**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»**

Факультет «Информационные технологии»

Кафедра «Математическая кибернетика и информационные технологии»

Дисциплина: «Информационные технологии и программирование»

**Лабораторная работа №1**

**«Типы данных в Java»**

Выполнил:

студент группы БВТ2402

Аралушкин Максим Дмитриевич

Руководитель:

Мосева Марина Сергеевна

Москва

2025

# Цель работы

Освоение объектно-ориентировочного программирования на Java. Изучение работы классов и объектов.

# Индивидуальное задание

Создание иерархии классов:

Базовый класс: Животные

Дочерние классы: Кошка, Попугай, Рыбка

Иерархия содержит:

* 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.

# Выполнение

Создаем абстрактный класс «Animal» с приватными переменными. Создаем функции получения и вноса информации:

abstract class Animal {

    private String name;

    private int age;

    private String color;

    private static int count = 0;

    public Animal() {

        count++;

    }

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

        this.name = name;

        this.age = age;

        this.color = color;

        count++;

    }

    public abstract void makeSound();

    public abstract void eat();

    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;

    }

    public void sleep() {

        System.out.println(name + " спит");

    }

    public void displayInfo() {

        System.out.println("Имя: " + name + ", Возраст: " + age + ", Цвет: " + color);

    }

}

Создаем дочерний класс «Cat», вводим приватные параметры и геттеры, сеттеры. Создаем функции для поведения класса:

class Cat extends Animal {

    private String breed;

    private String likesFish;

    private int lives;

    public Cat() {

        super();

        this.lives = 9;

    }

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

        super(name, age, color);

        this.breed = breed;

        this.likesFish = likesFish;

        this.lives = 9;

    }

    @Override

    public void makeSound() {

        System.out.println(getName() + ": Мяу-мяу~");

    }

    public void makeSound(String sound) {

        System.out.println(getName() + ": " + sound);

    }

    @Override

    public void eat() {

        if (likesFish.equals("yes")) {

            System.out.println(getName() + " ест рыбу");

        } else {

            System.out.println(getName() + " ест корм");

        }

    }

    public void play() {

        System.out.println(getName() + " играет с мячиком");

    }

    public void scratch() {

        System.out.println(getName() + " царапает диван");

    }

    public String getBreed() {

        return breed;

    }

    public void setBreed(String breed) {

        this.breed = breed;

    }

    public String getLikesFish() {

        return likesFish;

    }

    public void setLikesFish(String likesFish) {

        this.likesFish = likesFish;

    }

    public int getLives() {

        return lives;

    }

    public void setLives(int lives) {

        this.lives = lives;

    }

    @Override

    public void displayInfo() {

        super.displayInfo();

        System.out.println("Порода: " + breed + ", Любит рыбу: " + likesFish + ", Жизней: " + lives);

    }

}

Создаем дочерний класс «HomeCat», вводим нужные функции:

class HomeCat extends Cat {

    private String owner;

    private boolean hasHome;

    public HomeCat() {

        super();

        this.hasHome = true;

    }

    public HomeCat(String name, int age, String color, String breed, String likesFish, String owner) {

        super(name, age, color, breed, likesFish);

        this.owner = owner;

        this.hasHome = true;

    }

    @Override

    public void makeSound() {

        System.out.println(getName() + ": Mррр~");

    }

    public void sitOnSofa() {

        System.out.println(getName() + " лежит на диване");

    }

    public String getOwner() {

        return owner;

    }

    public void setOwner(String owner) {

        this.owner = owner;

    }

    public boolean getHasHome() {

        return hasHome;

    }

    public void setHasHome(boolean hasHome) {

        this.hasHome = hasHome;

    }

    @Override

    public void displayInfo() {

        super.displayInfo();

        System.out.println("Хозяин: " + owner + ", Есть дом: " + hasHome);

    }

}

Создаем дочерний класс «StreetCat», вводим нужные функции:

class StreetCat extends Cat {

    private String area;

    private boolean hasFriends;

    public StreetCat() {

        super();

    }

    public StreetCat(String name, int age, String color, String breed, String likesFish, String area, boolean hasFriends) {

        super(name, age, color, breed, likesFish);

        this.area = area;

        this.hasFriends = hasFriends;

    }

    @Override

    public void makeSound() {

        System.out.println(getName() + " мяукает на улице");

    }

    public void walkAtNight() {

        System.out.println(getName() + " гуляет ночью в районе " + area);

    }

    public String getArea() {

        return area;

    }

    public void setArea(String area) {

        this.area = area;

    }

    public boolean getHasFriends() {

        return hasFriends;

    }

    public void setHasFriends(boolean hasFriends) {

        this.hasFriends = hasFriends;

    }

    @Override

    public void displayInfo() {

        super.displayInfo();

        System.out.println("Район: " + area + ", Есть друзья: " + hasFriends);

