

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ
ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ТЕЛЕКОММУНИКАЦИЙ
КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ**

Лабораторная работа по дисциплине «ООП»

Выполнил:

Хасензода Муборакшох Латиф

Студент 2 курса группы ПИН-б-о-22-1

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

Ставрополь, 2023 г.

Цель работы: изучить базовые понятия (классы, подклассы и методы)

Реализовать фундаментальные принципы объектно-ориентированного программирования.

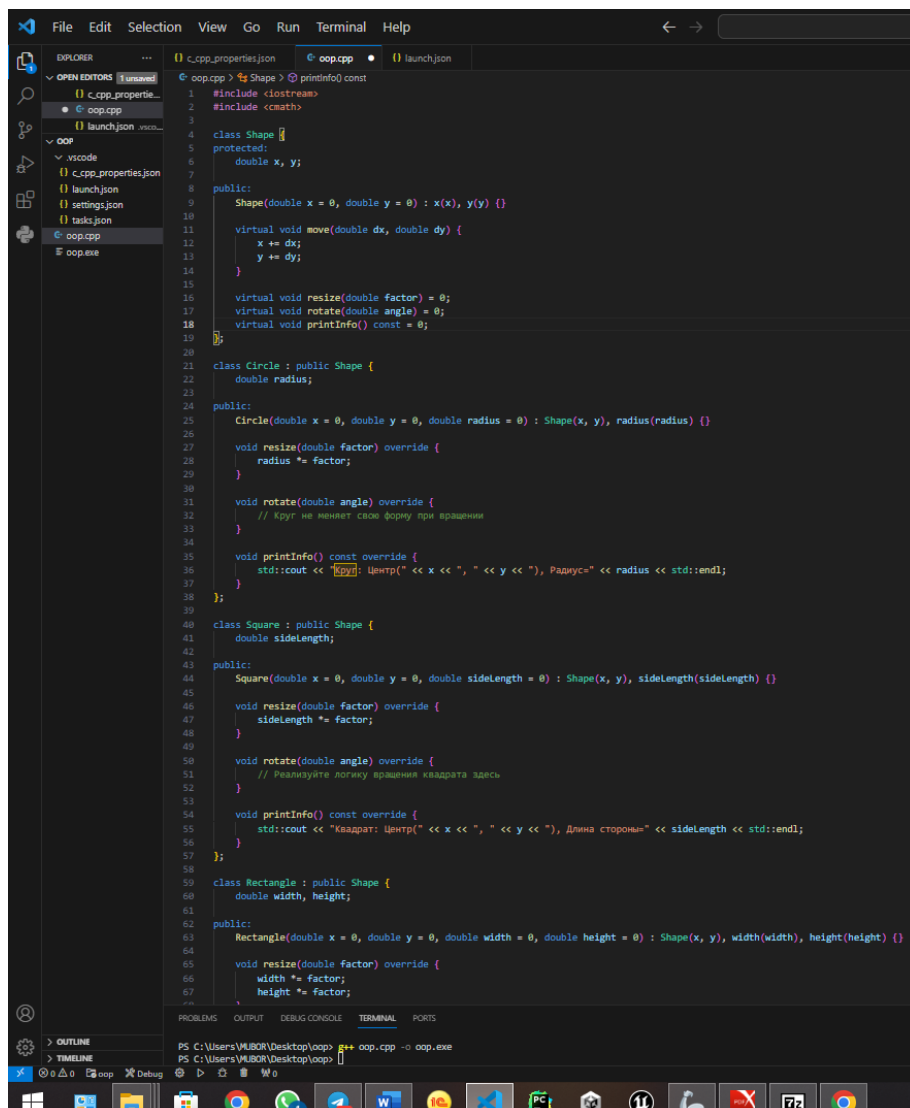
Вариант 3.

Построить систему классов для описания плоских геометрических фигур: круг, квадрат, прямоугольник. Предусмотреть методы для создания объектов, перемещения на плоскости, изменения размеров и вращения на заданный угол.

