Министерство образования Российской Федерации Пензенский государственный университет

Кафедра «Математическое обеспечение и применение ЭВМ»

**ОТЧЕТ**

по лабораторной работе №8

по курсу «ООП» на тему «Паттерн MVP»

Выполнили:

студенты группы 23ВП2 Соколов О.А.

Муравьев В.С.

Принял:

к.т.н. Афонин А.Ю.

Пенза 2025

### Название

Паттерн MVP.

### Цель работы

Научиться использовать паттерн MVP.

### Лабораторное задание

Реализовать паттерн программирования MVP.

Предметная область – фирма грузоперевозок.

### Листинг

### Файл View1.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 View1 : Form, IView

{

public event Add AddClicked;

public event Remove RemoveClicked;

public event Generate\_100000\_Oblects Generate\_100000\_objs\_clicked;

public event ShowAll ShowAllClicked;

public event SaveChanges SaveChangesClicked;

public string NameCompany => name.Text;

public int Price => (int)price.Value;

public float TransportedMass => (float)transportedMass.Value;

public int CompletedOrders => (int)completedOrders.Value;

public string PhoneNumber => phoneNumber.Text;

public string Email => email.Text;

public View1()

{

InitializeComponent();

var presenter = new Presenter(dataGridView1, objCount, this);

}

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

{

AddClicked.Invoke(Price, TransportedMass, NameCompany, CompletedOrders, PhoneNumber, Email);

}

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

{

RemoveClicked.Invoke();

TransportCompany.countObj--;

}

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

{

ShowAllClicked.Invoke(dataGridView1);

}

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

{

SaveChangesClicked.Invoke(dataGridView1);

}

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

{

Generate\_100000\_objs\_clicked.Invoke(dataGridView1, objCount);

}

}

}

Файл Model.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab2

{

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";

}

}

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;

return (float)Math.Round(10 \* (1 - company.price / 2100.0f) \* (1 - company.price / 2100.0f), 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);

}

}

public class StackTransportCompany

{

private Stack<TransportCompany> transportCompanies;

public delegate void AddToStack(TransportCompany company);

public delegate void RemoveFromStack();

public event AddToStack StackAdded;

public event RemoveFromStack 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();

}

}

public class MyException : Exception

{

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

}

}

Файл Presenter.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab2

{

public class Presenter

{

private IView view;

private StackTransportCompany companies;

private TransportCompany prototype;

private StackListener listener;

public Presenter(DataGridView dataGridView, TextBox objCount, IView view)

{

companies = new StackTransportCompany();

listener = new StackListener(companies, dataGridView, objCount);

this.view = view;

this.view.AddClicked += CreateCompany;

this.view.RemoveClicked += DeleteCompany;

this.view.ShowAllClicked += ShowAll;

this.view.SaveChangesClicked += SaveChanges;

this.view.Generate\_100000\_objs\_clicked += Generate10000Objects;

}

public void CreateCompany(int price, float transportedMass, string name, int completedOrders, string phoneNumber, string email)

{

if (string.IsNullOrWhiteSpace(name))

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

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

throw new MyException("Номер должен состоять из 11 цифр");

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

throw new MyException("Неверный формат почты");

if (prototype == null)

{

prototype = new LogisticCompany(

price,

transportedMass,

name,

completedOrders,

phoneNumber,

email,

new TrackTransport()

);

}

TransportCompany firm = (TransportCompany)prototype.Clone();

firm.price = price;

firm.transportedMass = transportedMass;

firm.name = name;

firm.completedOrders = completedOrders;

firm.phoneNumber = phoneNumber;

firm.email = email;

firm.RecalculateRating();

companies.AddCompany(firm);

}

public void DeleteCompany()

{

try

{

companies.DeleteCompany();

}

catch (MyException ex)

{

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

}

}

public void ShowAll(DataGridView dataGridView)

{

dataGridView.Rows.Clear();

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

{

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();

}

}

public void SaveChanges(DataGridView dataGridView)

{

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

{

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

if (i >= transportCompanies.Count)

break;

TransportCompany company = transportCompanies[i];

string selectedStrategy = dataGridView.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();

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

string selectedMethod = dataGridView.Rows[i].Cells[8].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);

}

public void Generate10000Objects(DataGridView dataGridView, TextBox objCount)

{

int elementCount = 10000;

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, 2000),

"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(100, 1000);

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

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

clonedCompany.RecalculateRating();

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;

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, 2000),

"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;

dataGridView.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);

}

}

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();

};

}

}

}

### Пояснительный текст к программе

В данной программе реализован паттерн **MVP** (Model-View-Presenter), который помогает разделить логику приложения на три части: модель, представление и представителя.

