**Занятие № 8**

**Дата выполнения работы:** 27.04.2023

# **Тема работы:** «Интерфейсы»

**Ход работы**

**Задание 1**

1. Абстрактный класс Книга (Шифр, Автор, Название, Год, Издательство).

Подклассы Справочник и Энциклопедия.

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public abstract class Book

{

public string Code { get; set; }

public string Author { get; set; }

public string Title { get; set; }

public int Year { get; set; }

public string Publisher { get; set; }

// абстрактный метод для получения информации о книге в виде строки

public abstract string GetBookInfo();

}

public class Encyclopedia : Book

{

public string Topic { get; set; }

public override string GetBookInfo()

{

return $"Encyclopedia: {Title} by {Author} ({Year}), Topic: {Topic}, Publisher: {Publisher}, Code: {Code}";

}

}

public class Directory : Book

{

public string Category { get; set; }

public override string GetBookInfo()

{

return $"Directory: {Title} by {Author} ({Year}), Category: {Category}, Publisher: {Publisher}, Code: {Code}";

}

}

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

// создаем экземпляр класса Справочник

var directory = new Directory

{

Code = "D-123",

Author = "John Doe",

Title = "Programming Guide",

Year = 2022,

Publisher = "ACME Publishing",

Category = "Programming"

};

// создаем экземпляр класса Энциклопедия

var encyclopedia = new Encyclopedia

{

Code = "E-456",

Author = "Jane Smith",

Title = "World History",

Year = 2021,

Publisher = "XYZ Books",

Topic = "History"

};

// выводим информацию о книгах в текстовом поле

textBox1.Text = directory.GetBookInfo() + Environment.NewLine + Environment.NewLine;

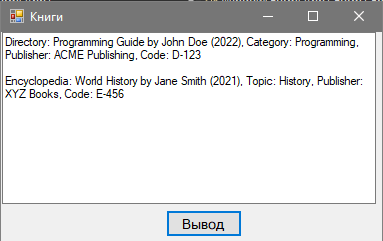
textBox1.Text += encyclopedia.GetBookInfo();

}

}

}

**Результат:**



**Задание 2**

В следующих заданиях требуется создать суперкласс (абстрактный класс,

интерфейс) и определить общие методы для данного класса. Создать

подклассы, в которых добавить специфические свойства и методы. Часть

методов переопределить. Создать массив объектов суперкласса и заполнить

объектами подклассов. Объекты подклассов идентифицировать

конструктором по имени или идентификационному номеру. Использовать

объекты подклассов для моделирования реальных ситуаций и объектов

1) Создать суперкласс Транспортное средство и подклассы Автомобиль,

Велосипед, Повозка. Подсчитать время и стоимость перевозки

пассажиров и грузов каждым транспортным средством.

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Transport[] transports = new Transport[]

{

new Car { Name = "BMW", PassengerCapacity = 5, CargoCapacity = 500, Speed = 120, CostPerKm = 10, NumberOfDoors = 4 },

new Bicycle { Name = "Author", PassengerCapacity = 1, CargoCapacity = 10, Speed = 20, CostPerKm = 0, NumberOfGears = 18 },

new Cart { Name = "Kustar", PassengerCapacity = 2, CargoCapacity = 200, Speed = 30, CostPerKm = 5, HorsePower = 1.5 }

};

foreach (Transport transport in transports)

{

string[] name = new string[] { $"Transport: {transport.Name}" };

string[] PassengerCapacity = new string[] { $"Passenger Capacity: {transport.PassengerCapacity}" };

string[] CargoCapacity = new string[] { $"Cargo Capacity: {transport.CargoCapacity}" };

string[] Speed = new string[] { $"Speed: {transport.Speed} km/h" };

string[] CostPerKm = new string[] { $"Cost per km: {transport.CostPerKm} rub" };

string[] CalculateTime = new string[] { $"Time for 100 km: {transport.CalculateTime(100)} hours" };

string[] CalculateCost = new string[] { $"Cost for 100 km with 3 passengers and 100 kg of cargo: {transport.CalculateCost(100, 3, 100)} rub" };

listBox1.Items.AddRange(name);

listBox1.Items.AddRange(PassengerCapacity);

listBox1.Items.AddRange(CargoCapacity);

listBox1.Items.AddRange(Speed);

listBox1.Items.AddRange(CostPerKm);

listBox1.Items.AddRange(CalculateTime);

listBox1.Items.AddRange(CalculateCost);

listBox1.Items.Add("");

}

}

}

abstract class Transport

{

public string Name { get; set; }

public int PassengerCapacity { get; set; }

public double CargoCapacity { get; set; }

public double Speed { get; set; }

public double CostPerKm { get; set; }

public abstract double CalculateTime(double distance);

public abstract double CalculateCost(double distance, int passengers, double cargo);

}

class Car : Transport

{

public int NumberOfDoors { get; set; }

public override double CalculateTime(double distance)

{

return distance / this.Speed;

}

public override double CalculateCost(double distance, int passengers, double cargo)

{

double cost = distance \* this.CostPerKm;

if (passengers > this.PassengerCapacity)

{

cost += (passengers - this.PassengerCapacity) \* 100; // Дополнительная плата за пассажиров

}

if (cargo > this.CargoCapacity)

{

cost += (cargo - this.CargoCapacity) \* 50; // Дополнительная плата за груз

}

return cost;

}

}

class Bicycle : Transport

{

public int NumberOfGears { get; set; }

public override double CalculateCost(double distance, int passengers, double cargo)

{

return 0;

}

public override double CalculateTime(double distance)

{

return distance / this.Speed \* 2; // Велосипед едет медленнее, поэтому удваиваем время

}

}

class Cart : Transport

{

public double HorsePower { get; set; }

public override double CalculateTime(double distance)

{

return distance / this.Speed \* 1.5; // Повозка едет медленнее, поэтому увеличиваем время на полтора

}

public override double CalculateCost(double distance, int passengers, double cargo)

{

double cost = distance \* this.CostPerKm;

if (passengers > this.PassengerCapacity)

{

cost += (passengers - this.PassengerCapacity) \* 50; // Дополнительная плата за пассажиров

}

if (cargo > this.CargoCapacity)

{

cost += (cargo - this.CargoCapacity) \* 20; // Дополнительная плата за груз

}

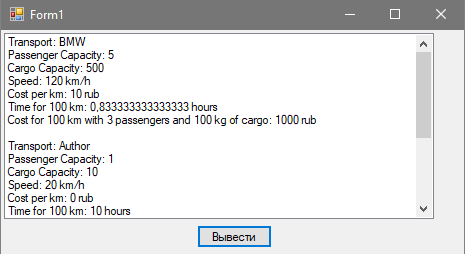
return cost;

}

}

}

**Результат:**

****

**Вывод:** Я понял, что интерфейсы используются для достижения более высокой степени абстракции и полиморфизма в программировании. Интерфейсы позволяют разделять обязанности между классами, делать код более гибким и модульным, а также позволяют использовать множественное наследование.

Я усвоил, что использование интерфейсов может привести к излишней сложности кода и требовать дополнительных ресурсов на реализацию. Поэтому, перед использованием интерфейсов в проекте, необходимо внимательно оценить их необходимость и целесообразность в конкретном случае.

В целом, использование интерфейсов в C# является мощным инструментом для проектирования и разработки гибких, модульных и масштабируемых систем.