

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования
"Омский государственный технический университет"

Кафедра «Автоматизированные системы обработки информации и управления»

ПРАКТИЧЕСКАЯ РАБОТА №4

по дисциплине

«Измерительные средства аналитики программных систем и технологий»

по теме

«Метрика Мартина.»

Выполнил:

Шмидт А.В.

гр. ИВТ - 244

Проверил:

Зубарев А.А.

к.т.н., доцент

Омск 2025

ЗАДАНИЕ

1. Выбрать программу, ОБЯЗАТЕЛЬНО реализованную в парадигме ООП на знакомом языке высокого уровня. Разобрать её работу, и представить схему алгоритма программы.
2. Представить графически и кратко описать «диаграмму классов программы» в нотации языка UML (Unified Modeling Language - унифицированный язык моделирования).
3. Рассчитать пять метрик Мартина для выбранной программы. Программа индивидуальна для каждого студента. Результаты расчётов внести в итоговую сводную таблицу.
4. На основе метрик Мартина построить рисунок, на котором показана линия как геометрическое место точек, являющихся суммой абстрактности и нестабильности компонентов программы. На рисунке также показать линию, заданную суммой абстрактности и нестабильности, равной «1» (линия «главной последовательности»). По полученным линиям пояснить сбалансированность между абстрактностью и нестабильностью программы.

ПРИМЕР ОПРЕДЕЛЕНИЯ МЕТРИК ПО МАРТИНУ

Программа на языке C# реализует модель мини-зоопарка в объектно-ориентированной парадигме. Содержит абстрактный класс Animal, интерфейс IFeedable, конкретные классы Lion, Elephant, Parrot и управляющий класс Zoo.

Программа создаёт животных, добавляет их в зоопарк, выводит информацию о них, демонстрирует полиморфизм при издании звуков и selective кормление через интерфейс.

