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»

Выполнили:

студенты группы 23ВП2 Соколов О.А.

Муравьев В.С.

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;

using System.Collections.Generic;

using System.Drawing.Printing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab2

{

// Паттерн Мост

public interface ITransportMethod

{

string Deliver();

}

public class ShipTransport: ITransportMethod

{

public string Deliver()

{

return "Перевозка кораблем";

}

}

public class TrackTransport: ITransportMethod

{

public string Deliver()

{

return "Перевозка грузовиком";

}

}

public class AirTransport : ITransportMethod

{

public string Deliver()

{

return "Перевозка самолетом";

}

}

// Паттерн Стратегия

public interface IRatingCalculationStrategy

{

float CalculateRating(TransportCompany company);

string TypeOfRating();

}

public class OrderBasedRating : IRatingCalculationStrategy

{

public float CalculateRating(TransportCompany company)

{

if (company.completedOrders == 0)

return 0;

return (float)Math.Round(10 \* (1 - Math.Exp(-company.completedOrders / 50.0)), 2);

}

public string TypeOfRating()

{

return "По заказам";

}

}

public class PriceBasedRating : IRatingCalculationStrategy

{

public float CalculateRating(TransportCompany company)

{

if (company.price == 0)

return 0;

float normalizedPrice = company.price / 1900.0f;

return (float)Math.Round(10 \* (1 - normalizedPrice) \* (1 - normalizedPrice), 2);

}

public string TypeOfRating()

{

return "По цене";

}

}

public class MixedRating : IRatingCalculationStrategy

{

public float CalculateRating(TransportCompany company)

{

float orderRating = new OrderBasedRating().CalculateRating(company);

float priceRating = new PriceBasedRating().CalculateRating(company);

return (float)Math.Round(0.6 \* orderRating + 0.4 \* priceRating, 2);

}

public string TypeOfRating()

{

return "Комбин.";

}

}

public abstract class TransportCompany : ICloneable

{

public ITransportMethod deliverMethod { get; set; }

public IRatingCalculationStrategy ratingCalculationStrategy { get; set; }

public int price { get; set; }

public float transportedMass { get; set; }

public string name { get; set; }

public float rating { get; set; }

public int completedOrders { get; set; }

public string phoneNumber { get; set; }

public string email { get; set; }

public static int countObj = 0;

public TransportCompany()

{

price = 0;

transportedMass = 0;

name = "";

rating = 0;

completedOrders = 0;

phoneNumber = "";

email = "";

countObj++;

deliverMethod = null;

}

public TransportCompany(int price) : this()

{

this.price = price;

}

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

{

this.name = name;

}

public TransportCompany(int price,

float transportedMass,

string name,

int completedOrders,

string phoneNumber,

string email,

ITransportMethod transportMethod) : this(price, name)

{

this.transportedMass = transportedMass;

this.completedOrders = completedOrders;

this.phoneNumber = phoneNumber;

this.email = email;

this.deliverMethod = transportMethod;

this.ratingCalculationStrategy = new MixedRating();

RecalculateRating();

}

public object Clone()

{

countObj++;

return this.MemberwiseClone();

}

public virtual void RecalculateRating()

{

rating = ratingCalculationStrategy.CalculateRating(this);

}

public override string ToString()

{

return "\tТрансопртная компания" + "\nНазвание: " + name + "\nЦена грузоперевозки: " + price + "\nМасса перевезенных грузов: " + transportedMass + "\nРейтинг: " + rating + "\nКоличество выполненных заказов: " + completedOrders + "\nНомер компании: " + phoneNumber + "\nПочта компании: " + email;

}

public string PrintName()

{

if (string.IsNullOrEmpty(name))

return "У фирмы нет названия";

return "Название фирмы: " + name;

}

public string PriceToHex()

{

return Convert.ToString(price, 16);

}

public virtual string DoWork()

{

return deliverMethod.Deliver();

