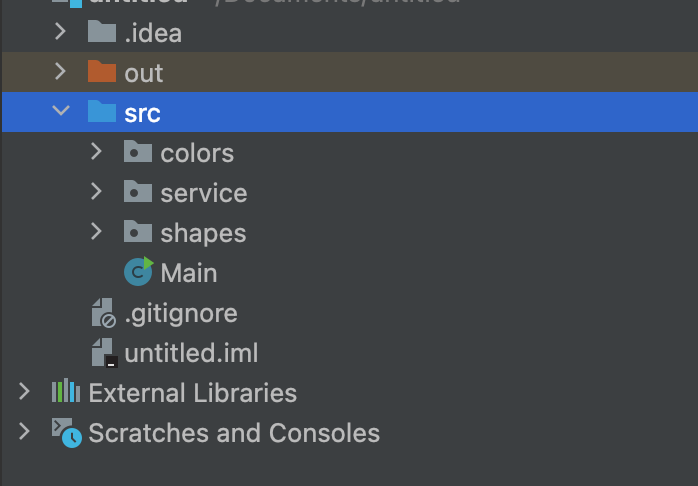
**Фигуры**

***Задание 1***

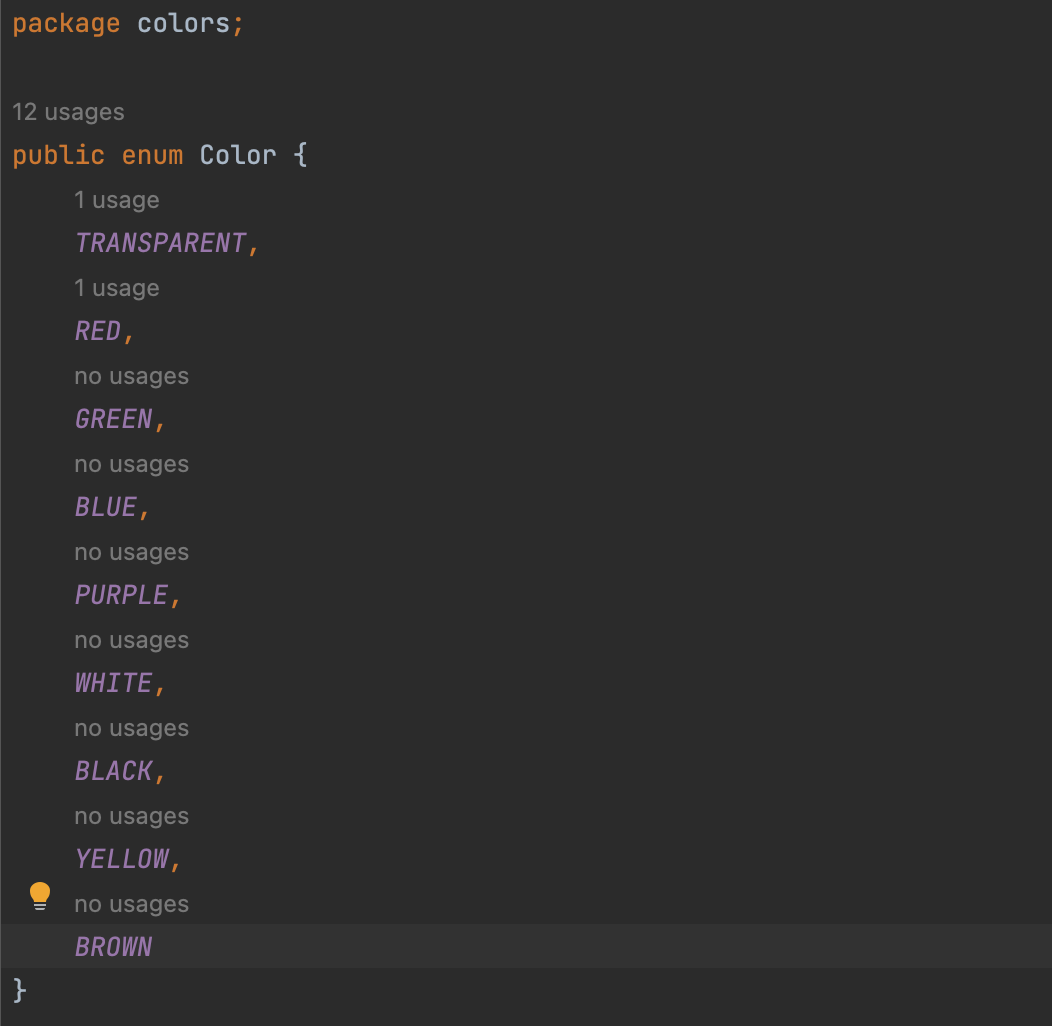
Создать новый проект и в папке **src** необходимо создать:

**shape** - папка, где будут храниться фигуры,

**color** - папка, где будут храниться цвета фигур



В папке **color** создать **enum** (*Тип enum — то специальный тип данных, который позволяет переменной быть набором предопределенных констант. Другими словами, он позволяет создать переменную, которая может принимать несколько значений*) вида:



В папке **shape** создать абстрактный класс **public abstract class Shape**

(*Кроме обычных классов в Java есть абстрактные классы. Абстрактный класс похож на обычный класс. В абстрактном классе также можно определить поля и методы, но в то же время нельзя создать объект или экземпляр абстрактного класса. Абстрактные классы призваны предоставлять базовый функционал для классов-наследников. А производные классы уже реализуют этот функционал.*).

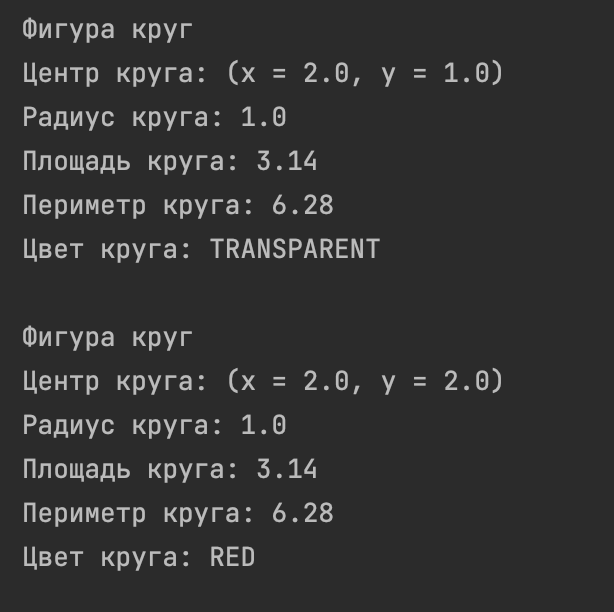
Данный класс должен содержать:

1. **private** поле **color** (дефолтный цвет - прозрачный)

**public Color color = Color.*TRANSPARENT***

1. **getter** и **setter** для поля **color**
2. **get**-методы для получения площади и периметра фигуры (дефолтное возвращаемое значение 0.0)
3. метод, который позволит переместить фигуру на плоскости **public public void move(double moveX, double moveY) {}**
4. **public void draw() {}** - метод который будет выводить название фигуры, ее цвет, координаты точек (для круга еще и радиус), площадь и периметр

Пример вывода для **круга** (остальные фигура аналогично):



После создания класса **Shape**, в папке **shape** необходимо создать следующие фигуры:

1. Точка (**Point**)
2. Прямоугольник (**Rectangle**),
3. Треугольник (**Triangle**)
4. Круг (**Circle**)