    }

}

Создаем дочерний класс «Parrot»:

class Parrot extends Animal {

    private boolean canTalk;

    private String favoriteWord;

    public Parrot() {

        super();

    }

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

        super(name, age, color);

        this.canTalk = canTalk;

        this.favoriteWord = favoriteWord;

    }

    @Override

    public void makeSound() {

        if (canTalk) {

            System.out.println(getName() + " говорит: " + favoriteWord);

        } else {

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

        }

    }

    @Override

    public void eat() {

        System.out.println(getName() + " ест зерно");

    }

    public void fly() {

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

    }

    public boolean getCanTalk() {

        return canTalk;

    }

    public void setCanTalk(boolean canTalk) {

        this.canTalk = canTalk;

    }

    public String getFavoriteWord() {

        return favoriteWord;

    }

    public void setFavoriteWord(String favoriteWord) {

        this.favoriteWord = favoriteWord;

    }

    @Override

    public void displayInfo() {

        super.displayInfo();

        System.out.println("Умеет говорить: " + canTalk + ", Любимое слово: " + favoriteWord);

    }

}

Создаем дочерний класс «Fish»:

class Fish extends Animal {

    private String waterType;

    private double size;

    public Fish() {

        super();

    }

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

        super(name, age, color);

        this.waterType = waterType;

        this.size = size;

    }

    @Override

    public void makeSound() {

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

    }

    @Override

    public void eat() {

        System.out.println(getName() + " ест корм для рыб");

    }

    public void swim() {

        System.out.println(getName() + " плавает в " + waterType + " воде");

    }

    public String getWaterType() {

        return waterType;

    }

    public void setWaterType(String waterType) {

        this.waterType = waterType;

    }

    public double getSize() {

        return size;

    }

    public void setSize(double size) {

        this.size = size;

    }

    @Override

    public void displayInfo() {

        super.displayInfo();

        System.out.println("Тип воды: " + waterType + ", Размер: " + size + " см");

    }

}

Вводим нужные параметры переменным и выводим необходимые результаты:

public class Animals {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        HomeCat cat1 = new HomeCat("Барсик", 2, "рыжий", "перс", "yes", "Маша");

        StreetCat cat2 = new StreetCat("Мурзик", 3, "серый", "дворовый", "no", "центр", true);

        Parrot parrot1 = new Parrot("Кеша", 1, "зеленый", true, "Привет");

        Fish fish1 = new Fish("Золотик", 1, "золотой", "пресная", 9);

        Animal[] animals = {cat1, cat2, parrot1, fish1};

        System.out.println("1) Полиморфизм динамический");

        for (Animal animal : animals) {

            animal.displayInfo();

            animal.makeSound();

            animal.eat();

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

        }

        System.out.println("2) Полиморфизм статический");

        cat1.makeSound();

        cat1.makeSound("Мяу мяу человек");

        System.out.println("-----");

        System.out.println("3) Наследование");

        cat1.play();

        cat1.sitOnSofa();

        cat2.walkAtNight();

        parrot1.fly();

        fish1.swim();

        System.out.println("-----");

        System.out.println("4) Инкапсуляция: Геттеры и сеттеры");

        System.out.println("Имя кота: " + cat1.getName());

        cat1.setAge(3);

        System.out.println("Новый возраст: " + cat1.getAge());

        System.out.println("-----");

        System.out.println("5) Счетчик животных");

        System.out.println("Всего животных: " + Animal.getCount());

        System.out.println("-----");

        System.out.println("6) Ввод нового кота:");

        System.out.print("Введите имя: ");

        String name = scanner.nextLine();

        System.out.print("Введите возраст: ");

        int age = Integer.parseInt(scanner.nextLine());

        System.out.print("Введите цвет: ");

        String color = scanner.nextLine();

        System.out.print("Введите породу: ");

        String breed = scanner.nextLine();

        System.out.print("Любит рыбу? (yes/no): ");

        String likesFish = scanner.nextLine();

        System.out.print("Домашняя или уличная? (home/street): ");

        String type = scanner.nextLine();

        if (type.equalsIgnoreCase("home")) {

            System.out.print("Введите хозяина: ");

            String owner = scanner.nextLine();

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

            HomeCat newCat = new HomeCat(name, age, color, breed, likesFish, owner);

            System.out.println("Создан новый кот:");

            newCat.displayInfo();

            newCat.makeSound();

        } else {

            System.out.print("Введите район: ");

            String area = scanner.nextLine();

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

            StreetCat newCat = new StreetCat(name, age, color, breed, likesFish, area, true);

            System.out.println("Создан новый кот:");

            newCat.displayInfo();

            newCat.makeSound();

        }

        scanner.close();

        System.out.println("-----");

        System.out.println("Итого животных: " + Animal.getCount());

    }

}

# Вывод

В ходе выполнения данной лабораторной работы были освоены навыки объектно-ориентировочного программирования на Java. Изучена работа классов и объектов.