Министерство образования Российской Федерации Пензенский государственный университет

Кафедра «Математическое обеспечение и применение ЭВМ»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «ООП» на тему «Структурные паттерны»

Выполнили:

студенты группы 23ВП2 Соколов О.А.

Муравьев В.С.

Принял:

к.т.н. Афонин А.Ю.

Пенза 2025

### Название

Структурные паттерны.

### Цель работы

Научиться использовать структурные паттерны.

### Лабораторное задание

Реализовать структурный паттерн программирования «мост».

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 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 abstract class TransportCompany : ICloneable

{

public ITransportMethod deliverMethod;

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,

float rating,

int completedOrders,

string phoneNumber,

string email,

ITransportMethod transportMethod) : this(price, name)

{

this.transportedMass = transportedMass;

this.rating = rating;

this.completedOrders = completedOrders;

this.phoneNumber = phoneNumber;

this.email = email;

this.deliverMethod = transportMethod;

}

public object Clone()

{

countObj++;

return this.MemberwiseClone();

}

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(int price,

float transportedMass,

string name,

float rating,

int completedOrders,

string phoneNumber,

string email,

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

phoneNumber, email, transportMethod) {}

}

}

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.Windows.Forms;

namespace Lab2

{

public class StackListener

{

private ListView listView;

private TextBox objCount;

public StackListener(StackTransportCompany stack, ListView listView, TextBox objCount)

{

this.listView = listView;

this.objCount = objCount;

stack.StackAdded += (TransportCompany company) =>

{

var listItem = new ListViewItem(company.name);

listItem.SubItems.Add(company.price.ToString());

listItem.SubItems.Add(company.transportedMass.ToString());

listItem.SubItems.Add(company.rating.ToString());

listItem.SubItems.Add(company.completedOrders.ToString());

listItem.SubItems.Add(company.phoneNumber);

listItem.SubItems.Add(company.email);

listItem.SubItems.Add(company.DoWork());

listView.Items.Add(listItem);

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

};

stack.StackRemoved += () =>

{

if (listView.Items.Count > 0)

listView.Items.RemoveAt(listView.Items.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;

namespace Lab2

{

public partial class Form1 : Form

{

private TransportCompany prototype = null;

private StackTransportCompany companies;

private StackListener stackListener;

public Form1()

{

InitializeComponent();

companies = new StackTransportCompany();

InitializeListView();

stackListener = new StackListener(companies, listView1, objCount);

}

private void InitializeListView()

{

listView1.Columns.Add("Название компании");

listView1.Columns.Add("Цена");

listView1.Columns.Add("Масса");

listView1.Columns.Add("Рейтинг");

listView1.Columns.Add("Количество заказов");

listView1.Columns.Add("Номер телефона");

listView1.Columns.Add("Почта");

listView1.Columns.Add("Способ перевозки");

listView1.View = View.Details;

}

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");

if (method.SelectedIndex == -1)

throw new MyException("Выберите способ перевозки");

TransportCompany firm;

ITransportMethod curMethod = null;

switch (method.SelectedItem.ToString())

{

case ("Кораблем"):

curMethod = new ShipTransport();

break;

case ("Грузовиком"):

curMethod = new TrackTransport();

break;

case ("Самолетом"):

curMethod = new AirTransport();

break;

}

if (prototype == null)

{

prototype = new LogisticCompany(

(int)price.Value,

(float)transportedMass.Value,

name.Text,

(float)rating.Value,

(int)completedOrders.Value,

phoneNumber.Text,

email.Text,

curMethod

);

}

firm = (LogisticCompany)prototype.Clone();

firm.price = (int)price.Value;

firm.transportedMass = (float)transportedMass.Value;

firm.name = name.Text;

firm.rating = (float)rating.Value;

firm.completedOrders = (int)completedOrders.Value;

firm.phoneNumber = phoneNumber.Text;

firm.email = email.Text;

firm.deliverMethod = curMethod;

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)

{

listView1.Items.Clear();

foreach(var company in companies.GetTransportCompanies())