Подробное описание классов

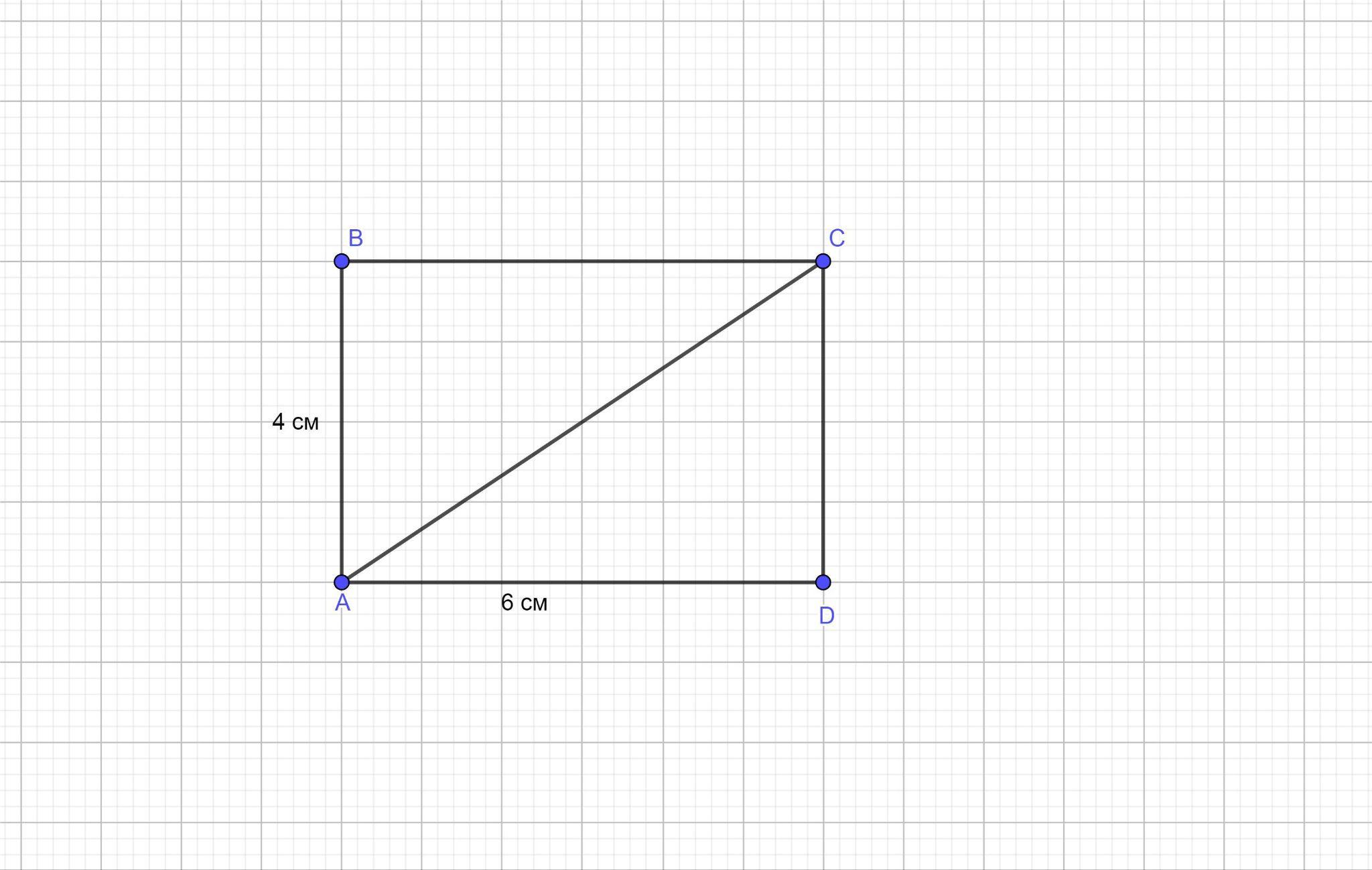
**Класс Точка**

1. класс должен наследоваться от класса **Shape**
2. должен содержать две **private double** переменные - **x** и **y**. Дефолтное значение **0.0**
3. **public getter** и **setter** для каждой переменной
4. конструктор без параметров, конструктор со всеми параметрами
5. метод **draw**, который выводит информацию о фигуре на экран
6. так как класс наследуется от класса **Shape** => имеет методы для вывода площади, периметра фигуры и перемещения фигуры на плоскости - эти методы необходимо переопределить и реализовать в соответствии указанной фигуры

**Класс Круг**

1. класс должен наследоваться от класса Shape
2. должен содержать две **private** переменные - **Point point** и **double radius**
3. **public getter** и **setter** для каждой переменной
4. **private** метод **validate()** - проверка, что радиус больше нуля, иначе кидать Exception() с сообщением, что не удалось создать фигуру, т.к. радиус не может быть меньше нуля
5. конструктор без параметров, конструктор со всеми параметрами
6. так как класс наследуется от класса **Shape** => имеет методы для вывода площади, периметра фигуры и перемещения фигуры на плоскости - эти методы необходимо переопределить в соответствии указанной фигуры

