pricePerKilometer, averageDeliveryTime, yearFounded, transportedMass, rating, phoneNumber, deliveryType) { }

public override string GetDeliveryInfo()

{

return base.GetDeliveryInfo();

}

}

}

Файл MyException.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab1

{

internal class MyException : Exception

{

public MyException(string message) : base(message) { }

}

}

Файл StackTransportCompany.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab2

{

internal class StackTransportCompany

{

private Stack<TransportCompany> transportCompanies;

public event Action<TransportCompany> StackAdded;

public event Action StackRemoved;

public StackTransportCompany()

{

this.transportCompanies = new Stack<TransportCompany>();

}

public Stack<TransportCompany> GetTransportCompanies()

{

return transportCompanies;

}

public void AddCompany(TransportCompany company)

{

transportCompanies.Push(company);

StackAdded?.Invoke(company);

}

public void DeleteCompany()

{

if (transportCompanies.Count == 0)

throw new MyException("Стек пуст");

TransportCompany top = transportCompanies.Pop();

StackRemoved?.Invoke();

}

}

}

Файл StackListener.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace Lab2

{

public class StackListener

{

private DataGridView dataGridView;

private TextBox objCount;

public StackListener(StackTransportCompany stack, DataGridView dataGridView, TextBox objCount)

{

this.dataGridView = dataGridView;

this.objCount = objCount;

stack.StackAdded += (TransportCompany company) =>

{

int rowIndex = dataGridView.Rows.Add();

dataGridView.Rows[rowIndex].Cells[0].Value = company.name;

dataGridView.Rows[rowIndex].Cells[1].Value = company.price.ToString();

dataGridView.Rows[rowIndex].Cells[2].Value = company.transportedMass.ToString();

dataGridView.Rows[rowIndex].Cells[3].Value = company.ratingCalculationStrategy.TypeOfRating();

dataGridView.Rows[rowIndex].Cells[4].Value = company.rating.ToString();

dataGridView.Rows[rowIndex].Cells[5].Value = company.completedOrders.ToString();

dataGridView.Rows[rowIndex].Cells[6].Value = company.phoneNumber;

dataGridView.Rows[rowIndex].Cells[7].Value = company.email;

dataGridView.Rows[rowIndex].Cells[8].Value = company.DoWork();

objCount.Text = TransportCompany.countObj.ToString();

};

stack.StackRemoved += () =>

{

if (dataGridView.Rows.Count > 0)

dataGridView.Rows.RemoveAt(dataGridView.Rows.Count - 1); ;

objCount.Text = TransportCompany.countObj.ToString();

};

}

}

}

Файл Form1.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Policy;

using System.Text;

using System.Threading.Tasks;

using System.Text.RegularExpressions;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using Lab1;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;

using System.Diagnostics;

using System.Xml.Linq;

namespace Lab2

{

public partial class Form1 : Form

{

private StackTransportCompany companies;

private StackListener stackListener;

public Form1()

{

InitializeComponent();

MessageBox.Show("(название)\n" +

"(ваши фамилии) 23ВП1\nВариант 8. Фирма грузоперевозок", "Лабораторная работа 5");

companies = new StackTransportCompany();

stackListener = new StackListener(companies, dataGridView1, objCount);

}

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

{

try

{

if (string.IsNullOrWhiteSpace(nameInput.Text))

throw new MyException("Фирма должна иметь название");

if (string.IsNullOrWhiteSpace(phonenumberInput.Text))

throw new MyException("Фирма должна иметь номер");

if (!Regex.IsMatch(phonenumberInput.Text.Trim(), @"^\d{11}$"))

throw new MyException("Номер должен состоять из 11 цифр и не содержать буквы или символы");

if (deliveryMethod.SelectedIndex == -1)

throw new MyException("Выберите метод доставки");

string type = deliveryMethod.SelectedItem.ToString();

IDeliveryType delivery = null;

switch (type)

{

case "По воздуху":

delivery = new AirDelivery();

break;

case "По воде":

delivery = new SeaDelivery();

break;

case "По земле":

delivery = new RoadDelivery();

break;

}

TransportCompany firm = new ExpressCompany(nameInput.Text,

(int)priceInput.Value,

(float)avgtimeInput.Value,

(int)yearInput.Value,

(float)massInput.Value,

(float)ratingInput.Value,

phonenumberInput.Text, delivery);

firm.RecalculateEfficiency();

objCount.Text = TransportCompany.countObj.ToString();

companies.AddCompany(firm);

}

catch (MyException ex)

{

MessageBox.Show(ex.Message, "Ошибка");

}

}

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

{

try

{

TransportCompany.countObj--;

companies.DeleteCompany();