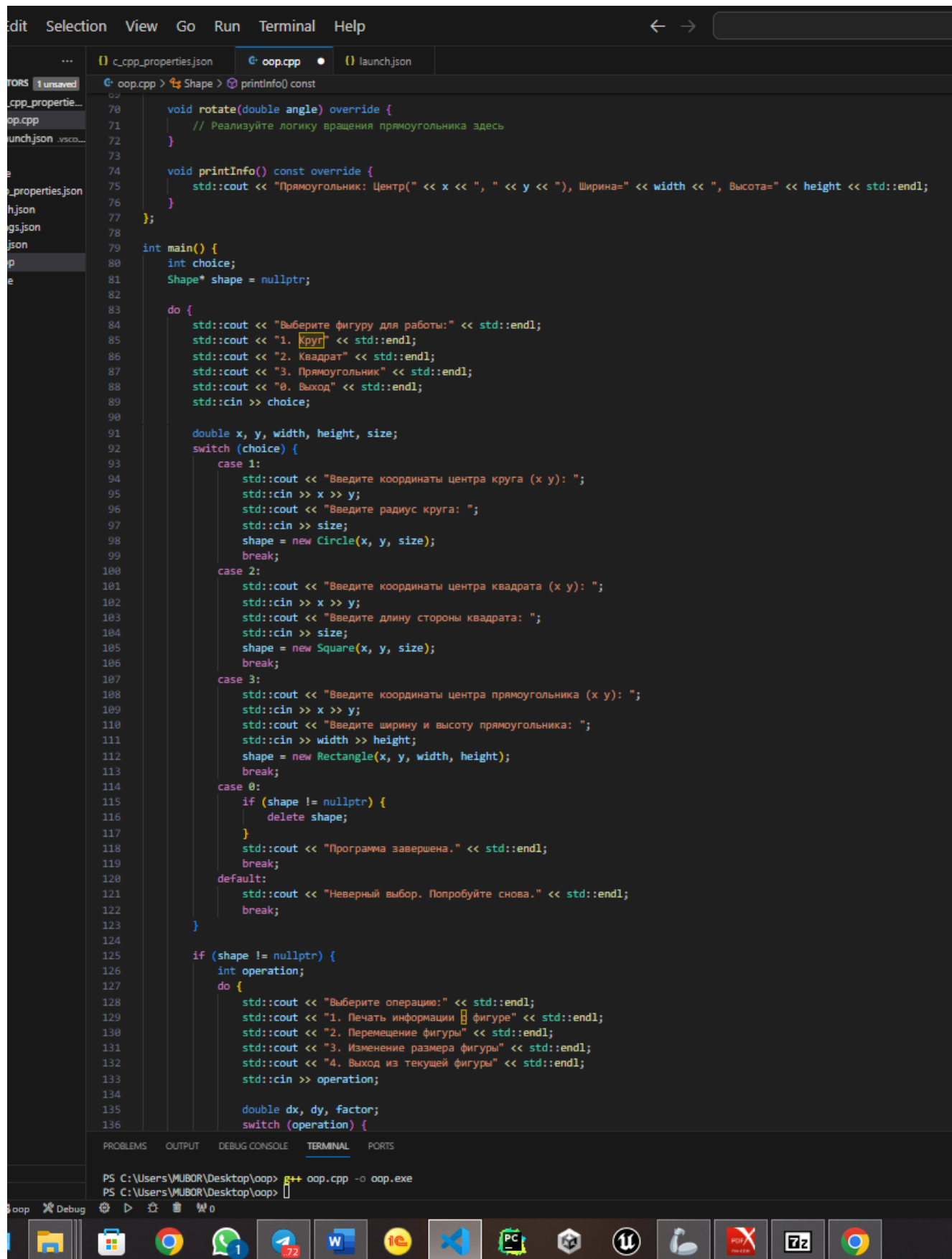
Написать программу, демонстрирующую работу с этими классами.

Программа должна содержать меню, позволяющее осуществить проверку всех

методов классов.

