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

Изучение принципов ООП

Выполнил: Студент группы

БВТ2303

Кунецкий Владислав

Москва

2024

Цели работы:

- познакомиться и изучить основные концепции ООП

- применить полученные знания на практике, написав свою иерархию классов

Ход работы:

В задании мне попалась следующая иерархия классов:

- Базовый класс: Геометрическая фигура,
- Дочерние классы: Шар, Параллелепипед, Цилиндр

Сначала я создал абстрактный класс, который включал общие методы и атрибуты, присущие любой фигуре (в 3д пространстве).

```java
package lab2.src;

public abstract class GeoBasicClass {
    private static int counter = 0;
    public static final double PI = 3.14;
    protected double volume;
    protected double surfaceArea;

    protected GeoBasicClass() {
        counter++;
    }

    public static int getCounter() {
        return counter;
    }

    public double getVolume() {
        return volume;
    }

    public double getSurfaceArea() {
        return surfaceArea;
    }

    protected abstract void recalculateGeometry();
}
```

Скрин 1 – Базовый класс

Я определил следующие методы и атрибуты:

- публичную статическую константу PI. Публичная, потому что в приватности нет необходимости, статическая для того, чтобы она существовала и могла использоваться без создания экзмепляров.

- Обьем и площадь поверхности. Они присуще любой 3д геометрической фигуре. Модификатор protected использован для того, чтобы доступ к этим атрибутам имели другие классы в нашем пакете, описывающие фигуры и при наследование они перенимали их. За пределами пакета доступ к ним запрещен.

- Счетчик определил по заданию. Приватный потому что мы не хотим, чтобы его можно было изменить при прямом обращении к классу и также не хотим, чтобы при наследовании создавался еще один счетчик. Статический потому что класс абстрактный, а поле существовать в памяти должно.

-Публичные геттер и сеттеры для объема и площади для получение и изменения их.

-Статический геттер для получения значения счетчика

-Конструктор, который при создании экземпляра класс, унаследовавшего наш абстрактный класс, увеличивает счетчик на 1

-абстрактный метод, который должен переопределить каждый класс, уноследовавший данный.

Следует также отменить, что я создал класс с модификатором abstract. Этот модификатор запрещает создание экземпляров и по сути делает класс базовым, для других классов. То, есть он нужен либо для наследования, либо для использования статический методов и переменных, определенных в нем.

```java
package lab2.src;

public class Sphere extends GeoBasicClass {
    private double radius;

    public Sphere() {
        this.radius = 0;
        this.volume = 0;
        this.surfaceArea = 0;
    }

    public Sphere(double radius) {
        if (radius < 0.0) {
            throw new IllegalArgumentException(s:"Радиус не может быть отрицательным");
        }
        this.radius = radius;
        recalculateGeometry();
    }

    @Override
    protected void recalculateGeometry() {
        this.volume = (4.0 / 3.0) * PI * Math.pow(radius, b:3);
        this.surfaceArea = 4 * PI * Math.pow(radius, b:2);
    }

    public void setRadius(double radius) {
        if (radius < 0.0) {
            throw new IllegalArgumentException(s:"Радиус не может быть отрицательным");
        }
        this.radius = radius;
        recalculateGeometry();
    }

    public double getRadius() {
        return radius;
    }
}
```

Скрин 2 – Класс, описывающий сферу (шар)

Далее начал создавать классы фигур. С помощью extend наследуем атрибуты и методы нашего базового класса.