}

}

public class LogisticCompany : TransportCompany

{

public LogisticCompany() {}

public LogisticCompany(int price) : base(price) {}

public LogisticCompany(int price,

float transportedMass,

string name,

int completedOrders,

string phoneNumber,

string email,

ITransportMethod transportMethod) : base(price, transportedMass, name, completedOrders,

phoneNumber, email, transportMethod) {}

public override string DoWork()

{

return deliverMethod.Deliver();

}

public override void RecalculateRating()

{

rating = ratingCalculationStrategy.CalculateRating(this);

}

}

}

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 System.Runtime.InteropServices;

using System.Collections;

namespace Lab2

{

public partial class Form1 : Form

{

private TransportCompany prototype = null;

private StackTransportCompany companies;

private StackListener stackListener;

public Form1()

{

InitializeComponent();

companies = new StackTransportCompany();

transportMethodColumn.DataSource = new List<string>

{

new TrackTransport().Deliver(),

new ShipTransport().Deliver(),

new AirTransport().Deliver()

};

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

}

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

{

try

{

if (string.IsNullOrWhiteSpace(name.Text))

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

if (string.IsNullOrWhiteSpace(phoneNumber.Text))

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

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

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

if (!Regex.IsMatch(email.Text.Trim(), @"^[a-zA-Z0-9\_]+@mail\.ru$"))

throw new MyException("Неверный формат почты");

TransportCompany firm;

if (prototype == null)

{

prototype = new LogisticCompany(

(int)price.Value,

(float)transportedMass.Value,

name.Text,

(int)completedOrders.Value,

phoneNumber.Text,

email.Text,

new TrackTransport()

);

}

firm = (LogisticCompany)prototype.Clone();

firm.price = (int)price.Value;

firm.transportedMass = (float)transportedMass.Value;

firm.name = name.Text;

firm.completedOrders = (int)completedOrders.Value;

firm.phoneNumber = phoneNumber.Text;

firm.email = email.Text;

firm.RecalculateRating();

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\_Click(object sender, EventArgs e)

{

ShowAll();

}

private void ShowAll()

{

dataGridView1.Rows.Clear();

foreach (var company in companies.GetTransportCompanies().Reverse())

{

int rowIndex = dataGridView1.Rows.Add();

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

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

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

dataGridView1.Rows[rowIndex].Cells[3].Value = company.ratingCalculationStrategy.TypeOfRating();

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

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

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

dataGridView1.Rows[rowIndex].Cells[7].Value = company.email;

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

}

}

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[3].Value.ToString();

if (selectedStrategy == "По заказам")

company.ratingCalculationStrategy = new OrderBasedRating();

else if (selectedStrategy == "По цене")

company.ratingCalculationStrategy = new PriceBasedRating();

else if (selectedStrategy == "Комбин.")

company.ratingCalculationStrategy = new MixedRating();

company.RecalculateRating();

dataGridView1.Rows[i].Cells[4].Value = company.rating;

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

if (selectedMethod == new TrackTransport().Deliver())

company.deliverMethod = new TrackTransport();

else if (selectedMethod == new ShipTransport().Deliver())

company.deliverMethod = new ShipTransport();

else if (selectedMethod == new AirTransport().Deliver())

company.deliverMethod = new AirTransport();

}

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

}

private void generate\_100000\_objects\_Click(object sender, EventArgs e)

{

int elementCount = 5000;

Random rand = new Random();

ITransportMethod curMethod = null;

switch (rand.Next(3))

{

case 0:

curMethod = new ShipTransport();

break;

case 1:

curMethod = new TrackTransport();

break;

case 2:

curMethod = new AirTransport();

break;

}

TransportCompany baseCompany = new LogisticCompany(

rand.Next(100, 1000),

(float)rand.NextDouble() \* 100,

"BaseCompany",

rand.Next(0, 1000),

"89990000000",

"basecompany@mail.ru",

curMethod

);

