**Лабораторная работа №6**

**Титов Павел 04-1**

**Порядок выполнения работы:**

**Общее для всех уровней**

 реализовать иерархию классов с использованием абстрактного класса в качестве основы иерархии;

 разработать поля, методы и свойства для каждого из определяемых классов;  все поля классов должны быть описаны с ключевым словом private;

 реализовать для каждого класса конструкторы по умолчанию и конструкторы с параметрами;

 свойства или индексаторы для изменения значений полей;

 для вывода полей класса переопределить метод ToString();

 методы поиска объектов по разным полям

 показать на примере одного из методов, присутствующих в каждом классе, свойство полиморфизма;

 в качестве хранилища данных использовать список, элементы которого являются элементами базового класса (класса - определяющего начало иерархии).

**Повышенный уровень** (на максимальный балл за лабораторную работу)

 реализовать для иерархии механизм интерфейсов, при этом один из классов должен реализовывать как минимум 2 интерфейса;

 предусмотреть возможность совпадения названий методов в интерфейсах;

 на примере класса, который наследует 2 интерфейса, показать каким образом осуществляется реализация методов при совпадении имен;

 использовать для проверки всех методов данного класса многоадресный делегат.

> Описать реализацию свойства полиморфизма, интерфейсов.

*Полиморфизм у меня реализован в большинстве методов, его можно увидеть по ключевому слову override. Смысл в том, что в абстрактном классе я описываю этот метод, а в дочерних классах уже реализую.*

*Интерфейсы начинаются с большой буквы I, в моём случае это — IRepair и IMod. У обоих интерфейсов описан метод Code, и в наследуемом классе Car я реализую оба эти поля.*

Реализовать программу на C# в соответствии с вариантом.

**Варианты заданий**

***Повышенный уровень*** Варианты заданий определяет преподаватель. Построить иерархию классов в соответствии с вариантом задания:

11) Автомобиль, поезд, транспортное средство, экспресс.

Диаграмма классов

interfaces:

IRepair

IMod

Vehicle:

→ Train

→ Car (наследуется от IRepair IMod)

→ Express

class Vehicle:

+ Name

+ Color

+ ZodiakSign

+ Speed

+ Move

+ SpeedUp

+ Teleport

+ BeepBeep

+ Info

**Car.cs**

namespace L6

{

public class Car : Vehicle, IRepair, IMod

{

private int doors;

private string carType = "Car";

public int Doors { get => doors; set => doors = value; }

public string CarType { get => carType; set => carType = value; }

void IRepair.Code(string? cheatCode)

{

if (cheatCode == "HESOYAM")

Console.WriteLine("Починка удалась");

else

Console.WriteLine("Починка не удалась");

}

void IMod.Code(string? cheatCode)

{

if (cheatCode == "freefarm")

Console.WriteLine("Модификация удалась");

else

Console.WriteLine("Модификация не удалась");

}

public bool checkDoor(int doorNum)

{

return (doorNum == this.doors);

}

public Car() : base() { }

public Car(string name) : base(name) { }

public Car(string name, string color) : base(name, color) { }

public Car(string name, string color, string zodiacSign) : base(name, color, zodiacSign) { }

public Car(string name, string color, string zodiacSign, int speed) : base(name, color, zodiacSign, speed) { }

public override void Move(string? endpoint) => Console.WriteLine($"ДВИЖЕМСЯ ПРЯМО ПО КУРСУ В {endpoint} НА {Name} {Color} цвета (между прочим {ZodiacSign}) НА СКОРОСТИ {Speed}км/с!!!!!!!");

public override void SpeedUp(int newSpeed)

{

Speed = newSpeed;

Console.WriteLine($"СКОРОСТЬ ВАШЕГО АВТОМОБИЛЯ \"{Name}\" ПОДНЯЛАСЬ ДО {newSpeed}, А ЭТО ЗНАЧИТ, ЧТО ...");

Console.WriteLine($"\tПусть луна нам светит ярко");