* **Model.cs** — это часть модели, которая отвечает за данные и бизнес-логику. В этом файле описаны классы для транспортных компаний, включая их свойства (например, название, цена, масса груза, рейтинг и т. д.), а также методы для расчёта рейтинга и работы с различными способами доставки (с помощью паттернов "Мост" и "Стратегия"). Классы, такие как TransportCompany, LogisticCompany и другие, содержат логику для создания и работы с объектами транспортных компаний, а также для вычисления и хранения их данных.
* **View1.cs** — это представление, которое отвечает за отображение данных пользователю и взаимодействие с ним. В этом файле находятся элементы формы Windows (например, DataGridView, кнопки и текстовые поля), которые отображают информацию о транспортных компаниях. Также здесь находятся обработчики событий, которые реагируют на действия пользователя, такие как создание, удаление и сохранение данных.
* **Presenter.cs**. Представитель — это связующее звено между моделью и представлением. Он содержит бизнес-логику и управляет взаимодействием между View и Model.  
  В Presenter.cs реализована подписка на события представления (AddClicked, RemoveClicked, SaveChangesClicked и т. д.), и в ответ на них вызываются методы модели. Интерфейс IView представляет собой абстракцию пользовательского интерфейса и необходим для реализации паттерна MVP. Он обеспечивает связь между Presenter и конкретным представлением (View1.cs), позволяя Presenter взаимодействовать с представлением через четко определённые события и свойства. Это делает код более гибким, модульным и удобным для тестирования. В интерфейсе определены события (AddClicked, RemoveClicked, ShowAllClicked и др.) и свойства, необходимые для получения данных от пользователя.

**Принцип работы:**

* **Модель** (Model.cs) управляет данными (например, транспортными компаниями) и бизнес-логикой (расчёт рейтинга, выбор метода доставки и т. д.).
* **Представление** (View1.cs) отображает эти данные на экране и получает ввод от пользователя (например, имя компании или её цену).
* **Представитель** (Presenter.cs) слушает события представления, обрабатывает их, взаимодействует с моделью и обновляет представление в ответ на изменения.

### Результаты работы программы

Результаты работы программы показаны на рисунках 1-3.