}

catch (MyException ex)

{

MessageBox.Show(ex.Message, "Ошибка");

}

}

private void ShowAll()

{

dataGridView1.Rows.Clear();

foreach (var company in companies.GetTransportCompanies())

{

int rowIndex = dataGridView1.Rows.Add();

dataGridView1.Rows[rowIndex].Cells[0].Value = company.name;

dataGridView1.Rows[rowIndex].Cells[1].Value = company.pricePerKilometer.ToString();

dataGridView1.Rows[rowIndex].Cells[2].Value = company.averageDeliveryTime.ToString();

dataGridView1.Rows[rowIndex].Cells[3].Value = company.yearFounded.ToString();

dataGridView1.Rows[rowIndex].Cells[4].Value = company.transportedMass.ToString();

dataGridView1.Rows[rowIndex].Cells[5].Value = company.rating.ToString();

dataGridView1.Rows[rowIndex].Cells[6].Value = company.phoneNumber;

dataGridView1.Rows[rowIndex].Cells[7].Value = company.GetDeliveryInfo();

dataGridView1.Rows[rowIndex].Cells[8].Value = company.efficiencyStrategy.Description();

dataGridView1.Rows[rowIndex].Cells[9].Value = company.efficiency;

}

}

private void save\_button\_Click(object sender, EventArgs e)

{

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

var transportCompanies = companies.GetTransportCompanies().Reverse().ToList();

if (i >= transportCompanies.Count)

break;

TransportCompany company = transportCompanies[i];

string selectedStrategy = dataGridView1.Rows[i].Cells[8].Value.ToString();

if (selectedStrategy == "По массе и времени")

company.efficiencyStrategy = new MassTimeEfficiency();

else if (selectedStrategy == "По возрасту и цене")

company.efficiencyStrategy = new AgePriceEfficiency();

else if (selectedStrategy == "Сбалансированная")

company.efficiencyStrategy = new BalancedEfficiency();

company.RecalculateEfficiency();

string selectedMethod = dataGridView1.Rows[i].Cells[7].Value.ToString();

if (selectedMethod == new RoadDelivery().GetDeliveryType())

company.deliveryType = new RoadDelivery();

else if (selectedMethod == new AirDelivery().GetDeliveryType())

company.deliveryType = new AirDelivery();

else if (selectedMethod == new SeaDelivery().GetDeliveryType())

company.deliveryType = new SeaDelivery();

}

ShowAll();

MessageBox.Show("Все изменения сохранены!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

### Пояснительный текст к программе

Абстрактный класс TransportCompany представляет общие характеристики всех транспортных компаний и содержит поля deliveryType (тип доставки), efficiencyStrategy (стратегия вычисления эффективности компании), pricePerKilometer (цена за километр), transportedMass (масса перевезенных грузов), name (наименование компании), rating (рейтинг), averageDeliveryTime (среднее время доставки), phoneNumber (номер телефона компании), yearFounded (год основания компании).

Интерфейс IDeliveryType определяет общий контракт для всех способов доставки. У него есть единственный метод GetDeliveryType().

В коде реализованы три класса, наследующие этот интерфейс:

* SeaDelivery – доставка по морю.
* RoadDelivery – доставка по суше.
* AirDelivery – доставка по воздуху.

Интерфейс IEfficiencyStrategy определяет контракт для стратегий, а три класса (MassTimeEfficiency, AgePriceEfficiency, BalancedEfficiency) реализуют разные способы расчета рейтинга:

* MassTimeEfficiency — эффективности по времени и массе.
* AgePriceEfficiency — эффективность по возрасту и массе.
* BalancedEfficiency — сбалансированная эффективность.

Каждая стратегия вычисляет рейтинг по своему методу и возвращает строку с типом рейтинга.

Класс ExpressCompany - это конкретная реализация транспортной компании, которая может использовать один из типов доставки.

Класс MyException – это класс пользовательских исключений. Является наследником базового класса Exception и содержит информацию об ошибке, сгенерированной во время выполнения программы.

Класс StackTransportCompany является классом, хранящим стек из транспортных компаний. Осуществляет добавление элемента в стек и удаление элемента из стека.

Класс StackListener – класс-слушатель, который обрабатывает события.

### Результаты работы программы

Результаты работы программы показаны на рисунках 1-3.

