Министерство образования Российской Федерации Пензенский государственный университет

Кафедра «Математическое обеспечение и применение ЭВМ»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «ООП» на тему «Порождающие паттерны»

Выполнил:

Студент группы 23ВП1 Рейимов М.

Принял:

к.т.н. Афонин А.Ю.

Пенза 2025

### Название

Порождающие паттерны.

### Цель работы

Научиться использовать порождающие паттерны.

### Лабораторное задание

Реализовать паттерн программирования «прототип».

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 class TransportCompany: ICloneable

{

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 static int countObj = 0;

public TransportCompany()

{

name = "";

pricePerKilometer = 0;

averageDeliveryTime = 0;

yearFounded = 2000;

transportedMass = 0;

rating = 0;

phoneNumber = "";

countObj++;

}

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) : this(name, pricePerKilometer)

{

this.averageDeliveryTime = averageDeliveryTime;

this.yearFounded = yearFounded;

this.transportedMass = transportedMass;

this.rating = rating;

this.phoneNumber = phoneNumber;

}

public object Clone()

{

countObj++;

return this.MemberwiseClone();

}

}

}

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 Lab1;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab2

{

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();

}

}

}

Файл StackListener.cs:

using Lab1;

using System;

using System.Windows.Forms;

namespace Lab2

{

internal 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.pricePerKilometer;

dataGridView.Rows[rowIndex].Cells[2].Value = company.averageDeliveryTime;

dataGridView.Rows[rowIndex].Cells[3].Value = company.yearFounded;

dataGridView.Rows[rowIndex].Cells[4].Value = company.transportedMass;

dataGridView.Rows[rowIndex].Cells[5].Value = company.rating;

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

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

};

stack.StackRemoved += () =>

{

if (dataGridView.RowCount > 0)

dataGridView.Rows.RemoveAt(dataGridView.RowCount - 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 TransportCompany prototype = null;

private StackTransportCompany companies;

private StackListener stackListener;

public Form1()

{

InitializeComponent();

MessageBox.Show("Порождающие паттерны. Прототип.\n" +

"Рейимов М. 23ВП1\nВариант 8. Фирма грузоперевозок", "Лабораторная работа 3");

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");

TransportCompany firm = new TransportCompany();

if (prototype == null)

{

prototype = new TransportCompany(nameInput.Text,

(int)priceInput.Value,

(float)avgtimeInput.Value,

(int)yearInput.Value,

(float)massInput.Value,

(float)ratingInput.Value,

phonenumberInput.Text);

}

firm = (TransportCompany)prototype.Clone();

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())

{

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;

}

}

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

{

Random rand = new Random();

int elementCount = 5000;

TransportCompany baseCompany = new TransportCompany();

baseCompany.averageDeliveryTime = (float)rand.NextDouble() \* 99 + 1;

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

baseCompany.name = "BaseCompany";

baseCompany.pricePerKilometer = rand.Next(1000, 10000);

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

baseCompany.rating = (float)rand.NextDouble() \* 10;

baseCompany.yearFounded = rand.Next(1950, 2025);

int start = Environment.TickCount;

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

{

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

clonedCompany.averageDeliveryTime = (float)rand.NextDouble() \* 99 + 1;

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

clonedCompany.name = "Company" + i;

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

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

clonedCompany.rating = (float)rand.NextDouble() \* 10;

clonedCompany.yearFounded = rand.Next(1950, 2025);

companies.AddCompany(clonedCompany);

}

int insertionTimeCollection = Environment.TickCount - start;

start = Environment.TickCount;

foreach (var company in companies.GetTransportCompanies())

{

var temp = company.pricePerKilometer;

}

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)].pricePerKilometer;

}

int randomReadTimeCollection = Environment.TickCount - start;

TransportCompany[] companyArray = new TransportCompany[elementCount];

start = Environment.TickCount;

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

{

TransportCompany arrCompany = new TransportCompany();

arrCompany.averageDeliveryTime = (float)rand.NextDouble() \* 99 + 1;

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

arrCompany.name = "Company" + i;

arrCompany.pricePerKilometer = rand.Next(1000, 10000);

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

arrCompany.rating = (float)rand.NextDouble() \* 10;

arrCompany.yearFounded = rand.Next(1950, 2025);

companyArray[i] = arrCompany;

}

int insertionTimeArray = Environment.TickCount - start;

start = Environment.TickCount;

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

{

var temp = companyArray[i].pricePerKilometer;

}

int sequentialReadTimeArray = Environment.TickCount - start;

start = Environment.TickCount;

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

{

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

}

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

Класс TransportCompany представляет собой модель транспортной компании и содержит поля name (наименование компании), pricePerKilometer (цена за километр), averageDeliveryTime (среднее время доставки), yearFounded (год основания), transportedMass (масса перевезенного груза), rating (рейтинг компании), phoneNumber (номер телефона).

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.