В классе я определил атрибут, присущий только шару, и это радиус(приватный, потому что мы не хотим, чтобы его можно было изменить при прямом обращении). Определил 2 конструктора, используя перегрузку. Переопределил абстрактный метод для шарика. Написал сеттер и геттер. Также, с помощью ключевого слово throw определил класс ошибки.

```
package lab2.src;

public class Parallelepiped extends GeoBasicClass {
    private double height;
    private double side;
    private double width;

    public Parallelepiped() {
        this.height = 0;
        this.side = 0;
        this.width = 0;
        this.volume = 0;
        this.surfaceArea = 0;
    }

    public Parallelepiped(double height, double side, double width) {
        if (height < 0.0 || side < 0.0 || width < 0.0) {
            throw new IllegalArgumentException(s:"Одно из значений не может быть отрицательным");
        }
        this.height = height;
        this.side = side;
        this.width = width;
        recalculateGeometry();
    }

    public Parallelepiped(double height, double side){
        if (height < 0.0 || side < 0.0) {
            throw new IllegalArgumentException(s:"Одно из значений не может быть отрицательным");
        }
        this.height = height;
        this.side = side;
    }

    public Parallelepiped(double height){
        if (height < 0.0) {
            throw new IllegalArgumentException(s:"Высота не может быть отрицательной");    "Высота"
        }
        this.height = height;
    }

    public void setHeight(double height) {
        if (height < 0.0) {
            throw new IllegalArgumentException(s:"Высота не может быть отрицательной");    "Высота"
        }
        this.height = height;
        recalculateGeometry();
    }

    public void setSide(double side) {
        if (side < 0.0) {
            throw new IllegalArgumentException(s:"Сторона не может быть отрицательной");    "Сторон
        }
        this.side = side;
        recalculateGeometry();
    }

    public void setWidth(double width) {
        if (width < 0.0) {
            throw new IllegalArgumentException(s:"Ширина не может быть отрицательной");    "Ширина"
        }
        this.width = width;
        recalculateGeometry();
    }

    private double calculateVolume() {
        if (height == 0 || side == 0 || width == 0) {
            System.out.println(x:"Одно из значений не задано");    "Одно": Unknown word.
        }
        return height * side * width;
    }

    private double calculateSurfaceArea() {
        if (height == 0 || side == 0 || width == 0) {
            System.out.println(x:"Одно из значений не задано");    "Одно": Unknown word.
        }
        return 2 * (height * side + side * width + height * width);
    }

    @Override
    protected void recalculateGeometry() {
        if (height > 0 && side > 0 && width > 0) {
            this.volume = calculateVolume();
            this.surfaceArea = calculateSurfaceArea();
        }
    }

    public double getHeight() {
        return height;
    }

    public double getSide() {
        return side;
    }

    public double getWidth() {
        return width;
    }
}
```

Скрин 3 – Класс параллелепипеда.