{

var listItem = new ListViewItem(company.name);

listItem.SubItems.Add(company.price.ToString());

listItem.SubItems.Add(company.transportedMass.ToString());

listItem.SubItems.Add(company.rating.ToString());

listItem.SubItems.Add(company.completedOrders.ToString());

listItem.SubItems.Add(company.phoneNumber);

listItem.SubItems.Add(company.email);

listItem.SubItems.Add(company.DoWork());

listView1.Items.Insert(0, listItem);

}

}

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

{

Random rand = new Random();

int elementCount = 5000;

TransportCompany baseCompany = new LogisticCompany(

rand.Next(1000, 10000),

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

"BaseCompany",

(float)(rand.NextDouble() \* 5),

rand.Next(0, 1000),

"89990000000",

"basecompany@mail.ru",

new TrackTransport()

);

// Для StackTransportCompany

int start = Environment.TickCount;

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

{

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 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";

clonedCompany.deliverMethod = curMethod;

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++)

{

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;

}

companyArray[i] = new LogisticCompany(

rand.Next(1000, 10000),

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

"Company" + i,

(float)(rand.NextDouble() \* 5),

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;

// Результаты для своей коллекции

listView1.Items.Clear();

var collectionResultItem = new ListViewItem("StackTransportCompany");

collectionResultItem.SubItems.Add("Вставка: " + insertionTimeCollection + " мс");

collectionResultItem.SubItems.Add("Последоват. выборка: " + sequentialReadTimeCollection + " мс");

collectionResultItem.SubItems.Add("Случайная выборка: " + randomReadTimeCollection + " мс");

listView1.Items.Add(collectionResultItem);

// Результаты для Array

var arrayResultItem = new ListViewItem("Array");

arrayResultItem.SubItems.Add("Вставка: " + insertionTimeArray + " мс");

arrayResultItem.SubItems.Add("Последоват. выборка: " + sequentialReadTimeArray + " мс");

arrayResultItem.SubItems.Add("Случайная выборка: " + randomReadTimeArray + " мс");

listView1.Items.Add(arrayResultItem);

MessageBox.Show("Генерация и замеры завершены!");

}

}

}

### Пояснительный текст к программе

Абстрактный класс TransportCompany представляет общие характеристики всех транспортных компаний и содержит поля deliverMethod (способ перевозки), 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.

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.