// Для StackTransportCompany

int start = Environment.TickCount;

for (int i = 0; i < elementCount; i++)

{

TransportCompany clonedCompany = (TransportCompany)baseCompany.Clone();

clonedCompany.transportedMass = (float)rand.NextDouble() \* 100;

clonedCompany.name = "Company" + i;

clonedCompany.price = rand.Next(1000, 10000);

clonedCompany.phoneNumber = "8999" + rand.Next(1000000, 9999999);

clonedCompany.email = "company" + i + "@mail.ru";

companies.AddCompany(clonedCompany);

}

int insertionTimeCollection = Environment.TickCount - start;

start = Environment.TickCount;

foreach (var company in companies.GetTransportCompanies())

{

var temp = company.price;

}

int sequentialReadTimeCollection = Environment.TickCount - start;

var allCompanies = companies.GetTransportCompanies().ToArray();

start = Environment.TickCount;

for (int i = 0; i < elementCount; i++)

{

var temp = allCompanies[rand.Next(0, elementCount)].price;

}

int randomReadTimeCollection = Environment.TickCount - start;

// Для Array

TransportCompany[] companyArray = new TransportCompany[elementCount];

start = Environment.TickCount;

for (int i = 0; i < elementCount; i++)

{

companyArray[i] = new LogisticCompany(

rand.Next(100, 1000),

(float)rand.NextDouble() \* 100,

"Company" + i,

rand.Next(0, 1000),

"8999" + rand.Next(1000000, 9999999),

"company" + i + "@mail.ru",

curMethod

);

}

int insertionTimeArray = Environment.TickCount - start;

start = Environment.TickCount;

for (int i = 0; i < elementCount; i++)

{

var temp = companyArray[i].price;

}

int sequentialReadTimeArray = Environment.TickCount - start;

start = Environment.TickCount;

for (int i = 0; i < elementCount; i++)

{

var temp = companyArray[rand.Next(0, elementCount)].price;

}

int randomReadTimeArray = Environment.TickCount - start;

dataGridView1.Rows.Clear();

string results = "Результаты для StackTransportCompany:\n" +

$"Вставка: {insertionTimeCollection} мс\n" +

$"Последоват. выборка: {sequentialReadTimeCollection} мс\n" +

$"Случайная выборка: {randomReadTimeCollection} мс\n\n" +

"Результаты для Array:\n" +

$"Вставка: {insertionTimeArray} мс\n" +

$"Последоват. выборка: {sequentialReadTimeArray} мс\n" +

$"Случайная выборка: {randomReadTimeArray} мс";

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

MessageBox.Show(results, "Результаты тестирования", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

### Пояснительный текст к программе

Абстрактный класс TransportCompany представляет общие характеристики всех транспортных компаний и содержит поля deliverMethod (способ перевозки), ratingCalculationStrategy (стратегия вычисления рейтинга), price (цена за грузоперевозку), transportedMass (масса перевезенных грузов), name (наименование компании), rating (рейтинг), completedOrders (количество выполненных заказов), phoneNumber (номер телефона компании), email (электронная почта компании).

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

В коде реализованы три класса, наследующие этот интерфейс:

* ShipTransport – доставка кораблем.
* TrackTransport – доставка грузовиком.
* AirTransport – доставка самолетом.

Интерфейс IRatingCalculationStrategy определяет контракт для стратегий, а три класса (OrderBasedRating, PriceBasedRating, MixedRating) реализуют разные способы расчета рейтинга:

* OrderBasedRating — рейтинг по количеству выполненных заказов.
* PriceBasedRating — рейтинг по цене за перевозку.
* MixedRating — комбинированный рейтинг, учитывающий оба фактора.

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

Класс LogisticCompany - это конкретная реализация транспортной компании, которая может использовать один из способов 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.

