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»

(МТУСИ)

Кафедра «Математическая Кибернетика и Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине

«Информационные технологии и программирование»

на тему

“Классы и их наследование”

Выполнил:

студент группы БВТ2302

Миронов А. А.

Москва, 2024 г.

Задание 1:

Создайте иерархию классов в соответствии с вариантом. 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.

Мне нужно создать иерархию классов по варианту **номер 6:**

6. Базовый класс: Геометрическая фигура

Дочерние классы:

Шар, Параллелепипед, Цилиндр

Ход работы.

Для начала создадим абстрактный класс:

```
package geometric_figures;

public abstract class GeometryFigure {
    private String figureType;
    private double xCoord;
    private double yCoord;
    private double zCoord;
    protected String color;
    protected double volume;
    public double radius;
    public double height;

    public GeometryFigure(String ft, double x, double y, double z, String col){
        figureType = ft;
        xCoord = x;
        yCoord = y;
        zCoord = z;
        color = col;
    }

    public String getType(){
        return figureType;
    }

    public abstract void assignColor(String cl);
```

```
public String getCol(){
    return color;
}

public abstract void displayFigInfo();

public double getVol(){
    return volume;
}

public double getcX(){
    return xCoord;
}

public double getcY(){
    return yCoord;
}

public double getcZ(){
    return zCoord;
}

public void setcX(double val){
    xCoord = val;
}

public void setcY(double val){
    yCoord = val;
```

```
    public void setcX(double val){
        xCoord = val;
    }

    public void setcY(double val){
        yCoord = val;
    }

    public void setcZ(double val){
        zCoord = val;
    }

    public double[] getCenter(){
        double[] centerCoord = {xCoord, yCoord, zCoord};
        return centerCoord;
    }

    public void setCenter(double x, double y,double z){
        xCoord = x;
        yCoord = y;
        zCoord = z;
    }
}
```

В этом классе мы обозначим основные свойства и методы, которые будут наследоваться другими классами, которые будут описывать шар, параллелепипед и цилиндр. Свойства, присущие всем фигурам это цвет, тип фигуры, координаты её центра по осям x,y,z, объём фигуры (он не задаётся, а высчитывается исходя из заданных параметров для каждой фигуры по своему). Свойства же высоты и радиуса конечно так же наследуются для всех, но используются они лишь в классах некоторых фигур (чтобы эти свойства были доступны нам напрямую в классах, которые наследуют абстрактный, мы используем у них модификатор доступа - protected). Также создадим здесь конструктор с переменными, которые используются во всех классах, чтобы вызывать его в дочерних классах для их собственных конструкторов. Пропишем

геттеры и сеттеры для получения доступа к переменным, чьи модификаторы доступа не позволят обратиться из определённой среды к ним напрямую. Пропишем два абстрактных метода (методы, которые должны использоваться во всех дочерних классах).

Теперь перейдём к описанию дочерних классов, которые наследуют суперкласс GeometryFigure. Для начала опишем класс шара - Sphere:

```
package geometric_figures;
public class Sphere extends GeometryFigure {

    public static double counter = 0;

    public Sphere(double x, double y, double z, String clr, double rad){
        super(ft:"Sphere", x, y, z, clr);
        radius = rad;
        counter++;
        volume = (4.0/3.0)*Math.PI*Math.pow(radius, b:3);

    }

    public Sphere(){
       this(x:0, y:0, z:0, clr:"White", rad:1);
    }

    @Override
    public void assignColor(String cl){
        color = cl;
    }

    @Override
    public void displayFigInfo(){
        volume = (4.0/3.0)*Math.PI*Math.pow(radius, b:3);
        System.out.println("The type of figure: " + this.getType() + "; Coordinates of center spot: " + getcX()+" , "+getcY()+" , "+getcZ()
        System.out.println("The amount of created objects: " + Sphere.counter);
    }
}
```

Создадим конструктор класса, который использует как конструктор суперкласса, так и собственные значения. Он присваивает унаследованным полям класса переданные в него значения (они передаются при создании экземпляра класса). Также в конструкторе класса мы прибавляем 1 к статической переменной, таким образом мы ведём счётчик созданных объектов класса Sphere. Вдобавок при создании объекта мы высчитываем объём фигуры, беря значения из унаследованных полей класса. Далее мы используем перегрузку конструктора, задавая конструктор по умолчанию (в том случае, если создаётся объект, без переданных в него каких-либо данных), в конструкторе по умолчанию используем наш главный конструктор (с параметрами). Переопределяем два наших абстрактных метода, один — отвечающий за назначение цвета шара, другой — за отображение (вывод) всей информации о шаре и перерасчёт объёма (чтобы при расчётах учитывались все изменения, внесённые в поля объекта до использования метода displayInfo).

