**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Математическая кибернетика и информационные технологии

Отчет по Лабораторной работе №2

По дисциплине: Информационные технологии и программирование

Выполнил: студент группы БВТ-2401

Язалиев Тимур Исламович

Руководитель: Харрасов Камиль Раисович

Москва, 2025

**Отчет по лабораторной работе №2**

Тема: Объектно-ориентированное программирование

Вариант: 1

1. Цель работы

Изучить основы объектно-ориентированного программирования (ООП) в Java, включая инкапсуляцию, наследование, полиморфизм и абстракцию. Создать иерархию классов, продемонстрировать их взаимодействие и использование принципов ООП.

2. Индивидуальное задание

Создать иерархию классов:

1)Базовый класс: Животные

2)Дочерние классы: Кошка, Попугай, Рыбка

Требования:

· Использовать абстрактный класс.

· Реализовать два уровня наследования.

· Каждый класс должен содержать не менее 3 полей и 2 методов.

· Продемонстрировать все принципы ООП.

· Добавить конструкторы (включая конструктор по умолчанию), геттеры и сеттеры.

· Реализовать ввод/вывод информации о создаваемых объектах.

· Добавить счетчик созданных объектов с использованием статической переменной.

3.Основная часть

package Lab\_2;  
  
import java.util.Scanner;  
  
abstract class Animal {  
 protected String name;  
 protected int age;  
 protected String habitat;  
   
 public Animal() {  
 this.name = "Неизвестно";  
 this.age = 0;  
 this.habitat = "Неизвестно";  
 }  
   
 public Animal(String name, int age, String habitat) {  
 this.name = name;  
 this.age = age;  
 this.habitat = habitat;  
 }  
   
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public String getHabitat() {  
 return habitat;  
 }  
  
 public void setHabitat(String habitat) {  
 this.habitat = habitat;  
 }  
   
 public abstract void makeSound();  
   
 public void displayInfo() {  
 System.*out*.println("Имя: " + name);  
 System.*out*.println("Возраст: " + age + " лет/год(а)");  
 System.*out*.println("Среда обитания: " + habitat);  
 }  
}  
  
class Cat extends Animal {  
 private String breed;  
   
 private static int *catCount* = 0;  
  
 // Конструктор по умолчанию  
 public Cat() {  
 super();  
 this.breed = "Неизвестно";  
 *catCount*++;  
 }  
   
 public Cat(String name, int age, String habitat, String breed) {  
 super(name, age, habitat);  
 this.breed = breed;  
 *catCount*++;  
 }  
   
 public String getBreed() {  
 return breed;  
 }  
  
 public void setBreed(String breed) {  
 this.breed = breed;  
 }  
   
 @Override  
 public void makeSound() {  
 System.*out*.println(name + " говорит: Мяу!");  
 }  
   
 @Override  
 public void displayInfo() {  
 super.displayInfo();  
 System.*out*.println("Порода: " + breed);  
 }  
   
 public static int getCatCount() {  
 return *catCount*;  
 }  
}  
  
class Parrot extends Animal {  
 private boolean canTalk;  
   
 private static int *parrotCount* = 0;  
   
 public Parrot() {  
 super();  
 this.canTalk = false;  
 *parrotCount*++;  
 }  
   
 public Parrot(String name, int age, String habitat, boolean canTalk) {  
 super(name, age, habitat);  
 this.canTalk = canTalk;  
 *parrotCount*++;  
 }  
   
 public boolean getCanTalk() {  
 return canTalk;  
 }  
  
 public void setCanTalk(boolean canTalk) {  
 this.canTalk = canTalk;  
 }  
   
 @Override  
 public void makeSound() {  
 if (canTalk) {  
 System.*out*.println(name + " говорит: Привет!");  
 } else {  
 System.*out*.println(name + " говорит: Чирик-чирик!");  
 }  
 }  
   
 @Override  
 public void displayInfo() {  
 super.displayInfo();  
 System.*out*.println("Умеет говорить: " + (canTalk ? "Да" : "Нет"));  
 }  
   
 public static int getParrotCount() {  
 return *parrotCount*;  
 }  
}  
  
class Fish extends Animal {  
 private String waterType;  
   
 private static int *fishCount* = 0;  
   
 public Fish() {  
 super();  
 this.waterType = "Пресная";  
 *fishCount*++;  
 }  
   
 public Fish(String name, int age, String habitat, String waterType) {  
 super(name, age, habitat);  
 this.waterType = waterType;  
 *fishCount*++;  
 }  
   
 public String getWaterType() {  
 return waterType;  
 }  
  
 public void setWaterType(String waterType) {  
 this.waterType = waterType;  
 }  
   
 @Override  
 public void makeSound() {  
 System.*out*.println(name + " говорит: (пускает пузыри)");  
 }  
   