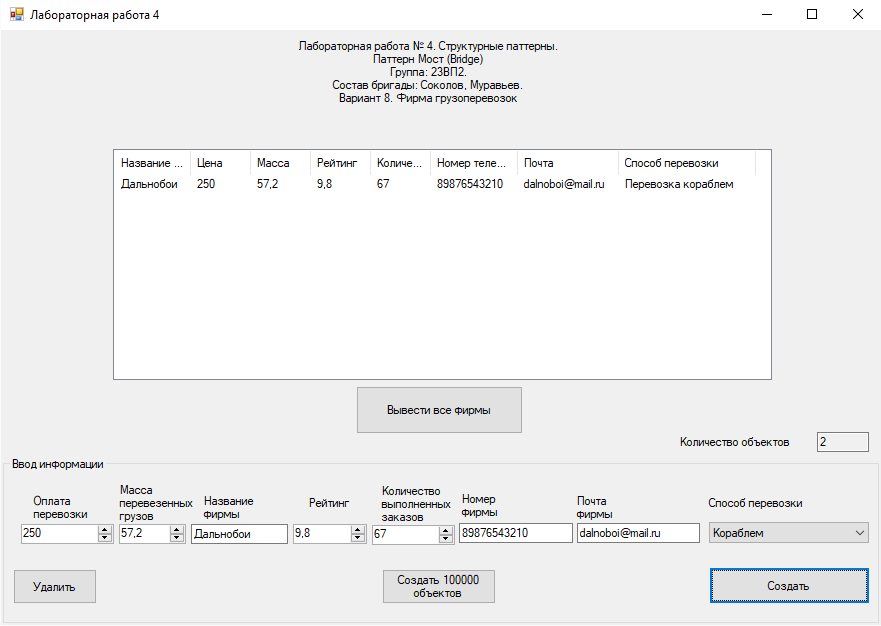


Рисунок 1 – Создание объекта, добавление в стек и вывод информации

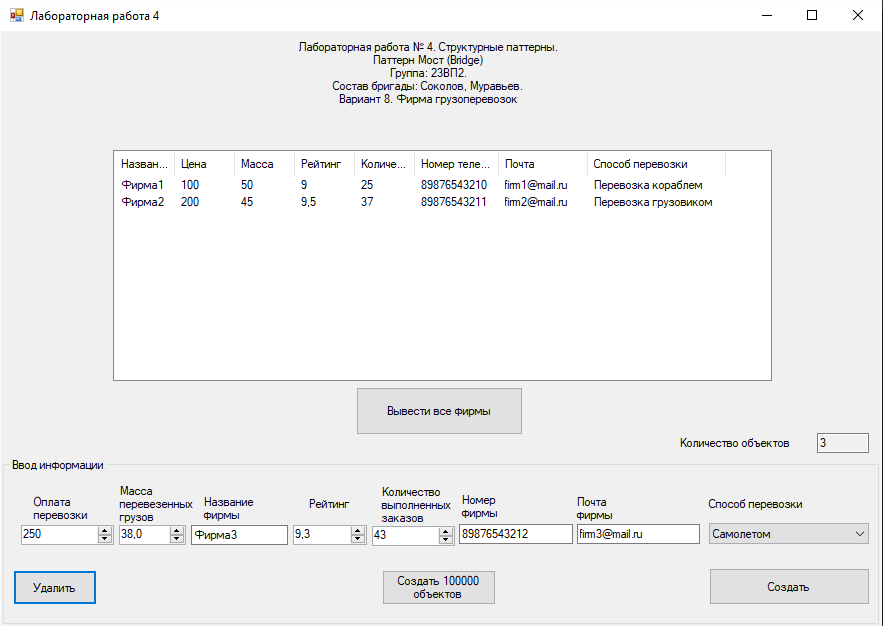


Рисунок 2 – Удаление фирмы из стека и вывод информации

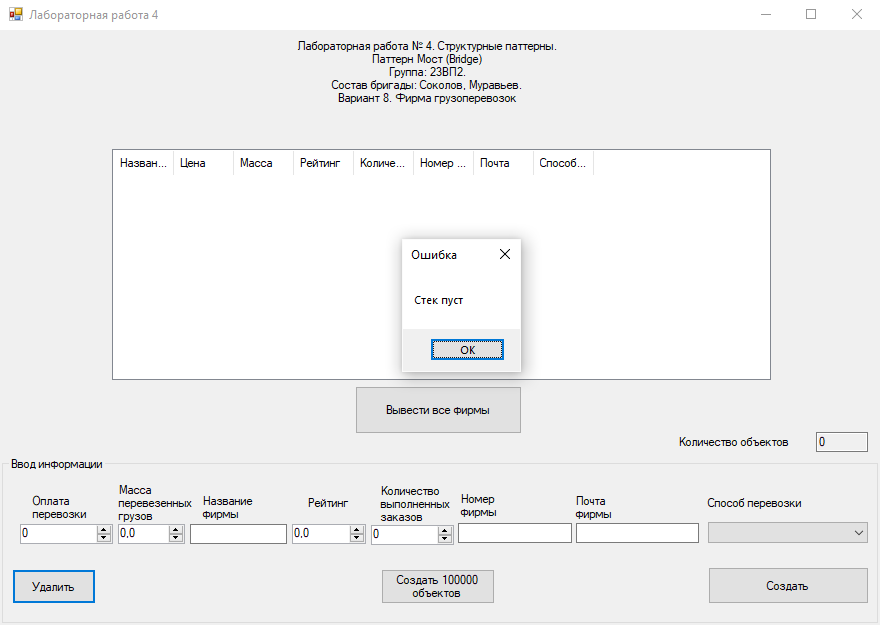


Рисунок 3 – Попытка удаления элемента из пустого стека

### Диаграмма вариантов использования

### C:\Users\Влад\Downloads\2025-02-25_01-28-01.png

**Рисунок 4 - Диаграмма вариантов использования**

### Выводы

В ходе выполнения лабораторной работы был реализован структурный паттерн «Мост». «Мост» - структурный шаблон проектирования, который позволяет отделить абстракцию от реализации таким образом, чтобы и абстракцию, и реализацию можно было изменять независимо друг от друга.