**Класс Прямоугольник**

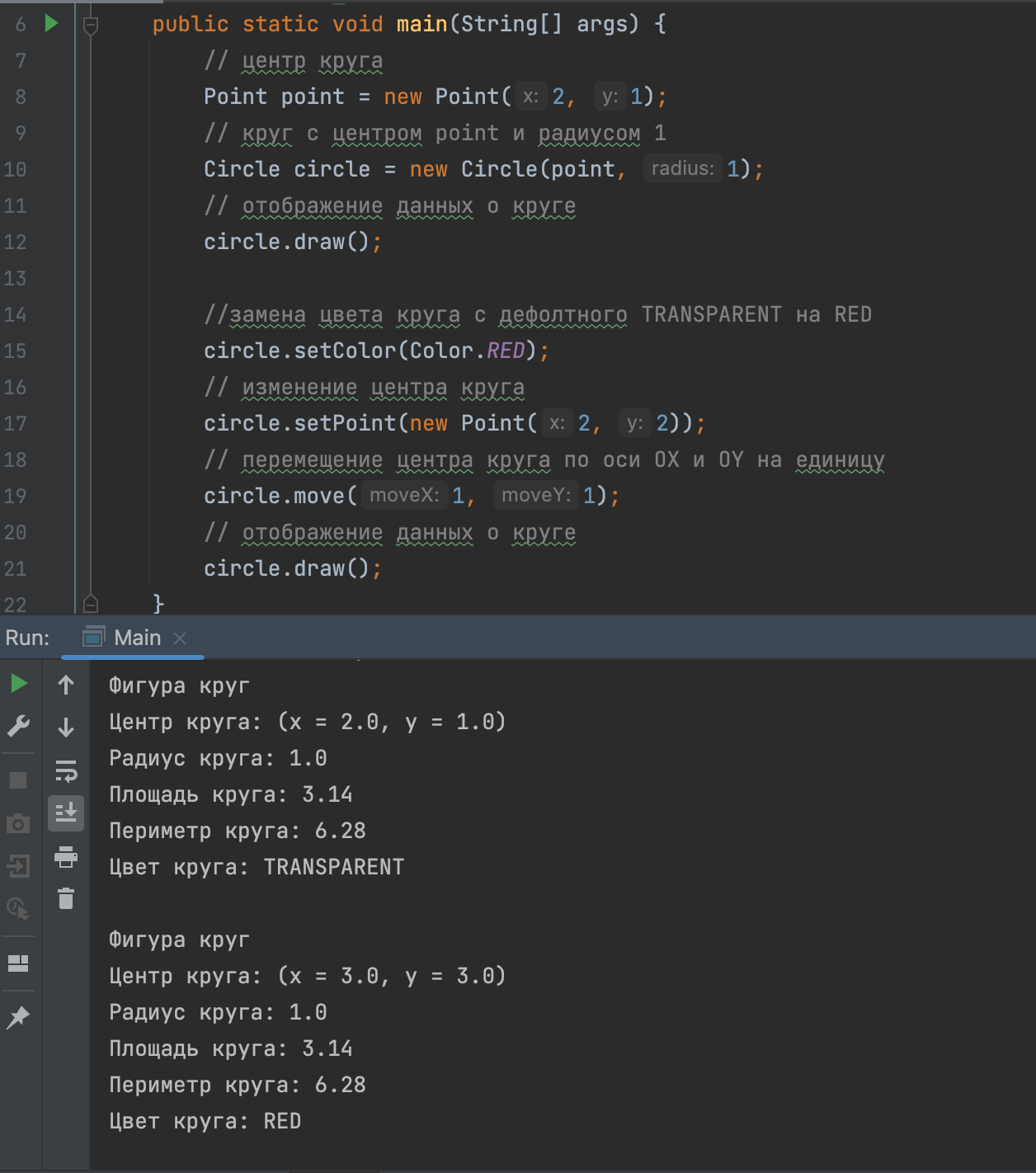
1. класс должен наследоваться от класса **Shape**
2. для реализации прямоугольника достаточно задать две точки, например **А и С** или **B и D** (см. рисунок), т.е класс должен содержать две **private** переменные **Point** 
3. **public getter** и **setter** для каждой переменной
4. **private** метод **validate()** - проверка, что точки x1 и x2 не лежат на одной прямой или y1 и y2 не лежат на одной прямой, иначе кидать Exception() с сообщением, что не удалось создать фигуру, т.к. точки х или y лежат на одной прямой
5. конструктор без параметров, конструктор со всеми параметрами
6. так как класс наследуется от класса **Shape** => имеет методы для вывода площади, периметра фигуры и перемещения фигуры на плоскости - эти методы необходимо переопределить в соответствии указанной фигуры