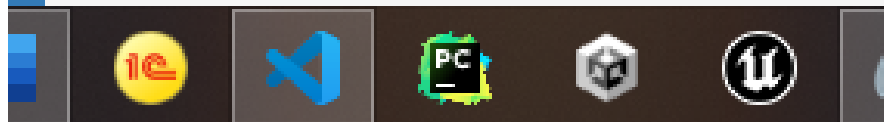


```
1 #include <iostream>
2 #include <cmath>
3
4 class Shape {
5 protected:
6     double x, y;
7
8 public:
9     Shape(double x = 0, double y = 0) : x(x), y(y) {}
10
11     virtual void move(double dx, double dy) {
12         x += dx;
13         y += dy;
14     }
15
16     virtual void resize(double factor) = 0;
17     virtual void rotate(double angle) = 0;
18     virtual void printInfo() const = 0;
19 };
20
21 class Circle : public Shape {
22     double radius;
23
24 public:
25     Circle(double x = 0, double y = 0, double radius = 0) : Shape(x, y), radius(radius) {}
26
27     void resize(double factor) override {
28         radius *= factor;
29     }
30
31     void rotate(double angle) override {
32         // Круг не меняет свою форму при вращении
33     }
34
35     void printInfo() const override {
36         std::cout << "Круг: Центр(" << x << ", " << y << "), Радиус=" << radius << std::endl;
37     }
38 };
39
40 class Square : public Shape {
41     double sideLength;
42
43 public:
44     Square(double x = 0, double y = 0, double sideLength = 0) : Shape(x, y), sideLength(sideLength) {}
45
46     void resize(double factor) override {
47         sideLength *= factor;
48     }
49
50     void rotate(double angle) override {
51         // Реализуйте логику вращение квадрата здесь
52     }
53
54     void printInfo() const override {
55         std::cout << "Квадрат: Центр(" << x << ", " << y << "), Длина стороны=" << sideLength << std::endl;
56     }
57 };
58
59 class Rectangle : public Shape {
60     double width, height;
61
62 public:
63     Rectangle(double x = 0, double y = 0, double width = 0, double height = 0) : Shape(x, y), width(width), height(height) {}
64
65     void resize(double factor) override {
66         width *= factor;
67         height *= factor;
68     }
69 }
```



Выберите фигуру для работы:

1. Круг
2. Квадрат
3. Прямоугольник
0. Выход



3. Прямоугольник

0. Выход

2

Введите координаты центра квадрата (x y): 5

6

Введите длину стороны квадрата: 3

Выберите операцию:

1. Печать информации о фигуре
2. Перемещение фигуры
3. Изменение размера фигуры
4. Выход из текущей фигуры



Код:

```
#include <iostream>

#include <cmath>

class Shape {
protected:
    double x, y;

public:
    Shape(double x = 0, double y = 0) : x(x), y(y) {}

    virtual void move(double dx, double dy) {
        x += dx;
        y += dy;
    }

    virtual void resize(double factor) = 0;
    virtual void rotate(double angle) = 0;
    virtual void printInfo() const = 0;
};

class Circle : public Shape {
    double radius;

public:
    Circle(double x = 0, double y = 0, double radius = 0) : Shape(x, y),
radius(radius) {}

    void resize(double factor) override {
        radius *= factor;
    }

    void rotate(double angle) override {
        // Круг не меняет свою форму при вращении
    }

    void printInfo() const override {
        std::cout << "Круг: Центр(" << x << ", " << y << "), Радиус=" << radius
<< std::endl;
    }
};

class Square : public Shape {
    double sideLength;

public:
    Square(double x = 0, double y = 0, double sideLength = 0) : Shape(x, y),
sideLength(sideLength) {}

    void resize(double factor) override {
```

```

        sideLength *= factor;
    }

    void rotate(double angle) override {
        // Реализуйте логику вращения квадрата здесь
    }

    void printInfo() const override {
        std::cout << "Квадрат: Центр(" << x << ", " << y << "), Длина стороны="
<< sideLength << std::endl;
    }
};

class Rectangle : public Shape {
    double width, height;

public:
    Rectangle(double x = 0, double y = 0, double width = 0, double height = 0) :
    Shape(x, y), width(width), height(height) {}

    void resize(double factor) override {
        width *= factor;
        height *= factor;
    }

    void rotate(double angle) override {
        // Реализуйте логику вращения прямоугольника здесь
    }

    void printInfo() const override {
        std::cout << "Прямоугольник: Центр(" << x << ", " << y << "), Ширина=" <<
width << ", Высота=" << height << std::endl;
    }
};

int main() {
    int choice;
    Shape* shape = nullptr;

    do {
        std::cout << "Выберите фигуру для работы:" << std::endl;
        std::cout << "1. Круг" << std::endl;
        std::cout << "2. Квадрат" << std::endl;
        std::cout << "3. Прямоугольник" << std::endl;
        std::cout << "0. Выход" << std::endl;
        std::cin >> choice;

        double x, y, width, height, size;
        switch (choice) {
            case 1:
                std::cout << "Введите координаты центра круга (x y): ";