Console.WriteLine($"\tОбгоняем иномарку...");

}

public override void Teleport(string? location) => Console.WriteLine($"Телепортация в {location} невозможна, пожалуйста, обновите ваше транспортное средство.");

public override void BeepBeep(int duration) { for (int i = 0; i < duration; i++) { Console.WriteLine($"Beep ..."); } }

delegate void Movement(string s);

delegate void Edit(int x);

delegate void Cheat(string s);

public override void DopInfo()

{

Console.WriteLine("Dop info for Car");

Cheat repair, modi, multiCheatDel;

Movement move, teleport, multiMoveDel;

Edit speedup, beeper, multiEditDel;

repair = IRepair.Code;

modi = IMod.Code;

move = Move;

teleport = Teleport;

speedup = SpeedUp;

beeper = BeepBeep;

multiMoveDel = move + teleport;

multiEditDel = speedup + beeper;

multiCheatDel = repair + modi;

multiMoveDel("Казахстан");

multiEditDel(2);

multiCheatDel("HESOYAM");

}

}

}

**Express.cs**

namespace L6

{

public class Express : Vehicle

{

private int vagons;

public int Vagons { get => vagons; set => vagons = value; }

public Express() : base() { }

public Express(string name) : base(name) { }

public Express(string name, string color) : base(name, color) { }

public Express(string name, string color, string zodiacSign) : base(name, color, zodiacSign) { }

public Express(string name, string color, string zodiacSign, int speed) : base(name, color, zodiacSign, speed) { }

public bool checkVagons(int vagonNum)

{

return vagonNum == this.vagons;

}

public override void Move(string? endpoint) => Console.WriteLine($"Экспресс ({Color} {Name}, {ZodiacSign}) отправляется в {endpoint}. У нас есть Wi-Fi.");

public override void SpeedUp(int newSpeed)

{

Speed = newSpeed;

Console.WriteLine($"Экспресс \"{Name}\" увеличил свою скорость до {newSpeed}...");

}

public override void Teleport(string? location) => Console.WriteLine($"&^%#!@#!@#!@#!JH@G#!@JKGH#!@J#HG!@#K Done. Добро пожаловать в {location}");

public override void BeepBeep(int duration) { for (int i = 0; i < duration; i++) { Console.WriteLine($"BEEP ..."); } }

public override void DopInfo()

{

Console.WriteLine("Dop info for Express");

}

}

}

**Interfaces.cs**

namespace L6

{

public interface IRepair

{

void Code(string? cheatCode);

}

public interface IMod

{

void Code(string? cheatCode);

}

}

**Train.cs**

namespace L6

{

public class Train : Vehicle

{

private int vagons;

private int maxSpeed;

public int Vagons { get => vagons; set => vagons = value; }

public Train() : base() { }

public Train(string name) : base(name) { }

public Train(string name, string color) : base(name, color) { }

public Train(string name, string color, string zodiacSign) : base(name, color, zodiacSign) { }

public Train(string name, string color, string zodiacSign, int speed) : base(name, color, zodiacSign, speed) { }

public bool checkMaxSpeed(int speedNum)

{

return (speedNum == this.maxSpeed);

}

public override void Move(string? endpoint) => Console.WriteLine($"Поезд ({Color} {Name}, {ZodiacSign}) отправляется в {endpoint}. У нас есть Wi-Fi + кофе + душ)");

public override void SpeedUp(int newSpeed)

{

Speed = newSpeed;

Console.WriteLine($"Поезд \"{Name}\" увеличил свою скорость до {newSpeed}. Туалет закрыт)");

}

public override void Teleport(string? location) => Console.WriteLine($"Извините, но пока что точка {location} временно недоступна для вашей страны");

public override void BeepBeep(int duration) => Console.WriteLine($"Чё? Какой бип бип, ты в поезде ...");

public override void DopInfo()

{

Console.WriteLine("Dop info for Train");

