**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: «Технологии разработки программного обеспечения»

на тему: «Развитие представлений о разработке программ. Объектно-ориентированное программирование в *Java*»

Выполнил: студент гр. ИТП-22

Зихич Д.В

Принял: преподаватель

Башаримов Ю.С

Гомель 2024

**Цель работы**: изучить основы разработки программ и объектно-ориентированное программирование в *Java*.

**Задание:**

1. Разработать *UML*-диаграмму иерархии классов, согласно варианта.

2. При наименовании компонентов руководствоваться соглашением о наименовании.

3. При описании иерархии использовать наследование и композицию.

4. На основе *UML*-диаграммы разработать иерархию классов на языке *Java*.

5. Весь код должен быть снабжен элементами документирования.

6. Разработанную иерархию поместить в .*jar* файл для дальнейшего использования в качестве библиотечных классов.

7. Создать консольное приложение для демонстрации работы созданных классов.

8. Составить отчет о проделанной работе

**Ход работы**

Для начала написания кода мы должны спроектировать *UML*-диаграмму для иерархии классов. *UML*-диаграмма представлена на рисунке 1.

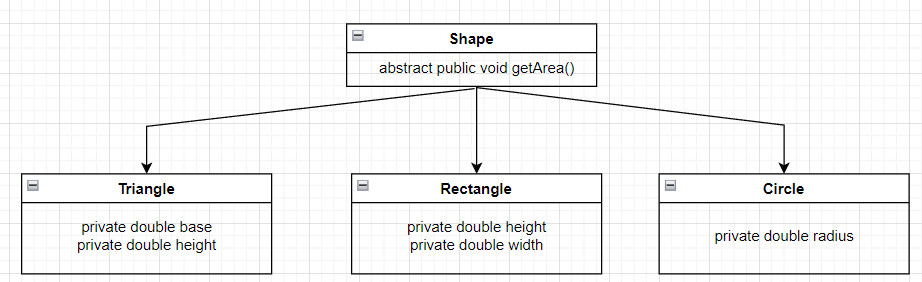


Рисунок 1 – *UML*-диаграмма иерархии классов

Для создания классов *Rectangle*, *Circe*, *Triangle* нужно воспользоваться наследованием. Листинг представлен в листинг «А». Результат выполнения кода представлен на рисунке 2.

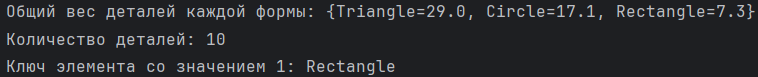


Рисунок 2 – Результат выполнения кода

На рисунке 3 показан результат сборки и запуска *jar* файла.

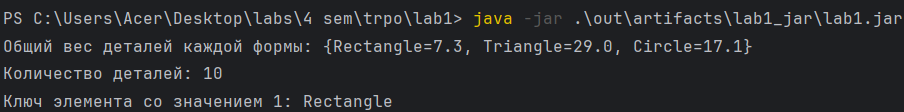


Рисунок 3 – Результат сборки и запуска *jar* файла

**Вывод:** в результате выполнения лабораторной были получены знания о основах разработки программ и объектно-ориентированном программировании в *Java*.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Текст программы**

**Main.java:**

import java.util.\*;

public class Main {

public static void main(String[] args) {

List<Detail> details = new ArrayList<>();

Circle circle = new Circle(4);

Triangle triangle = new Triangle(4.5, 1.2);

Rectangle rectangle = new Rectangle(5.1, 2.5);

details.add(new Detail(circle, "steel", 1.5));

details.add(new Detail(triangle, "wood", 0.7));

details.add(new Detail(circle, "plastic", 0.3));

details.add(new Detail(circle, "bronze", 2.0));

details.add(new Detail(circle, "plastic", 3.2));

details.add(new Detail(triangle, "bronze", 15.2));

details.add(new Detail(rectangle, "aluminium", 7.3));

details.add(new Detail(triangle, "uran", 11.1));

details.add(new Detail(circle, "gold", 10.1));

details.add(new Detail(triangle, "glass", 2.0));

Map<Shape, Double> weightByShape = new HashMap<>();

Map<Shape, Integer> countOfShapes = new HashMap<>();

for (Detail detail : details) {

weightByShape.put(detail.shape, weightByShape.getOrDefault(detail.shape, 0.0) + detail.weight);

countOfShapes.put(detail.shape, countOfShapes.getOrDefault(detail.shape, 0) + 1);

}

System.out.println("Общий вес деталей каждой формы: " + weightByShape);

System.out.println("Количество деталей: " + details.size());

for (Map.Entry<Shape, Integer> entry : countOfShapes.entrySet()) {

if (entry.getValue().equals(1)) {

System.out.println("Ключ элемента со значением 1: " + entry.getKey());

break;

}

}

}

}

**Shape.java:**

abstract class Shape {

public abstract double calculateArea();

}

**Circle.java:**

class Circle extends Shape {

private final double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

public double calculateArea() {

return Math.PI \* radius \* radius;

}

@Override

public String toString() {

return "Circle";

}

}

**Rectangle.java:**

class Rectangle extends Shape {

private final double width;

private final double height;

public Rectangle(double width, double height) {

this.width = width;

this.height = height;

}

@Override

public double calculateArea() {

return width \* height;

}

@Override

public String toString() {

return "Rectangle";

}

}

**Triangle.java:**

class Triangle extends Shape {

private final double base;

private final double height;

public Triangle(double base, double height) {

this.base = base;

this.height = height;

}

@Override

public double calculateArea() {

return 0.5 \* base \* height;

}

@Override

public String toString() {

return "Triangle";

}

}

**Detail.java:**

public class Detail {

Shape shape;

String material;

double weight;

Detail(Shape shape, String material, double weight) {

this.shape = shape;

this.material = material;

this.weight = weight;

}

}