Ниже представлен листинг программы.

```
namespace code;

using System;
using System.Collections.Generic;

// Абстрактный класс — базовый для всех животных
public abstract class Animal
{
    public string Name { get; set; }
    public int Age { get; set; }
    public string Species { get; set; }

    public Animal(string name, int age, string species)
    {
        Name = name;
        Age = age;
        Species = species;
    }

    // Абстрактный метод — должен быть переопределён в наследниках
    public abstract void MakeSound();

    // Виртуальный метод — можно переопределить
    public virtual void ShowInfo()
    {
        Console.WriteLine($"Животное: {Name}, Вид: {Species}, Возраст: {Age}");
    }
}

// Интерфейс для животных, которых можно кормить особым образом
public interface IFeedable
{
    void Feed(string food);
}

// Конкретные классы животных
public class Lion : Animal
{
    public Lion(string name, int age) : base(name, age, "Лев")
    {
    }
}
```

```
public override void MakeSound()
{
    Console.WriteLine($"{Name} рычит: ППППП!");
}

public class Elephant : Animal, IFeedable
{
    public Elephant(string name, int age) : base(name, age, "Слон")
    {

    }

    public override void MakeSound()
    {
        Console.WriteLine($"{Name} трубит: Ту-ду-ду!");
    }

    public void Feed(string food)
    {
        Console.WriteLine($"{Name} ест {food}. Спасибо!");
    }
}

public class Parrot : Animal
{
    public string Color { get; set; }

    public Parrot(string name, int age, string color) : base(name, age, "Попугай")
    {
        Color = color;
    }

    public override void MakeSound()
    {
        Console.WriteLine($"{Name} кричит: Каррр! Красиво, да?!");
    }

    public override void ShowInfo()
    {
        base.ShowInfo();
        Console.WriteLine($"Цвет оперения: {Color}");
    }
}

// Класс зоопарка — управляет коллекцией животных
public class Zoo
{
    private List<Animal> animals = new List<Animal>();

    public void AddAnimal(Animal animal)
    {
```

```

        animals.Add(animal);
        Console.WriteLine($" {animal.Name} добавлен в зоопарк.");
    }

    public void MakeAllSounds()
    {
        Console.WriteLine("\nВсе животные издают звуки:");
        foreach (var animal in animals)
        {
            animal.MakeSound();
        }
    }

    public void ShowAllAnimals()
    {
        Console.WriteLine("\nСписок животных в зоопарке:");
        foreach (var animal in animals)
        {
            animal.ShowInfo();
        }
    }

    // Кормим только тех, кто реализует IFeedable
    public void FeedAnimal(string name, string food)
    {
        var animal = animals.Find(a => a.Name == name);
        if (animal is IFeedable feedable)
        {
            feedable.Feed(food);
        }
        else
        {
            Console.WriteLine($"{name} не хочет это есть или не умеет так кормиться.");
        }
    }
}

// Точка входа — демонстрация работы
class Program
{
    static void Main(string[] args)
    {
        Zoo zoo = new Zoo();

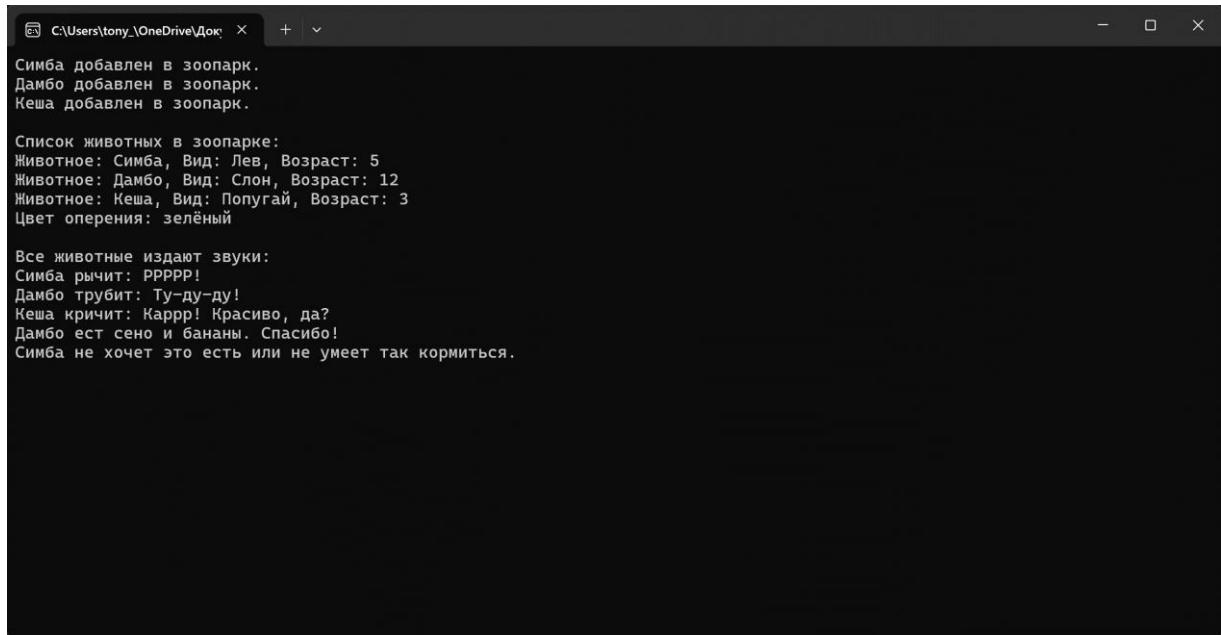
        zoo.AddAnimal(new Lion("Симба", 5));
        zoo.AddAnimal(new Elephant("Дамбо", 12));
        zoo.AddAnimal(new Parrot("Кеша", 3, "зелёный"));

        zoo.ShowAllAnimals();
        zoo.MakeAllSounds();

        zoo.FeedAnimal("Дамбо", "сено и бананы");
    }
}

```

```
zoo.FeedAnimal("Симба", "мясо"); // Лев не реализует IFeedable → сообщение  
Console.ReadKey();  
}  
}
```



The screenshot shows a terminal window titled 'C:\Users\tony_\OneDrive\Док' with the following text output:

```
Симба добавлен в зоопарк.  
Дамбо добавлен в зоопарк.  
Кеша добавлен в зоопарк.  
  
Список животных в зоопарке:  
Животное: Симба, Вид: Лев, Возраст: 5  
Животное: Дамбо, Вид: Слон, Возраст: 12  
Животное: Кеша, Вид: Попугай, Возраст: 3  
Цвет оперения: зелёный  
  
Все животные издают звуки:  
Симба рычит: РРРРР!  
Дамбо трубит: Ту-ду-ду!  
Кеша кричит: Каррр! Красиво, да?  
Дамбо ест сено и бананы. Спасибо!  
Симба не хочет это есть или не умеет так кормиться.
```

Рисунок 1 – Результат работы программы.