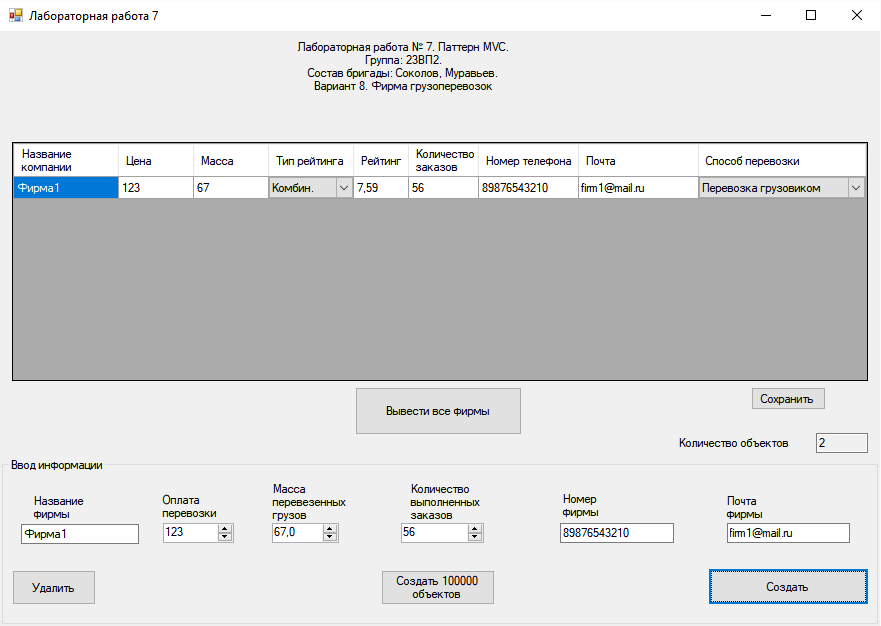


Рисунок 1 – Создание объекта, добавление в стек и вывод информации

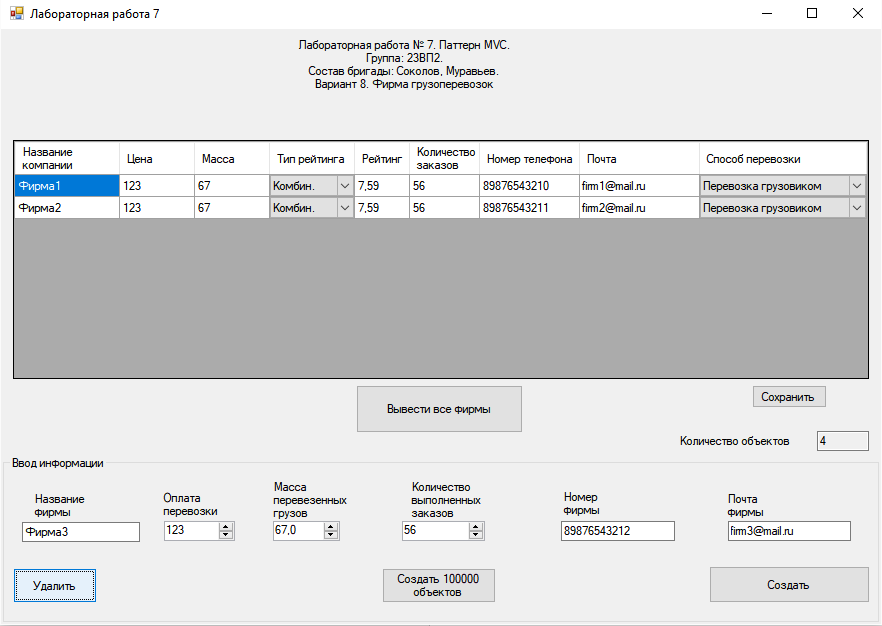


Рисунок 2 – Удаление фирмы из стека и вывод информации

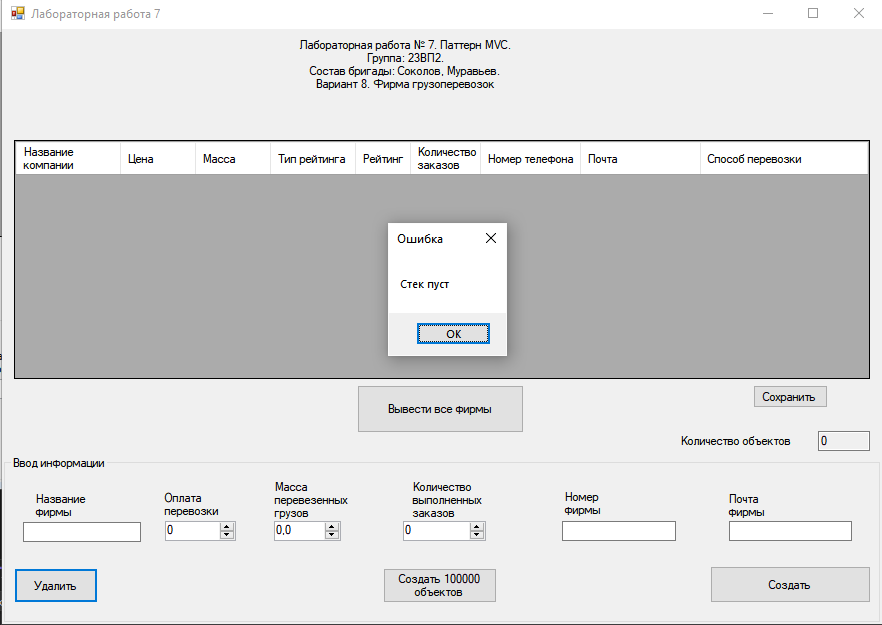


Рисунок 3 – Попытка удаления элемента из пустого стека

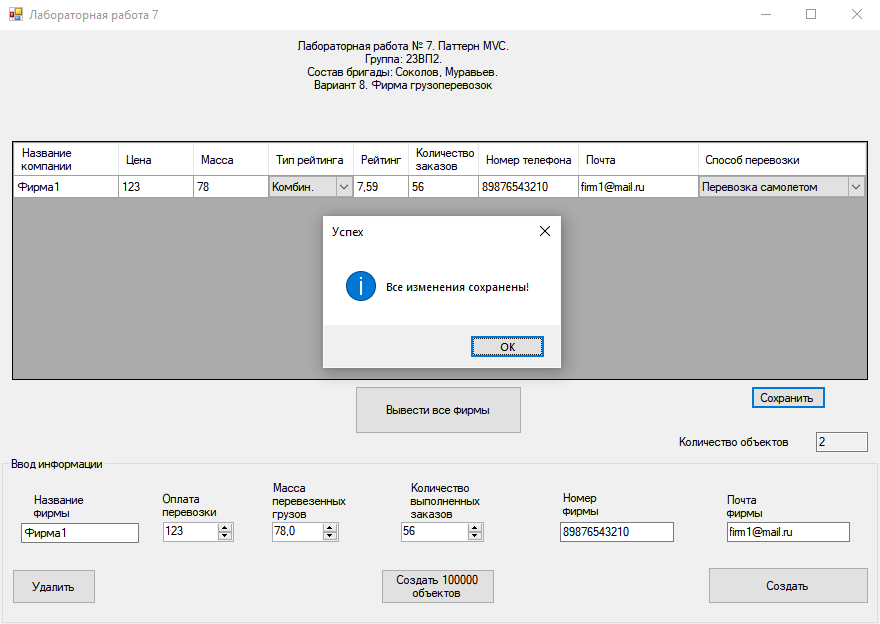


Рисунок 4 – Попытка изменения стратегии вычисления рейтинга и сохранения изменений

### Диаграмма вариантов использования

### C:\Users\Влад\Downloads\2025-03-31_19-50-47.png

Рисунок 5 - Диаграмма вариантов использования

### Выводы

В ходе лабораторной работы была реализована архитектура паттерна MVP (Model-View-Presenter), что позволило чётко разделить логику приложения на три компонента: модель, представление и посредник между ними — представитель. Это повысило модульность и читаемость кода, а также упростило сопровождение и масштабирование проекта. Благодаря использованию интерфейса IView удалось абстрагироваться от конкретной реализации интерфейса пользователя, что улучшает расширяемость и делает возможным лёгкое тестирование логики представителя независимо от формы.