}

}

}

**Vehicle.cs**

using System.Xml.Serialization; // нужно для сериализации объектов и сохранения в файлы

namespace L6

{

[XmlInclude(typeof(Car)), XmlInclude(typeof(Train)), XmlInclude(typeof(Express))]

abstract public class Vehicle

{

private string name = "имя не указано";

private string color = "цвет не указан";

private string zodiacSign = "зз не указан";

private int speed;

public string? Name { get => name; set => name = value is null ? name : value; }

public string? Color { get => color; set => color = value is null ? color : value; }

public string? ZodiacSign { get => zodiacSign; set => zodiacSign = value is null ? zodiacSign : value; }

public int Speed { get => speed; set => speed = value; }

public bool checkColor(string colorInput)

{

return colorInput == this.color;

}

public Vehicle() { }

public Vehicle(string name)

{

Name = name;

}

public Vehicle(string name, string color)

{

Name = name;

Color = color;

}

public Vehicle(string name, string color, string zodiacSign)

{

Name = name;

Color = color;

ZodiacSign = zodiacSign;

}

public Vehicle(string name, string color, string zodiacSign, int speed)

{

Name = name;

Color = color;

ZodiacSign = zodiacSign;

Speed = speed;

}

public abstract void Move(string? endpoint);

public abstract void SpeedUp(int newSpeed);

public abstract void Teleport(string? location);

public abstract void BeepBeep(int duration);

public abstract void DopInfo();

public override string ToString()

{

return $"Переопределённый вывод: \n{Info()}";

}

public string Info()

{

string s = ("----------------INFO--------------------\n");

s += ($"Название вашей ласточки: {Name}\n");

s += ($"Цвет: {Color}\n");

s += ($"Знак зодиака: {ZodiacSign}\n");

s += ($"Текущая скорость: {Speed}\n");

s += ("----------------------------------------\n");

DopInfo();

return s;

}

}

}

**Working.cs**

using System.Xml.Serialization; // нужно для сериализации объектов и сохранения в файлы

namespace L6

{

public class WorkingWithObjectsClass

{

public int StrToIntDef(string? s, int @default)

{

int number;

if (int.TryParse(s, out number))

return number;

return @default;

}

public void Serialize(List<Vehicle> vehicles)

{

XmlSerializer xmlSerializer = new XmlSerializer(typeof(Vehicle[])); // для сериализации указывается массив объектов базового класса

// получаем поток, куда будем записывать сериализованный объект

foreach (var item in vehicles)

Console.WriteLine(item);

using (FileStream fs = new FileStream("list.xml", FileMode.Create)) // для безопасной работы с потоком используется using

{

xmlSerializer.Serialize(fs, vehicles.ToArray()); // сериализация, передается массив объектов

Console.WriteLine("Данные сохранены");

Console.ReadKey();

}

}

public List<Vehicle> Deserialize()

{

try

{

List<Vehicle> vehicles = new List<Vehicle>();

XmlSerializer xmlSerializer = new XmlSerializer(typeof(Vehicle[])); // по аналогии с сериализацией

using (FileStream fs = new FileStream("list.xml", FileMode.OpenOrCreate))

{

Vehicle[]? temp = xmlSerializer.Deserialize(fs) as Vehicle[];

if (temp is not null)

vehicles.AddRange(temp);

}

Console.WriteLine("Данные загружены");

return vehicles;

}

catch (Exception e)

{

Console.WriteLine("Не удалось загрузить данные. {0}", e.Message);

return new List<Vehicle>();

}

}

public void PrintList(List<Vehicle> vehicles)

{

int index = 0;

if (vehicles.Count == 0)

{

Console.WriteLine("Список пуст");

return;

}

foreach (var item in vehicles)

{

Console.WriteLine("{0}:", index);

index++;

if (item is Car)

{

((Car)item).Info();

}

else if (item is Train)

{

((Train)item).Info();

}

else if (item is Express)

{

((Express)item).Info();

}

else

{

item.Info();

