МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Казанский национальный исследовательский технический

университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации

*(наименование института (факультета)*

Кафедра динамики процессов и управления

*(наименование кафедры)*

Лабораторная работа № 2

По дисциплине «Основы алгоритмизации и языки программирования»

|  |  |
| --- | --- |
| Выполнил: | обучающийся группы\_4165\_ |
|  | Шестаков М.Д. |

Казань 2025 г.

**Задание**

Освоить абстрактные классы, интерфейсы, виртуальные методы, наследование, инкапсуляцию, полиморфизм.

Создать абстрактный класс Animal с абстрактным методом MakeSound() и виртуальным Move().

Реализовать два класса: Dog и Cat, которые наследуют Animal.

Добавить интерфейс IPet с методом Play() и свойством Name. Dog и Cat реализуют IPet.

Написать простой менеджер Shelter (хранит List<Animal>), который вызывает у всех животных MakeSound() и Move().

Обработать некорректные входные данные (пустое имя) — бросать ArgumentException.

Пояснения:

Animal — абстрактный класс (абстрактный метод MakeSound и виртуальный Move).

Dog и Cat — наследники, реализуют IPet (интерфейс).

Shelter демонстрирует полиморфное поведение: один и тот же вызов MakeSound() вызывает разные реализации.

Валидация (инкапсуляция) — пустое имя/вид вызывает ArgumentException.

**Листинг кода**

Класс Program:

namespace StudyClass  
{  
 internal static class Program  
 {  
 private static void Main()  
 {  
 try  
 {  
 var shelter = new Shelter();  
  
 var dog = new Dog("Шарик");  
 var cat = new Cat("Мурзик");  
  
 shelter.AddAnimal(dog);  
 shelter.AddAnimal(cat);  
  
 shelter.ShowAll();  
  
 dog.Play();  
 cat.Play();  
 }  
 catch (ArgumentException ex)  
 {  
 Console.WriteLine($"Ошибка: {ex.Message}");  
 }  
 }  
 }  
}

Интерфейс IPet:

namespace StudyClass  
{  
 public interface IPet  
 {  
 string Name { get; }  
 void Play();  
 }  
}

Абстрактный класс Animal:

namespace StudyClass  
{  
 public abstract class Animal  
 {  
 protected Animal(string species)  
 {  
 if (string.IsNullOrWhiteSpace(species))  
 throw new ArgumentException("Вид животного не может быть пустым.", nameof(species));  
  
 Species = species;  
 }  
  
 private string Species { get; }  
  
 public abstract void MakeSound();  
  
 public virtual void Move()  
 {  
 Console.WriteLine($"{Species} движется...");  
 }  
 }  
}

Классы Dog и Cat:

public class Cat : Animal, IPet  
{  
 public Cat(string name) : base("Кошка")  
 {  
 if (string.IsNullOrWhiteSpace(name))  
 throw new ArgumentException("Имя питомца не может быть пустым.", nameof(name));  
  
 Name = name;  
 }  
  
 public string Name { get; }  
  
 public override void MakeSound()  
 {  
 Console.WriteLine($"{Name} говорит: Мяу-мяу!");  
 }  
  
 public override void Move()  
 {  
 Console.WriteLine($"{Name} крадётся на мягких лапках.");  
 }  
  
 public void Play()  
 {  
 Console.WriteLine($"{Name} играет с клубком ниток.");  
 }  
}

namespace StudyClass  
{  
 public class Dog : Animal, IPet  
 {  
 public Dog(string name) : base("Собака")  
 {  
 if (string.IsNullOrWhiteSpace(name))  
 throw new ArgumentException("Имя питомца не может быть пустым.", nameof(name));  
  
 Name = name;  
 }  
  
 public string Name { get; }  
  
 public override void MakeSound()  
 {  
 Console.WriteLine($"{Name} говорит: Гав-гав!");  
 }  
  
 public override void Move()  
 {  
 Console.WriteLine($"{Name} бегает на четырех лапах.");  
 }  
  
 public void Play()  
 {  
 Console.WriteLine($"{Name} играет с мячом.");  
 }  
 }  
}

Класс Shelter:

namespace StudyClass  
{  
 public class Shelter  
 {  
 private readonly List<Animal> \_animals = [];  
  
 public void AddAnimal(Animal animal)  
 {  
 ArgumentNullException.ThrowIfNull(animal);  
 \_animals.Add(animal);  
 }  
  
 public void ShowAll()  
 {  
 foreach (var animal in \_animals)  
 {  
 animal.MakeSound();  
 animal.Move();  
 }  
 }  
 }  
}

**Объяснение работы программы**

Программа реализует простую модель приюта для животных. В основе лежит абстрактный класс Animal, который определяет общий интерфейс для всех животных: абстрактный метод MakeSound(), обязательный к переопределению в наследниках, и виртуальный метод Move(), который можно изменить при необходимости. От него наследуются классы Dog и Cat, каждый из которых реализует собственное поведение — собака лает и бегает, а кошка мяукает и крадется. Оба класса дополнительно реализуют интерфейс IPet, содержащий свойство Name и метод Play(), что позволяет работать с ними как с домашними питомцами. В конструкторах предусмотрена валидация, и при передаче пустого имени выбрасывается ArgumentException, что демонстрирует инкапсуляцию и защиту от некорректных данных. Класс Shelter представляет собой менеджер приюта, внутри которого хранится список животных и есть метод для их отображения, вызывающий MakeSound() и Move() у каждого объекта, что наглядно показывает полиморфизм — одинаковый вызов приводит к разным результатам в зависимости от типа животного. В точке входа Main создается объект Shelter, в него добавляются собака и кошка, после чего демонстрируется работа приюта, а также выводится реакция животных на игру. Таким образом, программа объединяет использование абстрактных классов, интерфейсов, полиморфизма и обработку ошибок, создавая гибкую и расширяемую структуру.