Также по аналогии напишем дочерний класс цилиндра, что тоже для расчёта объёма будет использовать наследованное у суперкласса поле радиуса, но (в отличие от шара) он будет использовать и поле высоты.

```java
package geometric_figures;

public class Cyllinder extends GeometryFigure{

    public Cyllinder(double x, double y, double z, String clr, double rad, double ht){
        super(ft:"Cyllinder", x, y, z, clr);
        radius = rad;
        height = ht;
        volume = Math.PI*Math.pow(radius, b:2)*height;

}
    public Cyllinder(){
        this(x:0, y:0, z:0, clr:"Black", rad:1, ht:2);
    }

    @Override
    public void assignColor(String cl){
        color = cl;
    }

    @Override
    public void displayFigInfo(){
        volume = Math.PI*Math.pow(radius, b:2)*height;
        System.out.println("The type of figure: " + getType() + "; Coordinates of center spot: " + getcX()+" , "+getcY()+" , "+getcZ() + ";

}
}
```

И наконец опишем класс параллелепипеда, который использует наследованное свойство высоты и имеет два собственных свойства: ширины и длины:

```java
package geometric_figures;
public class Parallelepiped extends GeometryFigure {
    public double length;
    public double width;

    public Parallelepiped(double x, double y, double z, String clr, double ht, double len, double wd){
        super(ft:"Parallelepiped", x, y, z, clr);
        height = ht;
        length = len;
        width = wd;
        volume = height*length*width;

}
    public Parallelepiped(){
        this(x:0, y:0, z:0, clr:"Red", ht:1, len:3, wd:2);
    }

    @Override
    public void assignColor(String cl){
        color = cl;
    }

    @Override
    public void displayFigInfo(){
        volume = height*length*width;
        System.out.println("The type of figure: " + getType() + "; Coordinates of center spot: " + getcX()+" , "+getcY()+" , "+getcZ() + ";

    }
}
```

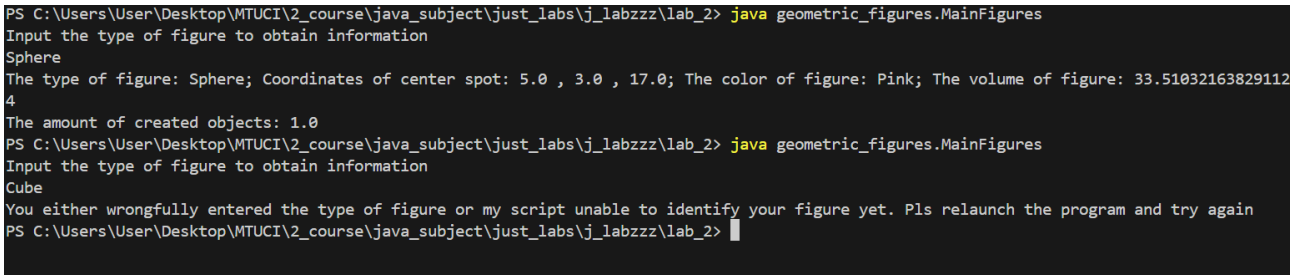
Теперь мы нужно создать файл с методом main, который и будет запускать нашу программу.

```java
package geometric_figures;
import java.util.Scanner;
public class MainFigures {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println(x:"Input the type of figure to obtain information");
        String figure = scanner.nextLine();

        if (figure.equals(anObject:"Parallelepiped")){
            Parallelepiped paral = new Parallelepiped();
            paral.assignColor(cl:"Yellow");
            paral.setCenter(x:2, y:5, z:3);
            paral.width = 4;
            paral.height = 3;
            paral.length = 6;
            paral.displayFigInfo();
        }
        else if (figure.equals(anObject:"Cyllinder")){
            Cyllinder cyl = new Cyllinder();
            cyl.assignColor(cl:"Blue");
            cyl.setCenter(x:1, y:8, z:3);
            cyl.height = 7;
            cyl.radius = 4;
            cyl.displayFigInfo();
        }
```

```java
        else if (figure.equals(anObject:"Cyllinder")){
            Cyllinder cyl = new Cyllinder();
            cyl.assignColor(cl:"Blue");
            cyl.setCenter(x:1, y:8, z:3);
            cyl.height = 7;
            cyl.radius = 4;
            cyl.displayFigInfo();
        }
        else if (figure.equals(anObject:"Sphere")){
            Sphere orb = new Sphere(x:1,y:2,z:3,clr:"Purple",rad:4);
            orb.assignColor(cl:"Pink");
            orb.setCenter(x:5, y:3, z:17);
            orb.radius = 2;
            orb.displayFigInfo();
            // System.out.println(orb.getcX()+" , "+orb.getcY()+" , "+orb.getcZ());
            // System.out.println(orb.getCol());
        }
        else {
            System.out.println(x:"You either wrongfully entered the type of figure or my script unable to identify your figure yet. Pls rel
        }
        scanner.close();
    }
}
```

В этом главным файле/классе мы используем класс Scanner (для этого его вначале нужно импортировать) и условия для того чтобы выводить на экран свойства фигуры, которую пользователь введёт с клавиатуры (для чтения и передачи в переменную строки, что ввёл пользователь используем метод scanner'a nextLine). Ну и естественно внутри условий создаём объекты классов наших фигур и применяем к этим объектам их различные методы, чтобы изменять значения их полей, включая метод displayInfo, который выводит информацию о текущих значениях полей соответствующей фигуры на экран.

Для удобной организации всех файлов я использую пакет geometric_figures. Вот результат выполнения моей программы в командной строке на примере ввода Sphere с клавиатуры, а затем на примере ввода с клавиатуры названия фигуры, которую моя программа не обрабатывает (тут не показана компиляция всех файлов, но она уже была произведена до этого):



Вывод: Мы разобрались с тем, как работает наследование в языке программирования джава на примере построения нашей собственной иерархии классов. Поняли смысл 4ёх основных принципов объектно-ориентированного программирования и продемонстрировали их реализацию в нашей программе.