}

}

}

public void Add(List<Vehicle> vehicles) // метод добавления элемента в список, список передается в качестве параметра

{

Console.Clear();

Console.WriteLine("1. Автомобиль");

Console.WriteLine("2. Поезд");

Console.WriteLine("3. Экспресс");

int c = StrToIntDef(Console.ReadLine(), 99);

Console.Write("Введите название (default: \"Valera\"): ");

string? name = Convert.ToString(Console.ReadLine());

if (name is null || name == "") name = "Valera";

Console.Write("Введите цвет (default: \"Blue\"): ");

string? color = Convert.ToString(Console.ReadLine());

if (color is null || color == "") color = "Blue";

Console.Write("Введите знак зодиака (default: \"Vodoley\"): ");

string? zz = Convert.ToString(Console.ReadLine());

if (zz is null || zz == "") zz = "Vodoley";

Console.Write("Введите скорость (default: \"0\"): ");

int speed = StrToIntDef(Console.ReadLine(), 0);

switch (c)

{

case 1:

vehicles.Add(new Car(name, color, zz, speed));

break;

case 2:

vehicles.Add(new Train(name, color, zz, speed));

break;

case 3:

vehicles.Add(new Express(name, color, zz, speed));

break;

default:

Console.WriteLine("Под номером вашим номером, транспорта пока нет.");

break;

}

}

public void Remove(List<Vehicle> vehicles)// метод для удаления из списка

// удаление просто по индексу элемента в списке

{

Console.WriteLine("Введите номер элемента, который хотите удалить:");

// int c = Convert.ToInt32(Console.ReadLine());

int c = StrToIntDef(Console.ReadLine(), -1);

if (c < 0 || c >= vehicles.Count)

{

Console.WriteLine("Элемент не найден");

return;

}

else vehicles.RemoveAt(c);

}

public void WorkWithObject(List<Vehicle> vehicles)

{

Console.WriteLine("Краткий список элементов:");

int index = 0;

foreach (var i in vehicles)

{

Console.WriteLine($"➻ {index}: {i.Name}");

index++;

}

Console.WriteLine("Введите номер элемента, с которым хотите работать:");

int c = StrToIntDef(Console.ReadLine(), -1);

if (c < 0 || c >= vehicles.Count)

{

Console.WriteLine("Элемент не найден");

return;

}

Console.WriteLine("1. Изменить имя");

Console.WriteLine("2. Изменить скорость");

Console.WriteLine("3. Изменить цвет");

Console.WriteLine("4. Изменить Знак Зодиака");

Console.WriteLine("5. Движение");

Console.WriteLine("6. Телепортация");

Console.WriteLine("7. SpeedUp");

Console.WriteLine("8. Побибикать");

switch (StrToIntDef(Console.ReadLine(), 99))

{

case 1:

Console.Write("Введите название: ");

vehicles[c].Name = Convert.ToString(Console.ReadLine());

break;

case 2:

Console.Write("Введите скорость: ");

vehicles[c].Speed = StrToIntDef(Console.ReadLine(), 99);

break;

case 3:

Console.Write("Введите цвет: ");

vehicles[c].Color = Convert.ToString(Console.ReadLine());

break;

case 4:

Console.Write("Введите Знак Зодиака: ");

vehicles[c].ZodiacSign = Convert.ToString(Console.ReadLine());

break;

case 5:

Console.Write("Введите конечную точку: ");

vehicles[c].Move(Convert.ToString(Console.ReadLine()));

break;

case 6:

Console.Write("Введите точку для телепортации: ");

vehicles[c].Teleport(Convert.ToString(Console.ReadLine()));

break;

case 7:

Console.Write("Введите скорость: ");

vehicles[c].SpeedUp(StrToIntDef(Console.ReadLine(), 99));

break;

case 8:

Console.Write("Введите кол-во биков: ");

vehicles[c].BeepBeep(StrToIntDef(Console.ReadLine(), 99));

break;

default:

Console.WriteLine("Такого метода нет");

break;