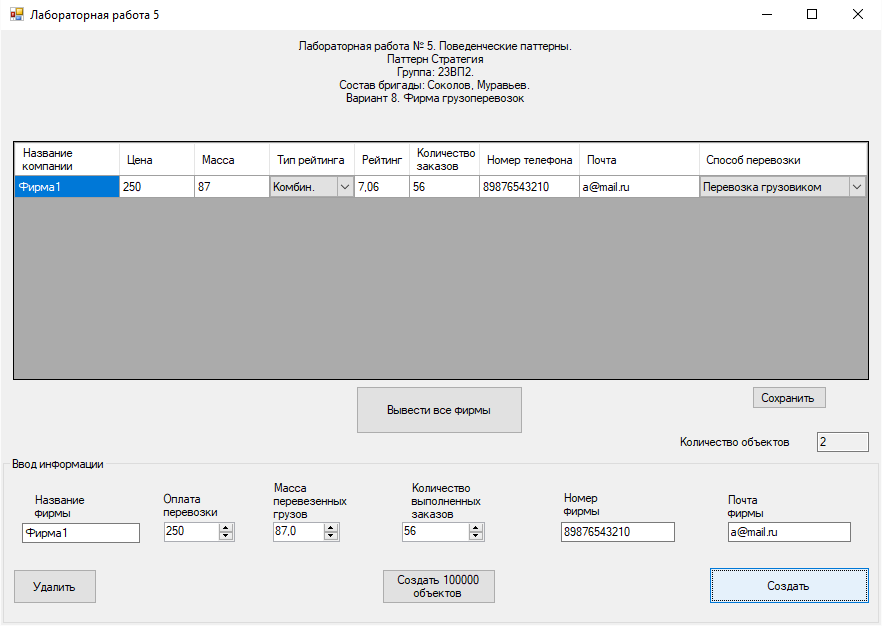


Рисунок 1 – Создание объекта, добавление в стек и вывод информации

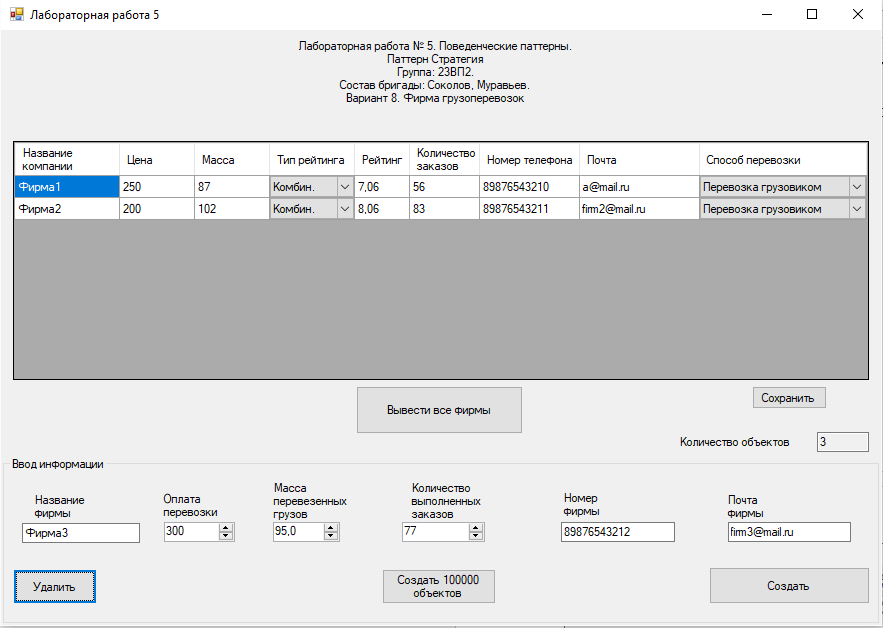


Рисунок 2 – Удаление фирмы из стека и вывод информации

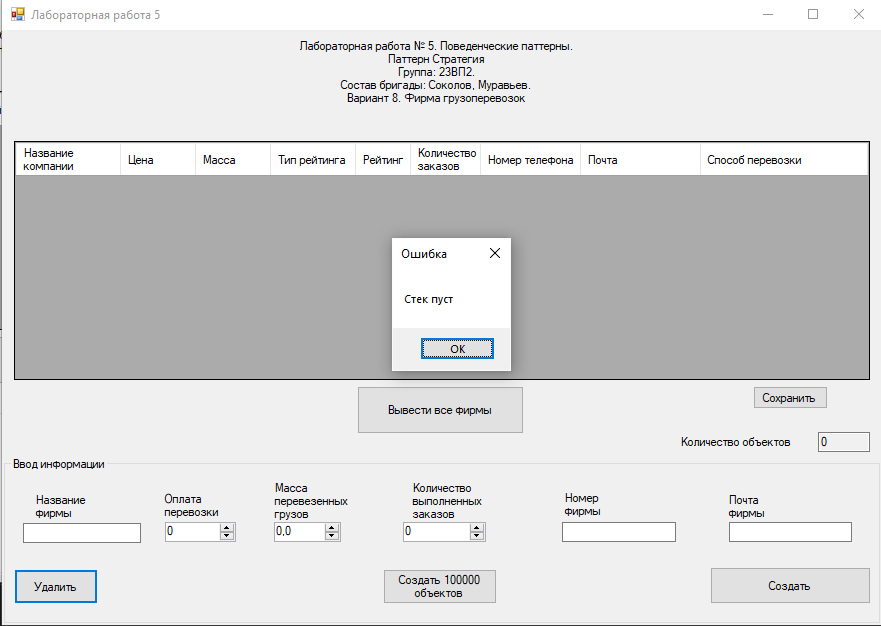


Рисунок 3 – Попытка удаления элемента из пустого стека

