**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики».**

Кафедра «Математическая Кибернетика и Информационные технологии»

Лабораторная работа №1

Выполнил студент группы:

БВТ2402

Янкович Павле

Mосква 2025

**1. Цель работы**

Целью данной лабораторной работы является приобретение практических навыков в работе с объектно-ориентированным программированием на Java: создание иерархии классов, реализация принципов ООП, работа с абстрактными классами, наследованием, полиморфизмом и инкапсуляцией.

**2. Индивидуальные задания:**

Задание 1. Создайте иерархию классов в соответствии с вариантом. Ваша иерархия должна содержать:

• абстрактный класс;

• два уровня наследуемых классов (классы должны содержать в себе минимум 3 поля и 2 метода, описывающих поведение объекта);

• демонстрацию реализации всех принципов ООП;

• наличие конструкторов (в том числе по умолчанию);

• наличие геттеров и сеттеров;

• ввод/вывод информации о создаваемых объектах;

• предусмотрите в одном из классов создание счетчика созданных объектов с использованием статической переменной, продемонстрируйте работу.

Вариант: Базовый класс: Животные. Дочерние классы: Кошка, Попугай, Рыбка

import java.util.Scanner;

abstract class Animal{

    private String name;

    private int age;

    private String color;

    private static int count = 0; //brojac

    public Animal(){

        count++;

    }

    public Animal(String name, int age, String color){

        this.name = name;

        this.age = age;

        this.color = color;

        count++;

    }

    abstract void makeSound();

    abstract void eat();

    //Getters and setters

    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 getColor() {return color; }

    public void setColor(String color) {this.color = color;}

    public static int getCount() {return count;}

    //display info

    public void displayInfo(){

        System.out.println("Name: " + name + ", Age: " + age + ", Color: " + color);

    }

}

//class Cat

class Cat extends Animal{

    private String breed;

    //constructor

    public Cat(String name, int age, String color, String breed){

        super(name, age, color);

        this.breed = breed;

    }

    @Override

    void makeSound(){

        System.out.println(getName() + " says: Meow!");

    }

    @Override

    void eat(){

        System.out.println(getName() + " eats fish");

    }

    //specific method

    void catchMouse(){

        System.out.println(getName() + " cathes a mouse");

    }

    //getters and setters

    public String getBreed() {return breed;}

    public void setBreed(String breed) {this.breed = breed;}

}

//class Parrot

class Parrot extends Animal{

    private boolean canTalk;

    //constructor

    public Parrot(String name, int age, String color, boolean canTalk){

        super(name, age, color);

        this.canTalk = canTalk;

    }

    @Override

    void makeSound(){

        if(canTalk){

            System.out.println(getName() + " says: Hello!");

        }else{

            System.out.println(getName() + " chirps");

        }

    }

    @Override

    void eat(){

        System.out.println(getName() + " eats seeds");

    }

    //specific method

    void fly(){

        System.out.println(getName() + " flies");

    }

    //getters and setters

    public boolean canTalk() {return canTalk;}

    public void canTalk(boolean canTalk){this.canTalk = canTalk;}

}

//class Fish

class Fish extends Animal{

    private String waterType;

    //Constructor

    public Fish(String name, int age, String color, String waterType){

        super(name, age, color);

        this.waterType = waterType;

    }

    @Override

    void makeSound(){

        System.out.println(getName() + " makes bubbles");

    }

    @Override

    void eat(){

        System.out.println(getName() + " eats fish food");

    }

    //specific method

    void swim(){

        System.out.println(getName() + " swims");

    }

    //getters and setters

    public String getWaterType(){return waterType;}

    public void getWaterType(String waterType){this.waterType = waterType;}

}

class SaltWaterFish extends Fish{

    private String SaltWater;

    public SaltWaterFish(String name, int age, String color, String SaltWater){

        super(name, age, color, null);

        this.SaltWater = SaltWater;

    }

    @Override

    void makeSound(){

        System.out.println(getName() + " makes bubbles");

    }

    @Override

    void eat(){

        System.out.println(getName() + " eats fish food");

    }

    //specific method

    @Override

    void swim(){

        System.out.println(getName() + " swims");

    }

    //getters and setters

    public String getSaltWater(){return SaltWater;}

    public void getSaltWater(String SaltWater){this.SaltWater = SaltWater;}

}

public class Main{

    public static void main(String[]args){

        Scanner scanner = new Scanner(System.in);

        //data input

        System.out.println("---CREATING ANIMALS---");

        //creating a cat

        System.out.print("Enter cat name: ");