**Класс Треугольник**

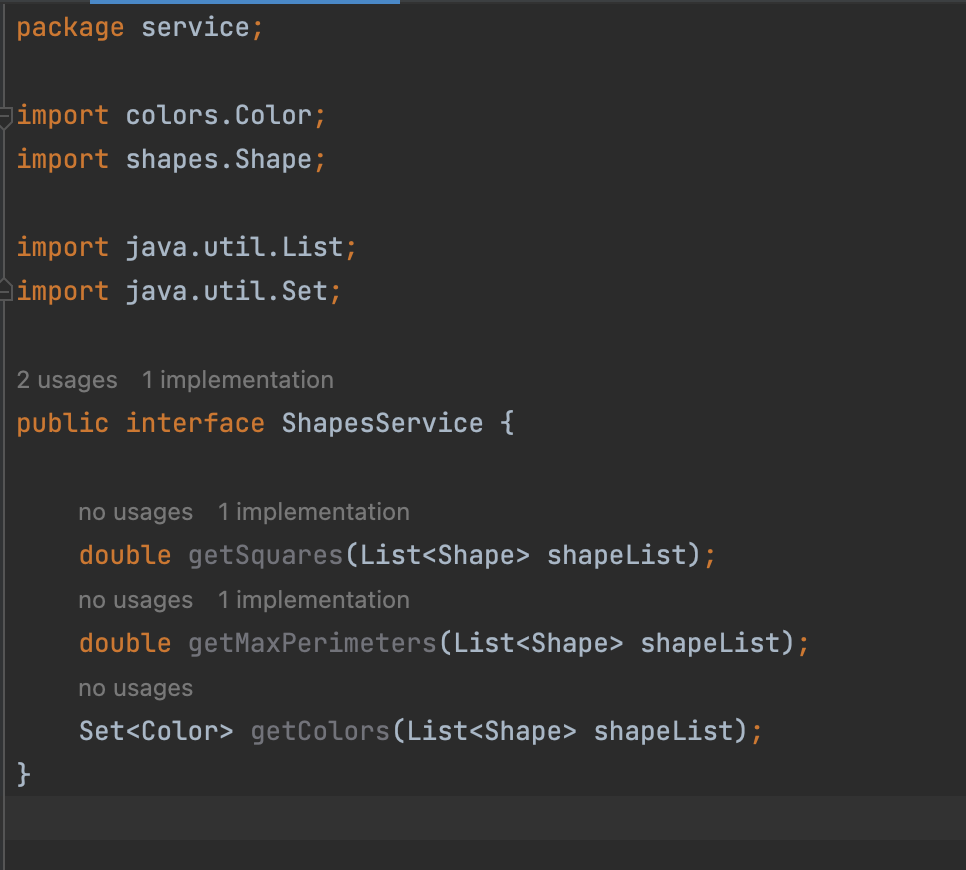
1. класс должен наследоваться от класса **Shape**
2. должен содержать три **private** переменные **Point**
3. **public getter** и **setter** для каждой переменной
4. метод валидации, что все три точки не лежат на одной прямой
5. конструктор без параметров, конструктор с параметрами
6. переопределить методы площади, периметра и перемещения

После того как все классы фигур созданы, проверим их работу. Проделать аналогичное с другими фигурами (лучше вынести данную логику для каждой фигуры в отдельные методы и вызывать их в методе **main(String[] args)**)