```

```

        std::cin >> x >> y;
        std::cout << "Введите радиус круга: ";
        std::cin >> size;
        shape = new Circle(x, y, size);
        break;
    case 2:
        std::cout << "Введите координаты центра квадрата (x y): ";
        std::cin >> x >> y;
        std::cout << "Введите длину стороны квадрата: ";
        std::cin >> size;
        shape = new Square(x, y, size);
        break;
    case 3:
        std::cout << "Введите координаты центра прямоугольника (x y): ";
        std::cin >> x >> y;
        std::cout << "Введите ширину и высоту прямоугольника: ";
        std::cin >> width >> height;
        shape = new Rectangle(x, y, width, height);
        break;
    case 0:
        if (shape != nullptr) {
            delete shape;
        }
        std::cout << "Программа завершена." << std::endl;
        break;
    default:
        std::cout << "Неверный выбор. Попробуйте снова." << std::endl;
        break;
}

if (shape != nullptr) {
    int operation;
    do {
        std::cout << "Выберите операцию:" << std::endl;
        std::cout << "1. Печать информации о фигуре" << std::endl;
        std::cout << "2. Перемещение фигуры" << std::endl;
        std::cout << "3. Изменение размера фигуры" << std::endl;
        std::cout << "4. Выход из текущей фигуры" << std::endl;
        std::cin >> operation;

        double dx, dy, factor;
        switch (operation) {
            case 1:
                shape->printInfo();
                break;
            case 2:
                std::cout << "Введите смещение по x и y: ";
                std::cin >> dx >> dy;
                shape->move(dx, dy);
                break;
            case 3:

```

```

        std::cout << "Введите коэффициент масштабирования: ";
        std::cin >> factor;
        shape->resize(factor);
        break;
    case 4:
        delete shape;
        shape = nullptr;
        break;
    default:
        std::cout << "Неверный выбор. Попробуйте снова." <<
std::endl;
        break;
    }
    } while (operation != 4);
}
} while (choice != 0);

return 0;
}