        String catName = scanner.nextLine();

        System.out.print("Enter cat age: ");

        int catAge = scanner.nextInt();

        scanner.nextLine(); //clear buffer

        System.out.print("Enter cat color: ");

        String catColor = scanner.nextLine();

        System.out.print("Enter cat breed: ");

        String catBreed = scanner.nextLine();

        Animal cat = new Cat(catName, catAge, catColor, catBreed);

        //creating a parrot

        System.out.print("\nEnter parrot name: ");

        String parrotName = scanner.nextLine();

        System.out.print("Enter parrot age: ");

        int parrotAge = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Enter parrot color: ");

        String parrotColor = scanner.nextLine();

        System.out.print("Can parrot talk (true/false): ");

        boolean canTalk = scanner.nextBoolean();

        scanner.nextLine();

        Animal parrot = new Parrot(parrotName, parrotAge, parrotColor, canTalk);

        //creating a fish

        System.out.print("\nEnter fish name: ");

        String fishName = scanner.nextLine();

        System.out.print("Enter fish age: ");

        int fishAge = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Enter fish color: ");

        String fishColor = scanner.nextLine();

        System.out.print("Enter water type for fish: ");

        String waterType = scanner.nextLine();

        Animal fish = new Fish(fishName, fishAge, fishColor, waterType);

        //creating a waterfish

        System.out.print("\nEnter waterfish name: ");

        String waterfishName = scanner.nextLine();

        System.out.print("Enter waterfish age: ");

        int waterfishAge = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Enter waterfish color: ");

        String waterfishColor= scanner.nextLine();

        System.out.print("Enter water type for waterfish: ");

        String waterfishType = scanner.nextLine();

        Animal waterfish = new SaltWaterFish(waterfishName, waterfishAge, waterfishColor, waterfishType);

        //output information

        System.out.print("\n=== Animal information ===\n");

        cat.displayInfo();

        cat.makeSound();

        cat.eat();

        ((Cat)cat).catchMouse();

        System.out.println();

        parrot.displayInfo();

        parrot.makeSound();

        parrot.eat();

        ((Parrot)parrot).fly();

        System.out.println();

        fish.displayInfo();

        fish.makeSound();

        fish.eat();

        ((Fish)fish).swim();

        System.out.println();

        waterfish.displayInfo();

        waterfish.makeSound();

        waterfish.eat();

        ((SaltWaterFish)waterfish).swim();

        //counter demonstration

        System.out.print("\n=== Statistics ===\n");

        System.out.println("Total animals created: " + Animal.getCount());

        System.out.print("\n=== OOP Principles Demonstration ===\n");

        //Inheritance - all classes inherit from Animal

        //polymorphism - same method makeSound() works differently

        Animal[] animals = {cat, parrot, fish, waterfish};

        for( Animal animal : animals){

            animal.makeSound(); //each makes his own sound

        }

        //encapsulation - using getters

        System.out.println("Names of all animals: ");

        for(Animal animal : animals){

            System.out.println(animal.getName());

        }

        scanner.close();

    }

}

**Описание решения:**

1. **Абстрактный класс Animal**:
   * Содержит общие поля: name, age, color
   * Имеет абстрактные методы makeSound() и eat()
   * Реализует статический счетчик count для подсчета созданных объектов
2. **Классы-наследники**:
   * **Cat**: добавляет поле breed и метод catchMouse()
   * **Parrot**: добавляет поле, canTalk и метод fly()
   * **Fish**: добавляет поле waterType и метод swim()
3. **Принципы ООП**:
   * **Наследование**: все классы наследуют от Animal
   * **Инкапсуляция**: private поля с public геттерами/сеттерами
   * **Полиморфизм**: метод makeSound() работает по-разному для каждого животного
   * **Абстракция**: абстрактный класс и методы

**4. Заключение**

В ходе выполнения лабораторной работы была успешно реализована иерархия классов животных, включающая:

1. Абстрактный класс Animal с общими характеристиками и поведением
2. Три класса-наследника: Cat, Parrot, Fish с уникальными полями и методами
3. Реализацию всех принципов ООП: наследования, инкапсуляции, полиморфизма и абстракции
4. Систему подсчета созданных объектов с использованием статической переменной
5. Интерфейс для ввода/вывода информации о животных

Работа позволила закрепить практические навыки объектно-ориентированного программирования на Java, включая работу с абстрактными классами, наследованием, переопределением методов и организацией сложных программных структур.

Ссылка на github: https://github.com/Pavle032/it3semlabi