**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 №2

По дисциплине «Информационные технологии и программирование»

Выполнил: Студент группы

БВТ2203

Попов Александр

Москва

2023

**Цель работы:** Изучение основ ООП языка программирования Java.

**Задание:**

1. Создайте иерархию классов в соответствии с вариантом (9). Ваша иерархия должна содержать: - абстрактный класс - 2 уровня наследуемых классов (классы должны содержать в себе минимум 3 поля и 2 метода, описывающих поведение объекта) - демонстрацию реализации всех принципов ООП (абстракция, модификаторы доступа, перегрузка, переопределение) - 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.

**Ход работы**

public *abstract* *class* Monstr {

*private* Integer height = null;

*private* Integer width = null;

*private* Integer weight = null;

*private* Integer countMonsters = 0;

*public* Monstr() {

        this.countMonsters++;

    }

*public* Monstr(int *height*, int *width*, int *weight*) {

        this.countMonsters++;

        this.height = *height*;

        this.width = *width*;

        this.weight = *weight*;

    }

*public* int getCountMonsters() {

*return* this.countMonsters;

    }

*public* int getHeight() {

*return* this.height;

    }

*public* void setHeight(int *height*) {

        this.height = *height*;

    }

*public* int getWidth() {

*return* this.width;

    }

*public* void setWidth(int *width*) {

        this.width = *width*;

    }

*public* int getWeight() {

*return* this.weight;

    }

*public* void setWeight(int *weight*) {

        this.weight = *weight*;

    }

*public* *abstract* void getSound();

Файл 1. Класс Monstr.

Сперва я создал абстрактный класс Monstr (Файл 1), реализовав все общие для монстров параметры, добавил геттеры, сеттеры, сделал 2 конструктора (один по умолчанию), добавил методы getCountMonsters и getSound, которые будут перегружены и переопределены соответственно дочерними классами.

public *class* Mermaid extends Monstr {

*private* Integer countMermaids = 0;

*private* boolean isSong;

    @Override

*public* void getSound() {

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

    }

*public* Mermaid() {

        super();

        this.countMermaids++;

        this.isSong=false;

    }

*public* Mermaid(int *height*, int *width*, int *weight*, boolean *isSong*) {

        super(*height*, *width*, *weight*);

        this.countMermaids++;

        this.isSong = *isSong*;

    }

*public* int getCountMonsters() {

*return* this.countMermaids;

    }

*public* void setSong() {

        this.isSong = !this.isSong;

    }

*public* boolean getSong() {

*return* this.isSong;

    }

}

Файл 2. Класс Mermaid.

Класс Mermaid (Файл 2), как и классы Goblin (Файл 3) и Dragon (Файл 4), имеют также по 2 конструктора, по 2 собственных поля, перегружают getCountMonsters и переопределяют getSound.

public *abstract* *class* Goblin extends Monstr {

*private* Integer countGoblins = 0;

*private* boolean isArmed;

    @Override

*public* void getSound() {

        System.out.println("Хлоп! Стоп! Вот тебе в лоб!" + "\n" +

        "Глядь! Хвать! Как тебя звать?"  + "\n" +

        "Живут гоблины тут, Эй, смелей, сюда!" + "\n" +

        "Стук! Звон! Топот и стон!" + "\n" +

        "Раз-два - молоток! Три-четыре - свисток! Вперед в подземный ход," + "\n" +

        "Эй, шевелись побыстрей! Треск! Свист! Кнут, хлыст! Бей и стучи! Рычи и мычи!" + "\n" +

        "Врешь, врешь! От нас не уйдешь! Гоблины пьют, хохочут, поют" + "\n" +

        "А ну-ка вперед, в подземный ход, Живей, живей, живей" + "\n");

    }

*public* Goblin() {

        super();

        this.countGoblins++;

        this.isArmed=false;

    }

*public* Goblin(int *height*, int *width*, int *weight*, boolean *isArmed*) {

        super(*height*, *width*, *weight*);

        this.countGoblins++;

        this.isArmed = *isArmed*;

    }

*public* int getCountMonsters() {

*return* this.countGoblins;

    }

*public* void setArmed() {

        this.isArmed = !this.isArmed;

    }

*public* boolean getArmed() {

*return* this.isArmed;

    }

}

Файл 3. Класс Goblin.

public *class* Dragon extends Monstr {

*private* Integer countDragons = 0;

*private* boolean isBreatheFlame;

    @Override

*public* void getSound() {

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

    }

*public* Dragon() {

        super();

        this.countDragons++;

        this.isBreatheFlame=false;

    }

*public* Dragon(int *height*, int *width*, int *weight*, boolean *isBreatheFlame*) {

        super(*height*, *width*, *weight*);

        this.countDragons++;

        this.isBreatheFlame = *isBreatheFlame*;

    }

*public* int getCountMonsters() {

*return* this.countDragons;

    }

*public* void setBreatheFlame() {

        this.isBreatheFlame = !this.isBreatheFlame;

    }

*public* boolean getBreatheFlame() {

*return* this.isBreatheFlame;

    }

}

Файл 4. Класс Dragon.

Мы создаем еще один уровень наследования, добавляя класс GreenGoblin (Файл 5), наследуемый от Goblin.



Файл 5. Класс GreenGoblin.

**Выводы**

1. Я научился создавать классы, поля, перегружать и переопределять методы, наследовать классы и создавать экземпляры классов