 @Override  
 public void displayInfo() {  
 super.displayInfo();  
 System.*out*.println("Тип воды: " + waterType);  
 }  
   
 public void swim() {  
 System.*out*.println(name + " плавает в воде");  
 }  
   
 public static int getFishCount() {  
 return *fishCount*;  
 }  
}  
  
public class lab\_1 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
   
 Cat cat1 = new Cat();  
 Parrot parrot1 = new Parrot();  
 Fish fish1 = new Fish();  
   
 System.*out*.println("=== Ввод информации о животных ===");  
 System.*out*.println("\n--- Ввод информации о кошке ---");  
 System.*out*.print("Введите имя кошки: ");  
 cat1.setName(scanner.nextLine());  
 System.*out*.print("Введите возраст кошки: ");  
 cat1.setAge(scanner.nextInt());  
 scanner.nextLine();   
 System.*out*.print("Введите среду обитания кошки: ");  
 cat1.setHabitat(scanner.nextLine());  
 System.*out*.print("Введите породу кошки: ");  
 cat1.setBreed(scanner.nextLine());  
   
 System.*out*.println("\n--- Ввод информации о попугае ---");  
 System.*out*.print("Введите имя попугая: ");  
 parrot1.setName(scanner.nextLine());  
 System.*out*.print("Введите возраст попугая: ");  
 parrot1.setAge(scanner.nextInt());  
 scanner.nextLine();  
 System.*out*.print("Введите среду обитания попугая: ");  
 parrot1.setHabitat(scanner.nextLine());  
 System.*out*.print("Попугай умеет говорить? (true/false): ");  
 parrot1.setCanTalk(scanner.nextBoolean());  
 scanner.nextLine();   
   
 System.*out*.println("\n--- Ввод информации о рыбке ---");  
 System.*out*.print("Введите имя рыбки: ");  
 fish1.setName(scanner.nextLine());  
 System.*out*.print("Введите возраст рыбки: ");  
 fish1.setAge(scanner.nextInt());  
 scanner.nextLine();   
 System.*out*.print("Введите среду обитания рыбки: ");  
 fish1.setHabitat(scanner.nextLine());  
 System.*out*.print("Введите тип воды для рыбки: ");  
 fish1.setWaterType(scanner.nextLine());  
   
 Cat cat2 = new Cat("Барсик", 3, "Дом", "Сиамская");  
 Parrot parrot2 = new Parrot("Кеша", 2, "Джунгли", true);  
 Fish fish2 = new Fish("Немо", 1, "Аквариум", "Соленая");  
   
 System.*out*.println("\n=== Информация о животных ===");  
   
 Animal[] animals = {cat1, parrot1, fish1, cat2, parrot2, fish2};  
  
 for (Animal animal : animals) {  
 System.*out*.println("\n--- " + animal.getClass().getSimpleName() + " ---");  
 animal.displayInfo();  
 animal.makeSound();  
   
 if (animal instanceof Fish) {  
 ((Fish) animal).swim();  
 }  
 }  
   
 System.*out*.println("\n=== Статистика созданных объектов ===");  
 System.*out*.println("Количество созданных кошек: " + Cat.*getCatCount*());  
 System.*out*.println("Количество созданных попугаев: " + Parrot.*getParrotCount*());  
 System.*out*.println("Количество созданных рыбок: " + Fish.*getFishCount*());  
 System.*out*.println("Общее количество животных: " + (Cat.*getCatCount*() + Parrot.*getParrotCount*() + Fish.*getFishCount*()));  
  
 scanner.close();  
 }  
}

4. Заключение

В ходе выполнения лабораторной работы:

· Изучены принципы ООП: инкапсуляция, наследование, полиморфизм и абстракция.

· Создана иерархия классов с использованием абстрактного класса и наследования.

· Реализованы конструкторы, геттеры, сеттеры и статический счетчик объектов.

· Продемонстрирована работа программы с выводом информации о созданных объектах.

Результаты работы подтверждают корректность реализации принципов ООП в Java.

*Ссылка на репозиторий: https://github.com/Maggistr/lab-1.git*