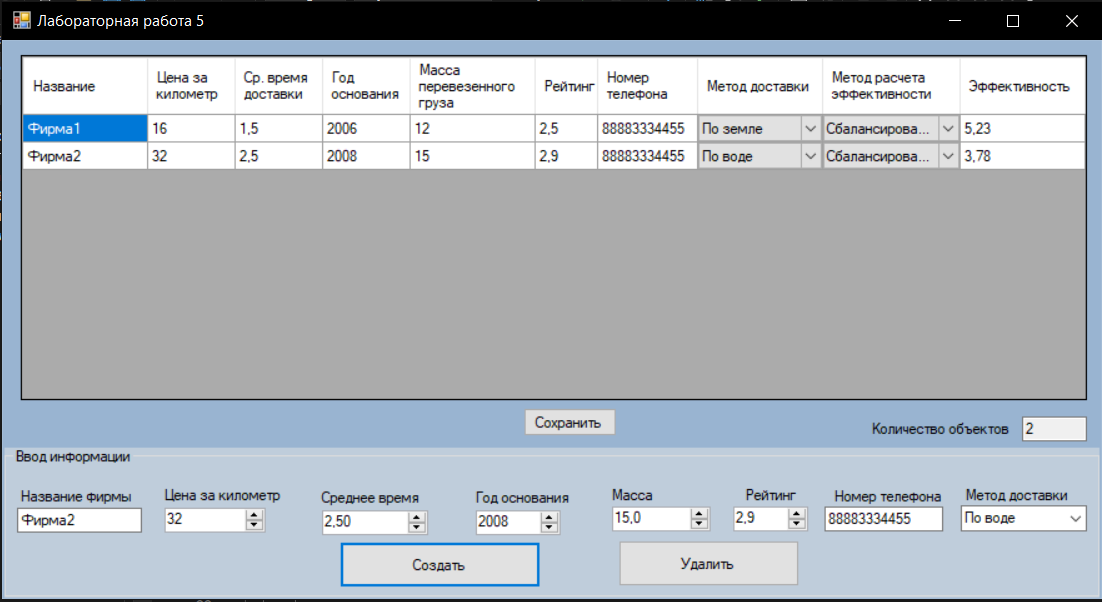


Рисунок 1 – Создание объекта, добавление в стек и вывод информации

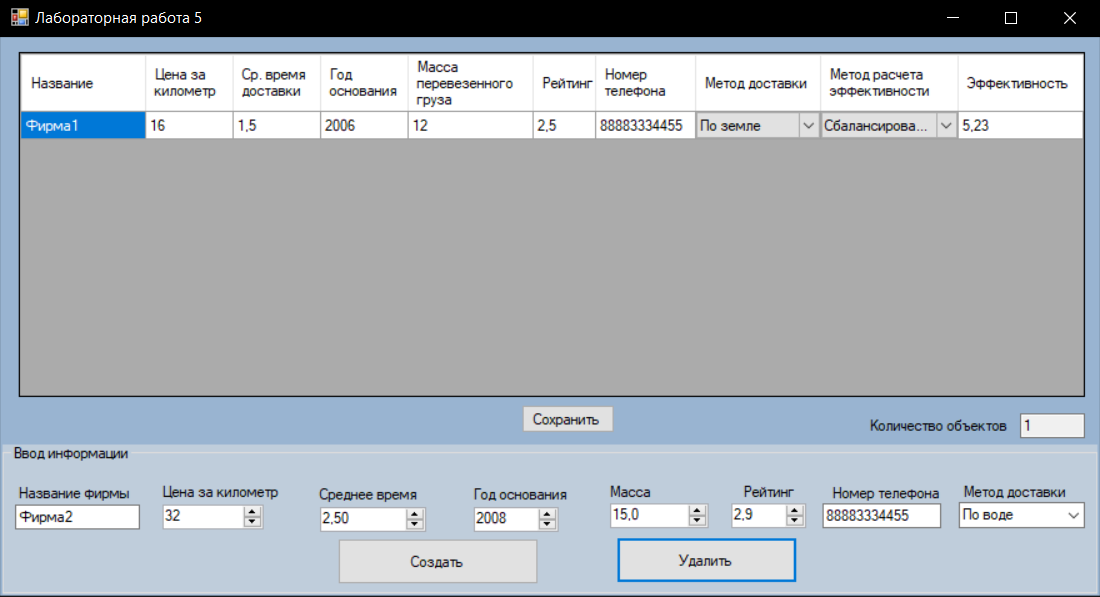


Рисунок 2 – Удаление фирмы из стека и вывод информации

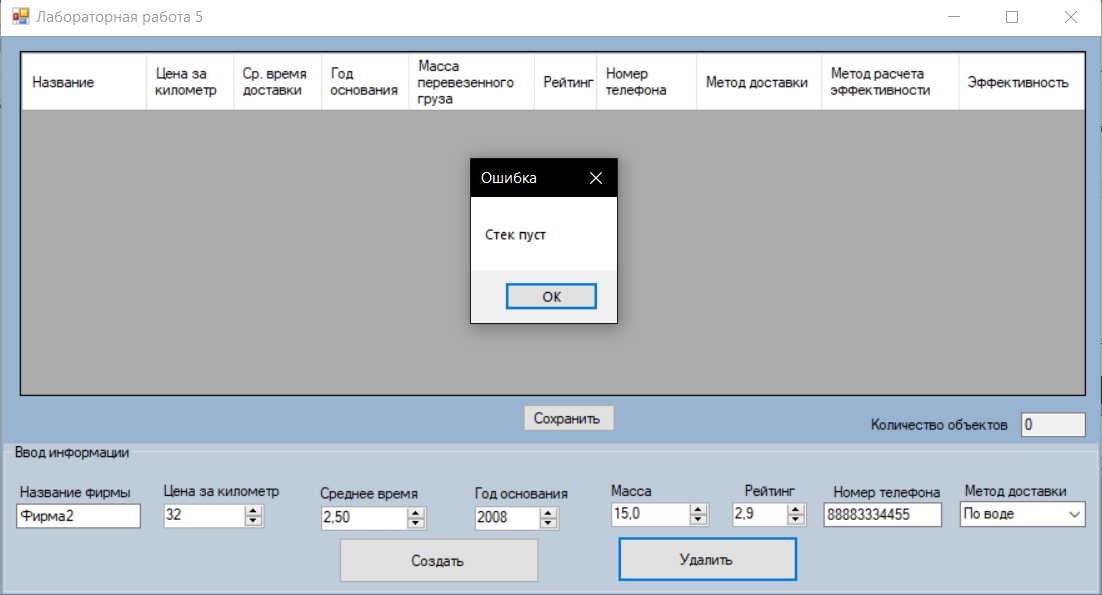


Рисунок 3 – Попытка удаления элемента из пустого стека

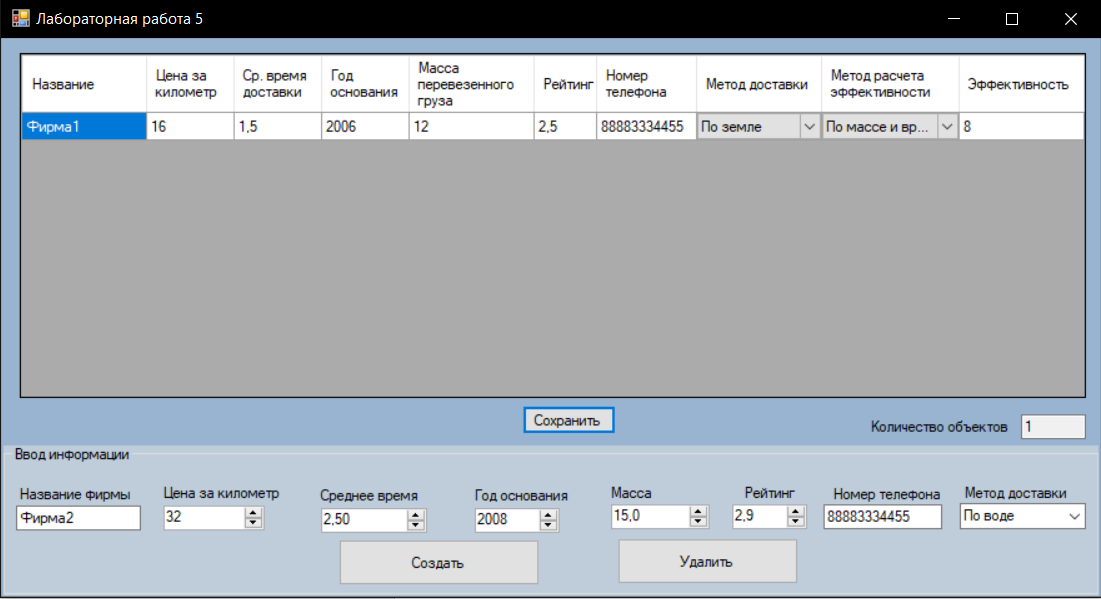


Рисунок 4 – Попытка изменения стратегии вычисления эффективности и сохранения изменений

### Диаграмма вариантов использования

### C:\Users\Влад\Downloads\2025-05-10_17-40-48.png

Рисунок 4 - Диаграмма вариантов использования

### Выводы

В ходе выполнения лабораторной работы был реализован поведенческий паттерн «Стратегия». «Стратегия» - поведенческий шаблон проектирования, который определяет набор алгоритмов, инкапсулирует каждый из них и обеспечивает их взаимозаменяемость. В данной лабораторной работе этот паттерн позволил рассчитывать разными способами рейтинг компании, опираясь на различные факторы.