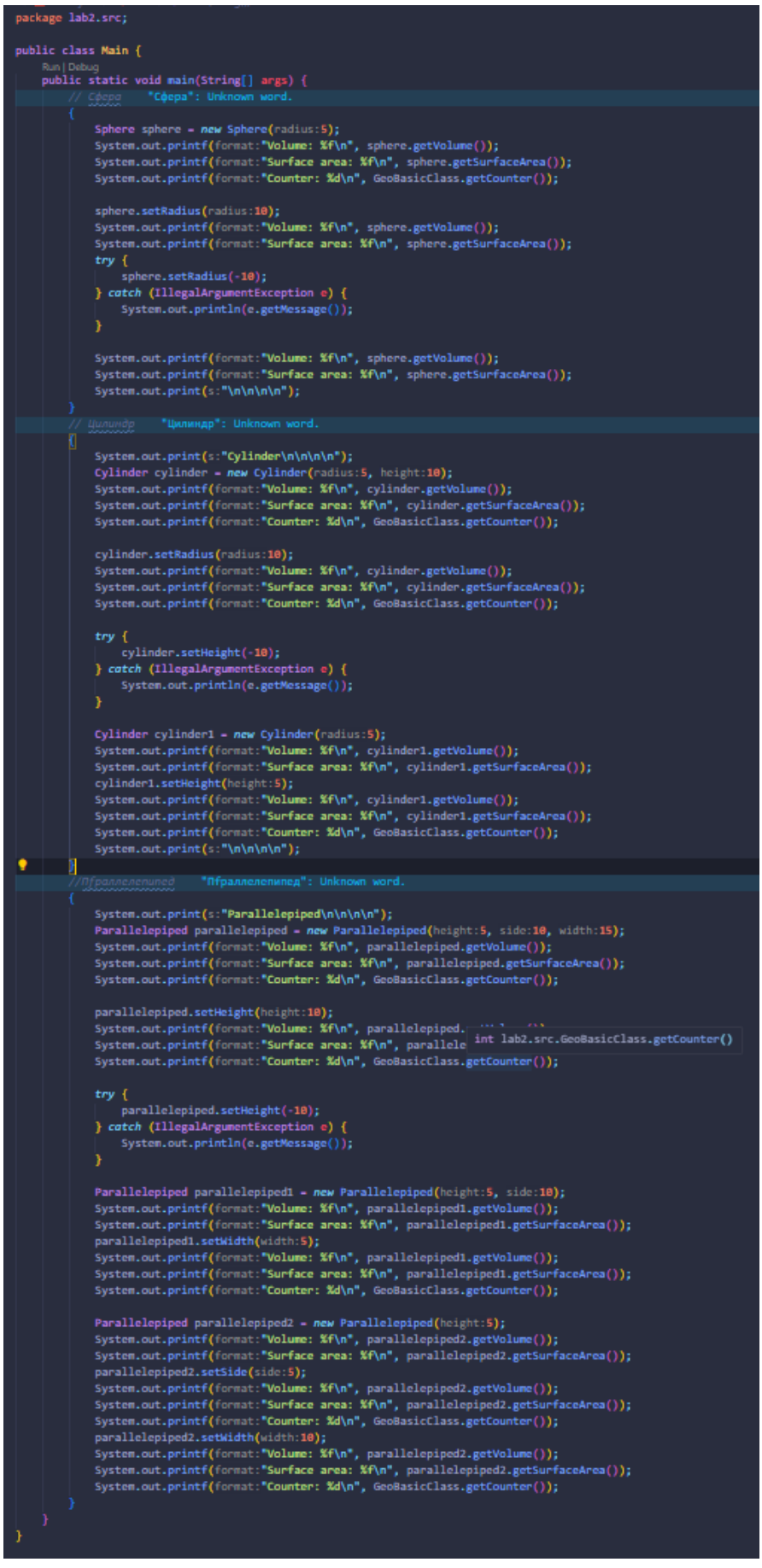
Скрин 4 – Цилиндр.

В классах, описывающих цилиндр и параллелепипед, проводим те же действия.

Иерархия классов готова. Теперь протестируем ее. Ниже, на 5 скрине, размещен код, для теста нашей иерархии. Я не смог проверить все возможные случае, но постарался предоставить как можно больше вариантов.

