Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

­­­­­

Конструирование программного обеспечения

Отчет по лабораторной работе №7

««Интерфейсы. Коллекции»

Выполнил: Усов А.М.

Студент группы 310901

Преподаватель: Давыдович К. И.

Минск 2023

Цель: : получить навыки создания и реализации интерфейсов. Изучить стандартные коллекции языка C#.

Задание 1. Реализовать интерфейс, который будет расширять предметную область из предыдущих лабораторных работ. Добавить в интерфейс:

* + метод вывода на экран всех полей класса;
  + свойство содержащее название объекта;
  + метод, который делает реверс названия объекта.

2. Добавить в интерфейс для одного из методов реализацию по умолчанию.

3. Реализовать интерфейс всеми классами потомками, которые находятся в самом низу иерархии наследования.

4. Добавить в программу любой класс, который будет реализовывать интерфейс.

5. В главном классе создать метод, который будет создавать экземпляр класса из стандартной библиотеки коллекции, в качестве типа хранимых объектов использовать тип интерфейса.

6. Вывести на экран все элементы коллекции, затем сделать реверс названий объектов и повторить операцию.

7. Для обработки всех ошибочных ситуаций использовать конструкцию try…catch().

Код программы 1:

namespace ЛР7;

public static class Program

{

public static void Main(String[] arg)

{

MushroomContainer container = new MushroomContainer();

/\*container.FillArray(3);

container.Print();

container.DeleteByName("Name");

container.Print();\*/

Poganka poganka = new Poganka("Poganka", 5, ClassMush.Poisonous, 100);

poganka.Print();

redMush redMush = new redMush("redMush", 5, ClassMush.Inedible, "poison", "no");

redMush.Print();

WhiteMush whiteMush = new WhiteMush("whiteMush", 5, ClassMush.Edible, "good", "good");

whiteMush.Print();

InedibleMushroom inedibleMushroom = new InedibleMushroom("inedibleMushroom", 5, ClassMush.Inedible, "poison");

inedibleMushroom.Print();

EdibleMushroom edibleMushroom = new EdibleMushroom("edibleMushroom", 5, ClassMush.Edible, "good");

edibleMushroom.Print();

container.Print();

for (int i = 0; i < container.getSize(); i++)

{

container.Mushrooms[i].Name = container.Mushrooms[i].ReverseName(container.Mushrooms[i].Name);

}

}

enum ClassMush

{

Poisonous,

Edible,

Inedible

}

class Mushroom : ImyInterface

{

public string Name { get; set; }

public int Size { get; set; }

public ClassMush? ClassMush { get; set; }

public Mushroom()

{

}

public Mushroom(string name, int size, ClassMush classMush) :base()

{

Name = name;

Size = size;

ClassMush = classMush;

}

public void Print() => Console.WriteLine($"Name: {Name}, Size: {Size}, Class: {ClassMush};");

public string ReverseName(string name)

{

return name.Reverse().ToString();

}

}

class MushroomContainer

{

public List<Mushroom> Mushrooms { get; set; }

public MushroomContainer()

{

}

public MushroomContainer(int size) => Mushrooms = new List<Mushroom>(size);

public MushroomContainer(List<Mushroom> mushrooms) => Mushrooms = mushrooms;

public void AddMushroom(Mushroom obj)

{

Mushrooms.Add(obj);

}

public void Print()

{

foreach (var item in Mushrooms)

{

item.Print();

}

}

public void deleteByIndex(int index)

{

Mushrooms.RemoveAt(index);

}

public void DeleteByName(string name)

{

Mushrooms.RemoveAll(x => x.Name == name);

}

public void FillArray(int size)

{

Mushrooms = new List<Mushroom>(size + 1);

for (int i = 0; i < size;)

{

try

{

Console.WriteLine("Enter name: ");

string name = Console.ReadLine();

Console.WriteLine("Enter size: ");

int size1 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter class: ");

ClassMush classMush = (ClassMush)Enum.Parse(typeof(ClassMush), Console.ReadLine());

Mushrooms.Add(new Mushroom(name, size1, classMush));

i++;

}

catch (Exception e)

{

Console.WriteLine("Error: " + e.Message + " Try change input data");

}

}

}

public int getSize()

{

return Mushrooms.Count();

}

}

class PoisenousMushroom : Mushroom

{

public string PoisonousSubstance { get; set; }

public int dangerLevel { get; set; }

public PoisenousMushroom()

{

}

public PoisenousMushroom(string name, int size, ClassMush classMush) : base(name, size, classMush)

{

}

}

class Poganka : PoisenousMushroom, ImyInterface

{

public int Population { get; set; }

public Poganka()

{

}

public Poganka(string name, int size, ClassMush classMush, int population) : base(name, size, classMush)

{

PoisonousSubstance = "Amanitin";

dangerLevel = 5;

Population = population;

}

}

class InedibleMushroom : Mushroom

{

public string Reason { get; set; }

public InedibleMushroom()

{

}

public InedibleMushroom(string name, int size, ClassMush classMush, string reason) : base(name, size, classMush)

{

Reason = reason;

}

}

class redMush : InedibleMushroom, ImyInterface

{

public string howUtilyze { get; set; }

public redMush()

{

}

public redMush(string name, int size, ClassMush classMush, string reason, string howUtilyze) : base(name, size,

classMush, reason)

{

this.howUtilyze = howUtilyze;

}

}

class EdibleMushroom : Mushroom

{

public string callorize { get; set; }

public EdibleMushroom()

{

}

public EdibleMushroom(string name, int size, ClassMush classMush, string recipe) : base(name, size, classMush)

{

callorize = recipe;

}

}

class WhiteMush : EdibleMushroom, ImyInterface

{

public string reciept { get; set; }

public WhiteMush()

{

}

public WhiteMush(string name, int size, ClassMush classMush, string recipe, string reciept) : base(name, size,

classMush, recipe)

{

this.reciept = reciept;

}

}

}

// interface

interface ImyInterface

{

public void Print();

public string Name { get; set; }

public string ReverseName(string name) => name.Reverse().ToString();

}

// class that realise my interface

public class Realiseclass : ImyInterface

{

public string Name

{

get => Name;

set => Name = value;

}

public void Print()

{

Console.WriteLine(Name);

}

public string ReverseName(string name)

{

return name.Reverse().ToString();

}

}

# 

Рисунок 1.5 – Результат выполнения программы 1

Вывод: в ходе выполнения данной лабораторной работы номер 4, мы научились организовывать структуру файла, создавать пользовательские методы и организовывать их взаимодействие с полями класса. Научились на практике использовать наследование классов и методов , переопределение методов и организацию конструкторов, что позволит в будущем создавать нам более сложные и компле5ксные программы для решения конкретных задач.