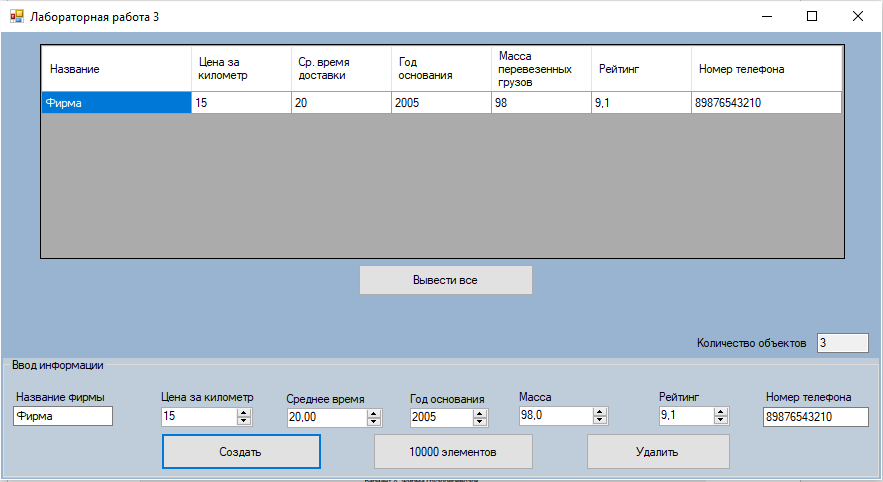


Рисунок 1 – Создание объекта, добавление в стек и вывод информации

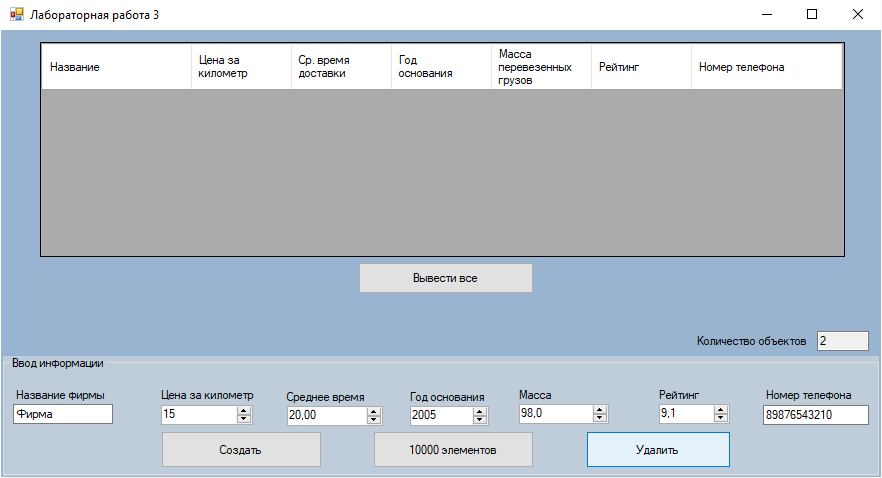


Рисунок 2 – Удаление фирмы из стека и вывод информации

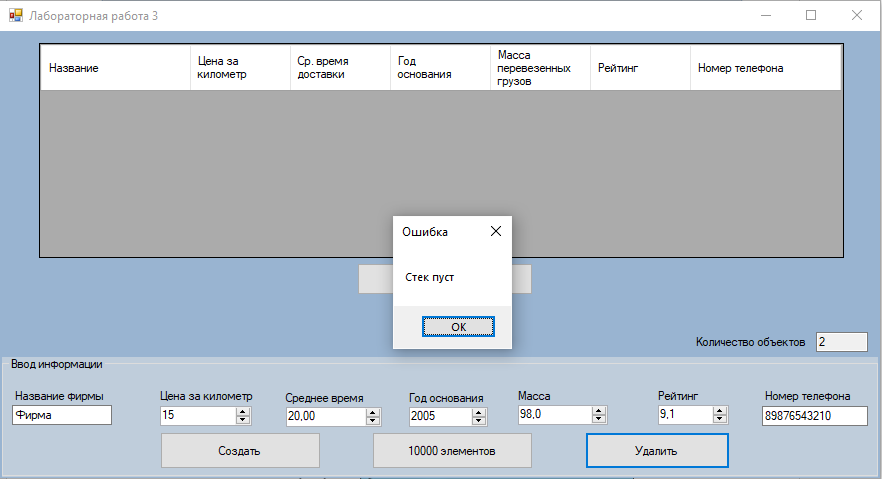


Рисунок 3 – Попытка удаления элемента из пустого стека

### Диаграмма вариантов использования

### C:\Users\Влад\Downloads\2025-05-10_17-36-20.png

Рисунок 4 - Диаграмма вариантов использования

### Выводы

В ходе выполнения лабораторной работы был реализован порождающий паттерн «Прототип». «Прототип» - это паттерн создания объекта через клонирование другого объекта вместо создания его через конструктор.