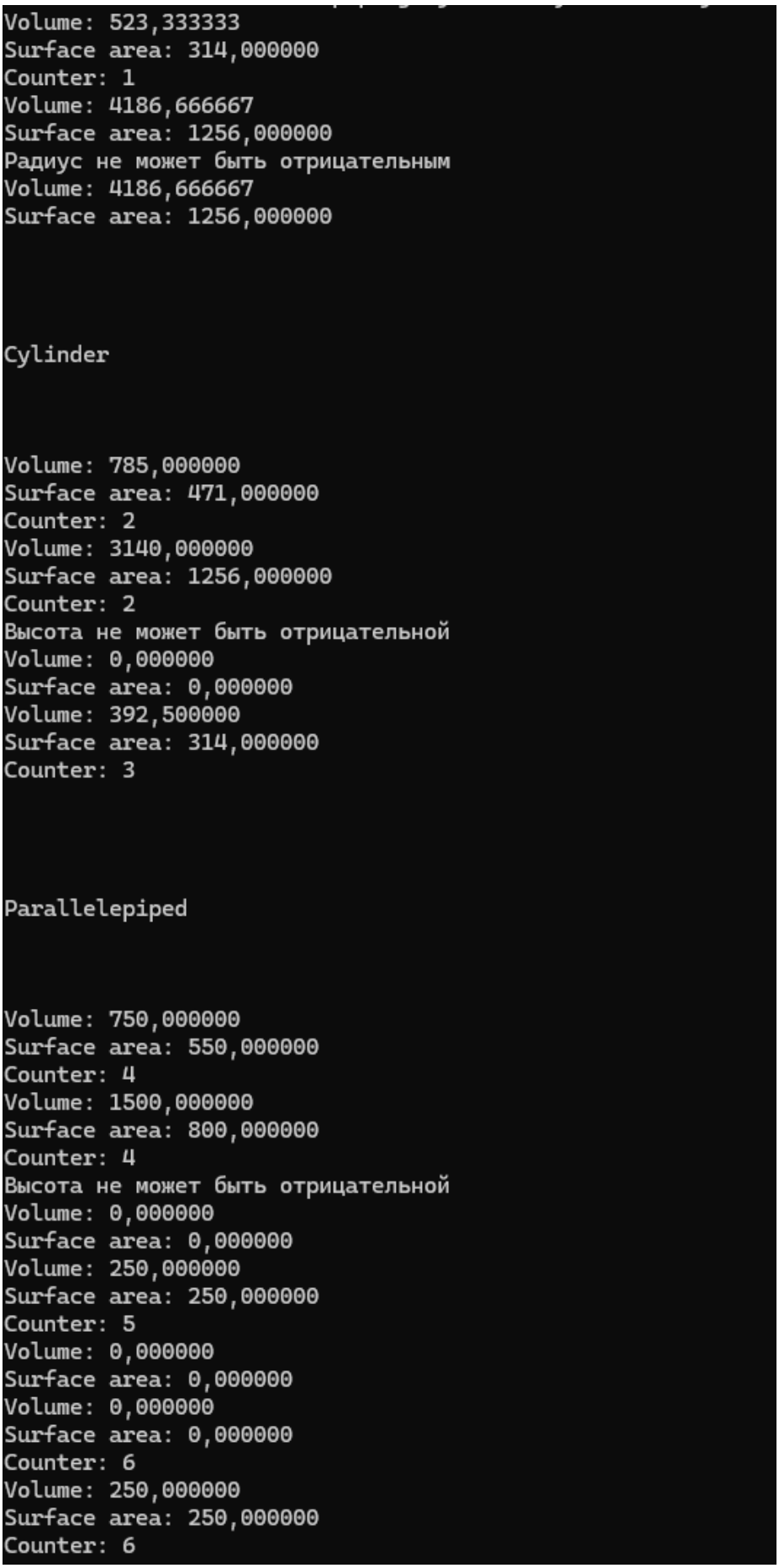
Так как в классах у нас определены исключения, мы можем ловить их с помощью конструкции try catch.