}

Console.WriteLine("Уникальные методы: ");

if (vehicles[c] is Car) // список методов выводится в зависимости от выбранного объекта

{

Car car = (Car)vehicles[c];

Console.WriteLine("1. Вывод количества дверей");

Console.WriteLine("2. Вывод типа авто");

switch (StrToIntDef(Console.ReadLine(), 99))

{

case 1:

Console.WriteLine(car.Doors);

break;

case 2:

Console.WriteLine(car.CarType);

break;

default:

Console.WriteLine("Такого метода нет");

break;

}

}

else if (vehicles[c] is Train)

{

Train train = (Train)vehicles[c];

Console.WriteLine("1. Вывод количества вагонов");

switch (StrToIntDef(Console.ReadLine(), 99))

{

case 1:

Console.WriteLine(train.Vagons);

break;

default:

Console.WriteLine("Такого метода нет");

break;

}

}

else if (vehicles[c] is Express)

{

Express express = (Express)vehicles[c];

Console.WriteLine("1. Вывод количества вагонов");

switch (StrToIntDef(Console.ReadLine(), 99))

{

case 1:

Console.WriteLine(express.Vagons);

break;

default:

Console.WriteLine("Такого метода нет");

break;

}

}

}

}

}

**Program.cs**

namespace L6

{

class Program

{

static void SearchVagons(List<Vehicle> vehicles, int i)

{

var selectedVehicles = vehicles.Where(el => el is Express && ((Express)el).checkVagons(i));

foreach (var vehicle in selectedVehicles)

Console.WriteLine(vehicle);

}

static void SearchDoors(List<Vehicle> vehicles, int i)

{

var selectedVehicles = vehicles.Where(el => el is Car && ((Car)el).checkDoor(i));

foreach (var vehicle in selectedVehicles)

Console.WriteLine(vehicle);

}

static void SearchMaxSpeed(List<Vehicle> vehicles, int i)

{

var selectedVehicles = vehicles.Where(el => el is Train && ((Train)el).checkMaxSpeed(i));

foreach (var vehicle in selectedVehicles)

Console.WriteLine(vehicle);

}

static void SearchColor(List<Vehicle> vehicles, string c)

{

var selectedVehicles = vehicles.Where(el => el.checkColor(c));

foreach (var vehicle in selectedVehicles)

Console.WriteLine(vehicle);

}

static void Main(string[] args)

{

WorkingWithObjectsClass wwoc = new();

List<Vehicle> vehicles = wwoc.Deserialize();

SearchVagons(vehicles, 2);

SearchDoors(vehicles, 2);

SearchMaxSpeed(vehicles, 2);

SearchColor(vehicles, "Blue");

try

{

Console.WriteLine("Нажмите любую клавишу для начала взаимодействия с данными");

Console.ReadKey();

bool work = true;

while (work)

{

Console.Clear();

Console.WriteLine(" \* МЕНЮ \* ");

Console.WriteLine("1. Вывести весь список");

Console.WriteLine("2. Добавить новый элемент в список");

Console.WriteLine("3. Удалить элемент из списка");

Console.WriteLine("4. Работа с элементом");

Console.WriteLine("Любая другая клавиша - выход");

switch (wwoc.StrToIntDef(Console.ReadLine(), 99))

{

case 1:

wwoc.PrintList(vehicles);

break;

case 2:

wwoc.Add(vehicles);

break;

case 3:

wwoc.Remove(vehicles);

break;

case 4:

wwoc.WorkWithObject(vehicles);

break;

default:

work = false;

break;

}

if (work)

{

Console.WriteLine("Чтобы продолжить нажмите любую клавишу");

Console.ReadKey();

}

}

}

catch (Exception e)

{

Console.WriteLine("Очень жаль, что не получилось");

Console.WriteLine(e);

foreach (var i in vehicles)

Console.WriteLine(i);

}

finally

{

wwoc.Serialize(vehicles); // в конце работы сохраняем список в файл

}

}

}

}

**Работа программы**