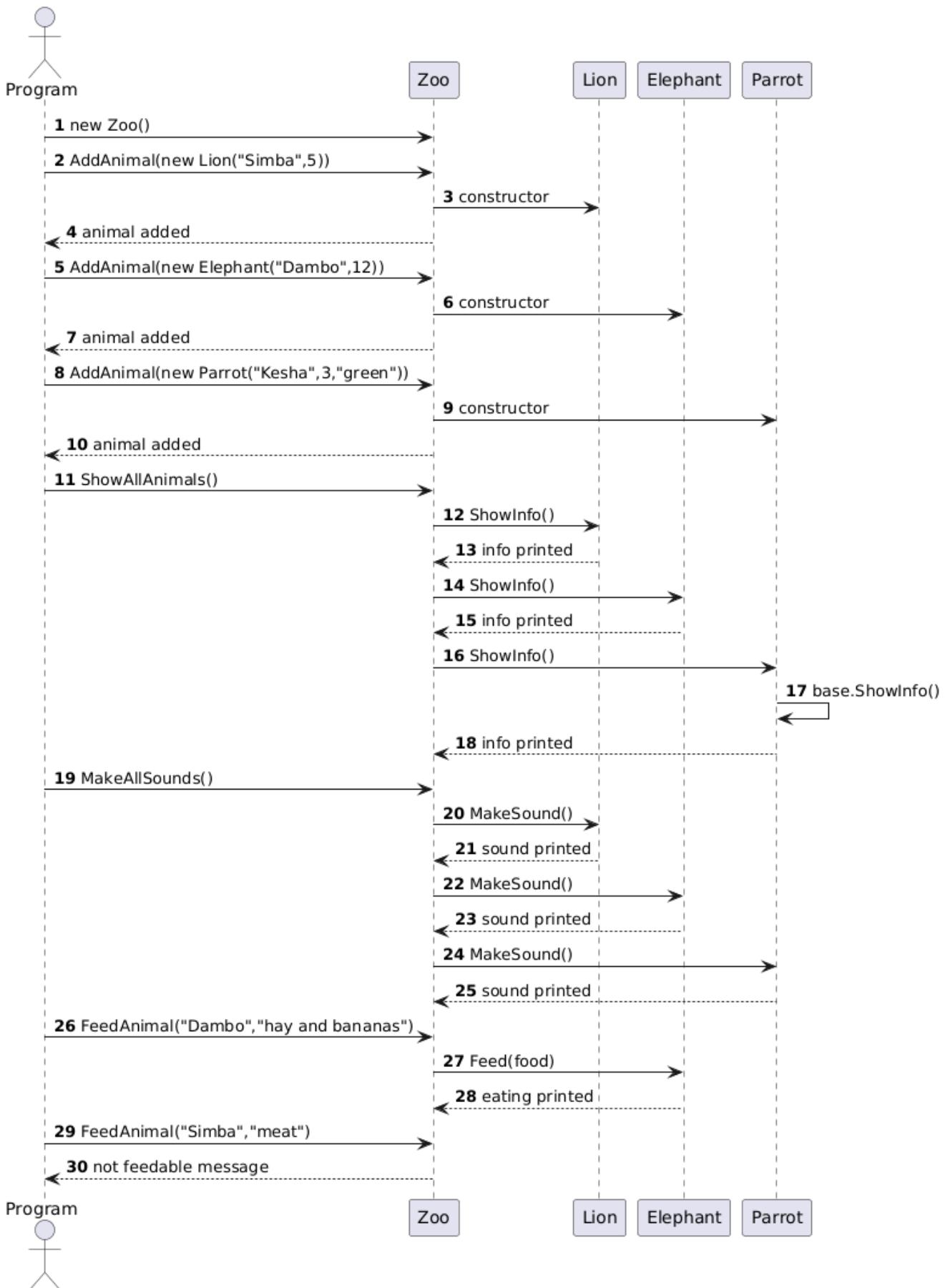


Рисунок 2 – Диаграмма классов программы.

Таблица 1 – Подсчёт метрик Мартина.

Класс	Се (Исходя щие)	Са (Входя щие)	Нестабильн ость I	Абстрактно сть A (для класса)	Комментарий
Animal	0	5	0	1	Максимально стабилен и абстрактен (Идеал).
IFeedable	0	2	0	1	Максимально стабилен и абстрактен (Идеал).
Lion	1	1	0.5	0	Средняя устойчивость.
Elephant	2	1	0.67	0	Склонен к нестабильности.
Parrot	1	1	0.5	0	Средняя устойчивость.
Zoo	2	1	0.67	0	Склонен к нестабильности.
Program	4	0	1.0	0	Максимально нестабилен (норма для Main).
ВСЯ ПРОГРАММА	-	-	1.0*	0.29**	Dn=0.29

Нестабильность (I):

Согласно методичке, для программы берется наибольшее значение нестабильности среди классов, так как это "зона риска"⁸. В данном случае I_max = 1.0 (класс Program).