Результаты тестов можно посмотреть в скрине 6



```
package lab2.src;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        // Сфера    "Сфера": Unknown word.
        {
            Sphere sphere = new Sphere(radius:5);
            System.out.printf(format:"Volume: %f\n", sphere.getVolume());
            System.out.printf(format:"Surface area: %f\n", sphere.getSurfaceArea());
            System.out.printf(format:"Counter: %d\n", GeoBasicClass.getCounter());

            sphere.setRadius(radius:10);
            System.out.printf(format:"Volume: %f\n", sphere.getVolume());
            System.out.printf(format:"Surface area: %f\n", sphere.getSurfaceArea());
            try {
                sphere.setRadius(-10);
            } catch (IllegalArgumentException e) {
                System.out.println(e.getMessage());
            }

            System.out.printf(format:"Volume: %f\n", sphere.getVolume());
            System.out.printf(format:"Surface area: %f\n", sphere.getSurfaceArea());
            System.out.print(s:"\n\n\n\n");
        }
        // Цилиндр    "Цилиндр": Unknown word.
        {
            System.out.print(s:"Cylinder\n\n\n\n");
            Cylinder cylinder = new Cylinder(radius:5, height:10);
            System.out.printf(format:"Volume: %f\n", cylinder.getVolume());
            System.out.printf(format:"Surface area: %f\n", cylinder.getSurfaceArea());
            System.out.printf(format:"Counter: %d\n", GeoBasicClass.getCounter());

            cylinder.setRadius(radius:10);
            System.out.printf(format:"Volume: %f\n", cylinder.getVolume());
            System.out.printf(format:"Surface area: %f\n", cylinder.getSurfaceArea());
            System.out.printf(format:"Counter: %d\n", GeoBasicClass.getCounter());

            try {
                cylinder.setHeight(-10);
            } catch (IllegalArgumentException e) {
                System.out.println(e.getMessage());
            }

            Cylinder cylinder1 = new Cylinder(radius:5);
            System.out.printf(format:"Volume: %f\n", cylinder1.getVolume());
            System.out.printf(format:"Surface area: %f\n", cylinder1.getSurfaceArea());
            cylinder1.setHeight(height:5);
            System.out.printf(format:"Volume: %f\n", cylinder1.getVolume());
            System.out.printf(format:"Surface area: %f\n", cylinder1.getSurfaceArea());
            System.out.printf(format:"Counter: %d\n", GeoBasicClass.getCounter());
            System.out.print(s:"\n\n\n\n");
        }
        //Параллелепипед    "Параллелепипед": Unknown word.
        {
            System.out.print(s:"Parallelepiped\n\n\n\n");
            Parallelepiped parallelepiped = new Parallelepiped(height:5, side:10, width:15);
            System.out.printf(format:"Volume: %f\n", parallelepiped.getVolume());
            System.out.printf(format:"Surface area: %f\n", parallelepiped.getSurfaceArea());
            System.out.printf(format:"Counter: %d\n", GeoBasicClass.getCounter());

            parallelepiped.setHeight(height:10);
            System.out.printf(format:"Volume: %f\n", parallelepiped.
            System.out.printf(format:"Surface area: %f\n", parallele  int lab2.src.GeoBasicClass.getCounter()
            System.out.printf(format:"Counter: %d\n", GeoBasicClass.getCounter());

            try {
                parallelepiped.setHeight(-10);
            } catch (IllegalArgumentException e) {
                System.out.println(e.getMessage());
            }

            Parallelepiped parallelepiped1 = new Parallelepiped(height:5, side:10);
            System.out.printf(format:"Volume: %f\n", parallelepiped1.getVolume());
            System.out.printf(format:"Surface area: %f\n", parallelepiped1.getSurfaceArea());
            parallelepiped1.setWidth(width:5);
            System.out.printf(format:"Volume: %f\n", parallelepiped1.getVolume());
            System.out.printf(format:"Surface area: %f\n", parallelepiped1.getSurfaceArea());
            System.out.printf(format:"Counter: %d\n", GeoBasicClass.getCounter());

            Parallelepiped parallelepiped2 = new Parallelepiped(height:5);
            System.out.printf(format:"Volume: %f\n", parallelepiped2.getVolume());
            System.out.printf(format:"Surface area: %f\n", parallelepiped2.getSurfaceArea());
            parallelepiped2.setSide(side:5);
            System.out.printf(format:"Volume: %f\n", parallelepiped2.getVolume());
            System.out.printf(format:"Surface area: %f\n", parallelepiped2.getSurfaceArea());
            System.out.printf(format:"Counter: %d\n", GeoBasicClass.getCounter());
            parallelepiped2.setWidth(width:10);
            System.out.printf(format:"Volume: %f\n", parallelepiped2.getVolume());
            System.out.printf(format:"Surface area: %f\n", parallelepiped2.getSurfaceArea());
            System.out.printf(format:"Counter: %d\n", GeoBasicClass.getCounter());
        }
    }
}
```

Скрин 5 - Тест



Скрин 6 – Результаты тестов