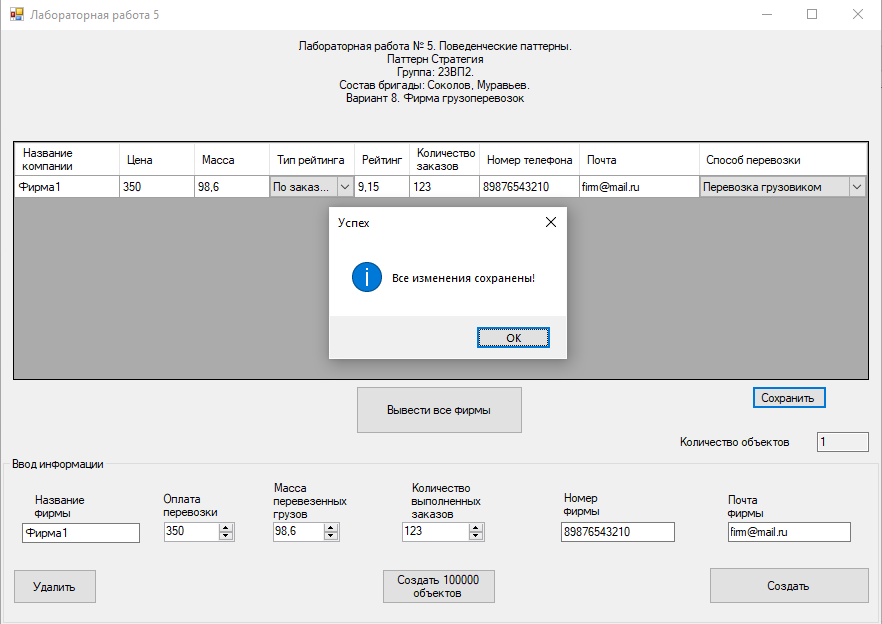


Рисунок 4 – Попытка изменения стратегии вычисления рейтинга и сохранения изменений

### Диаграмма вариантов использования

### C:\Users\Влад\Downloads\2025-03-31_19-50-47.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 расчитывать разными способами рейтинг компании, опираясь на различные факторы.