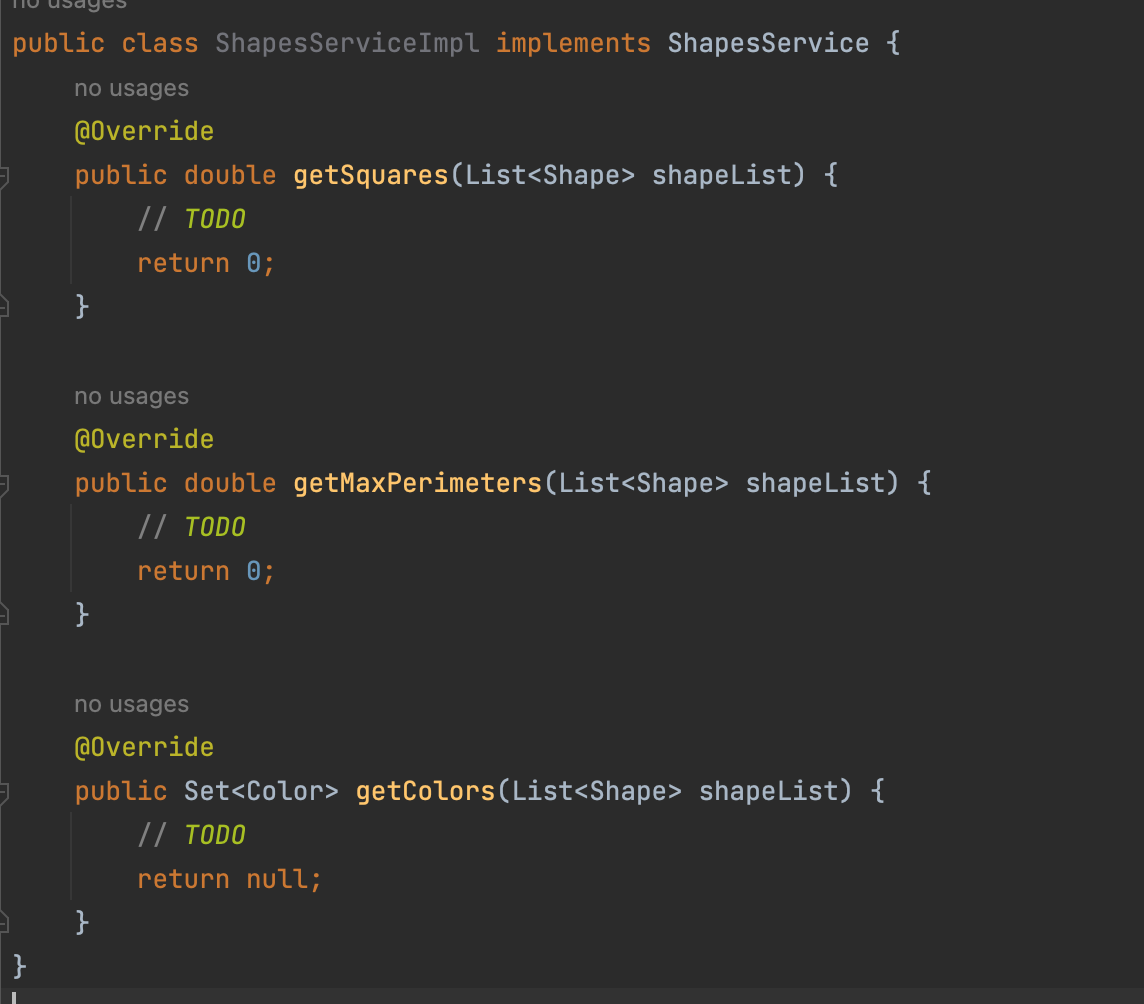
Пример с кругом:



***Задание 2***

В корне проекта создать папку **service**, добавить в эту папку **interface** **ShapesService** следующего вида:

После в папке **service** создать папку **impl**, в данной папке создать класс **ShapesServiceImpl**, который будет имплементировать (т.е. реализовывать) методы интерфейса **ShapesService**:



В данных методах необходимо написать следующую реализацию:

1. метод **public double getSquares(List<Shape> shapeList)**должен выводить сумму всех площадей фигур
2. метод **public double getMaxPerimeters(List<Shape> shapeList)**должен выводить максимальный периметр
3. метод **public Set<Color> getColors(List<Shape> shapeList)**должен выводить все цвета фигур без повторения

После реализации методов создать лист фигур (**List<Shape> shapes = new ArrayList();**) в методе **main(String[] args)**, который должен содержать

1. красный треугольник с координатами (0, 0), (0, 4), (4, 0)
2. круг белого цвета с радиусом 2 и центром в точке (3, 3)
3. круг желтого цвета с радиусом 4 и центром в точке (2, 5)
4. прямоугольник с координатами (1, 1) и (5, 2)
5. точка с координатой (100, 100)
6. синий треугольник с координатами (-2, 3), (4, 3), (2, 5)

Вызвать методы **ShapesService**, где входным параметром будет **List<Shape> shapes**. Вывести результаты на экран