```

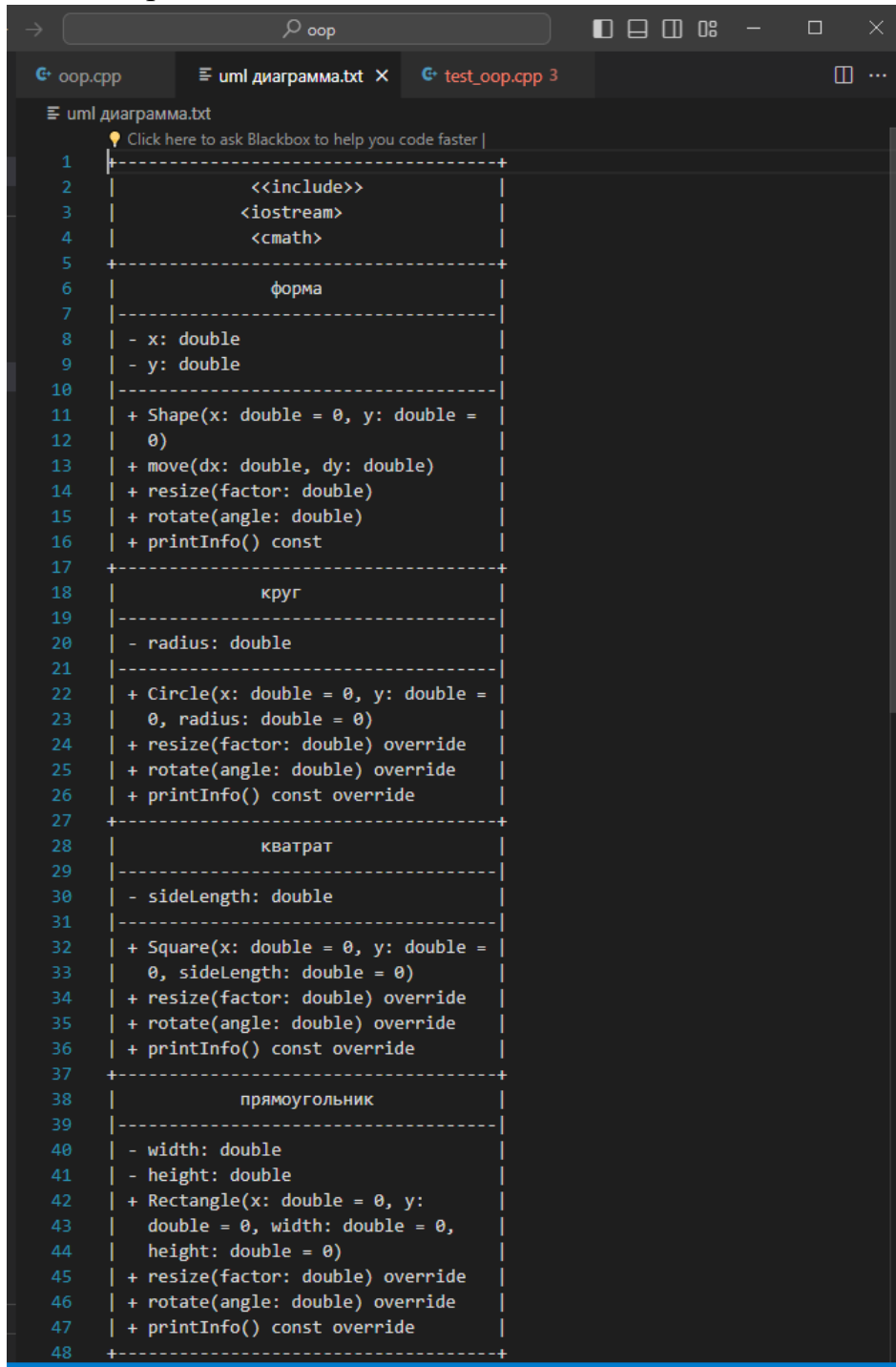
Тест:

```

3
Comment Code
4 TEST(ShapeTest, MoveTest) {
5     Shape* shape = new Circle(1.0, 2.0, 3.0);
6     shape->move(1.0, 1.5);
7     EXPECT_DOUBLE_EQ(2.0, shape->getX());
8     EXPECT_DOUBLE_EQ(3.5, shape->getY());
9     delete shape;
10 }
11
Comment Code
12 TEST(CircleTest, ResizeTest) {
13     Circle circle(0.0, 0.0, 4.0);
14     circle.resize(2.0);
15     EXPECT_DOUBLE_EQ(8.0, circle.getRadius());
16 }
17

```


Uml диаграмма:



```
1 |-----+
2 |         <<include>>
3 |         <iostream>
4 |         <cmath>
5 |-----+
6 |         форма
7 |-----+
8 |         - x: double
9 |         - y: double
10 |-----+
11 |         + Shape(x: double = 0, y: double =
12 |           0)
13 |         + move(dx: double, dy: double)
14 |         + resize(factor: double)
15 |         + rotate(angle: double)
16 |         + printInfo() const
17 |-----+
18 |         круг
19 |-----+
20 |         - radius: double
21 |-----+
22 |         + Circle(x: double = 0, y: double =
23 |           0, radius: double = 0)
24 |         + resize(factor: double) override
25 |         + rotate(angle: double) override
26 |         + printInfo() const override
27 |-----+
28 |         кватрат
29 |-----+
30 |         - sideLength: double
31 |-----+
32 |         + Square(x: double = 0, y: double =
33 |           0, sideLength: double = 0)
34 |         + resize(factor: double) override
35 |         + rotate(angle: double) override
36 |         + printInfo() const override
37 |-----+
38 |         прямоугольник
39 |-----+
40 |         - width: double
41 |         - height: double
42 |         + Rectangle(x: double = 0, y:
43 |           double = 0, width: double = 0,
44 |           height: double = 0)
45 |         + resize(factor: double) override
46 |         + rotate(angle: double) override
47 |         + printInfo() const override
48 |-----+
```