Абстрактность (A):

$$n_A = 2 \text{ (Animal, IFeedable)}, n_{All} = 7.$$

$$A = 2 / 7 = 0.286.$$

Расстояние (D_n):

$$D_n = |0.286 + 1.0 - 1| = 0.286 = 0.29.$$

Ниже приведён график баланса абстрактности и нестабильности по подсчитанным ранее метрикам Мартина.

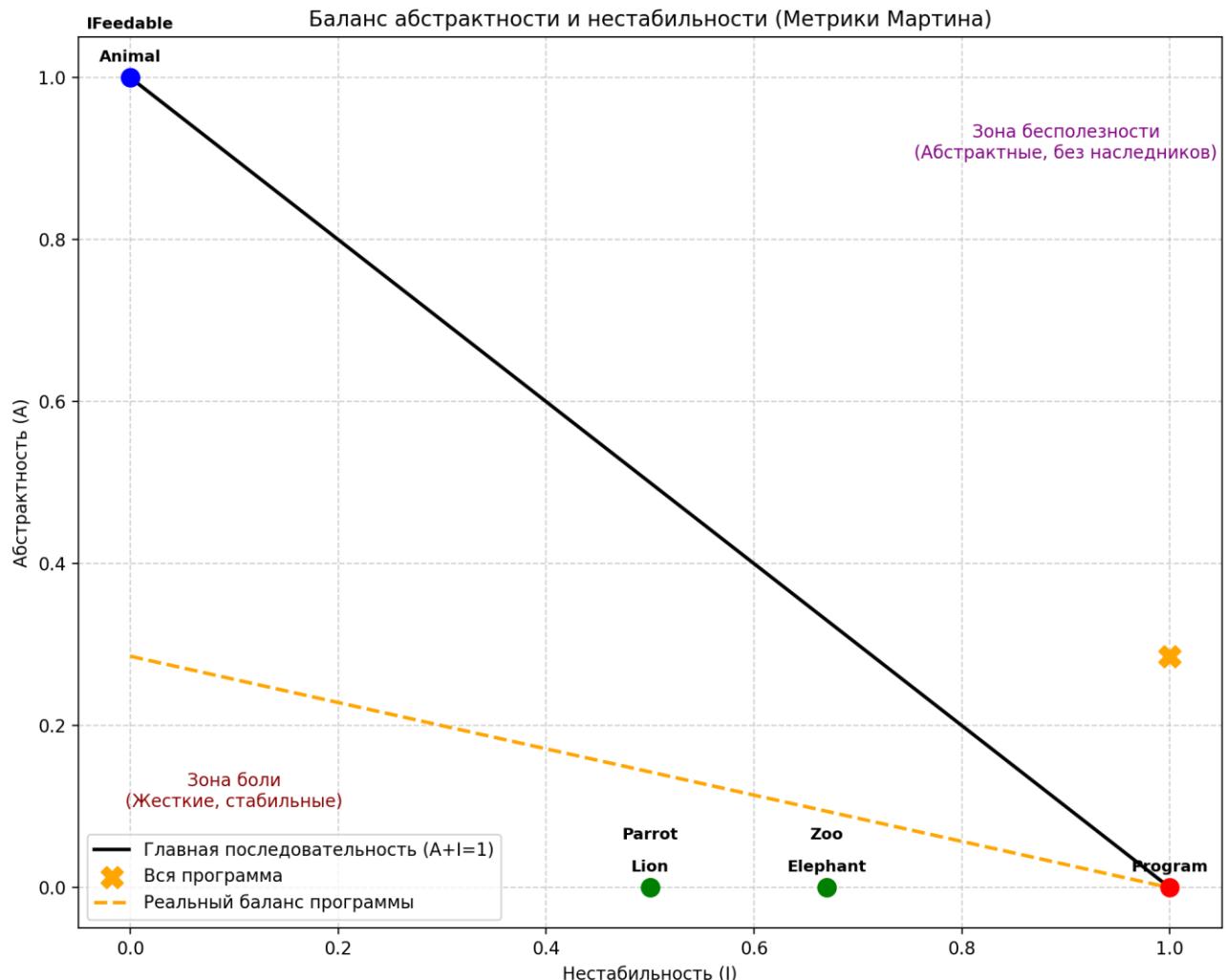


Рисунок 3 – График по метрикам Мартина.

ВЫВОД

Оценка программы по метрикам Мартина показала хорошую сбалансированность архитектуры: нормализованное расстояние до главной последовательности составляет 0,29, что свидетельствует о близости к идеальному балансу. Классы программы не попадают в «зону боли» или «зону бесполезности». Четкое разделение на стабильные абстракции (Animal, IFeedable) и конкретные реализации обеспечивает гибкость системы, позволяя расширять функционал без нарушения работы существующих компонентов.