```java
package refactoring;

public class Color {

    private String colorAsHex;
    private final String colorAsText;
    private String colorAsRGB_Red;
    private String colorAsRGB_Green;
    private String colorAsRGB_Blue;
    private String errorMessage;

    public Color(String colorAsText) {
        this.colorAsText = colorAsText;
        convertTextValueToRGBAndHex();
    }

    private void convertTextValueToRGBAndHex() {
        errorMessage = "";
        // set to Red
        if ("Red".equals(colorAsText)) {
            colorAsRGB_Red = "255";
            colorAsRGB_Blue = "0";
            colorAsRGB_Green = "0";
            colorAsHex = "#FF0000";
        } else if ("Blue".equals(colorAsText)) {
            // set to Blue
            colorAsRGB_Red = "0";
            colorAsRGB_Blue = "255";
            colorAsRGB_Green = "0";
            colorAsHex = "#00FF00";
        } else if ("Green".equals(colorAsText)) {
            // set to Green
            colorAsRGB_Red = "0";
            colorAsRGB_Blue = "0";
            colorAsRGB_Green = "255";
            colorAsHex = "#0000FF";
        } else {
            errorMessage = "Color not recognized";
        }
    }

    public String getBlue() {
        return colorAsRGB_Blue;
    }

    public String getGreen() {
        return colorAsRGB_Green;
    }

    public String getRed() {
        return colorAsRGB_Red;
    }

    public String getErrorMessage() {
        return errorMessage;
    }

    public String getColorFormatted(boolean includeHexAndRGB) {
        if (includeHexAndRGB) {
            return "%s %s %s:%s:%s".formatted(colorAsText, colorAsHex, colorAsRGB_Red,
                    colorAsRGB_Green, colorAsRGB_Blue);
        } else {
            return colorAsText;
        }
    }
}
```

```java
package refactoring;

public abstract class Shape {
    public abstract String format();
}

```

```java
package refactoring;


public class Circle extends Shape {
    private int x;
    private int y;
    private int r;
    private Color color = new Color("Green");
    private int numberOfContainedPoints;

    public Circle(int x, int y, int r) {
        if (r <= 0) {
            throw new RuntimeException("Radius needs to be larger 0");
        }
        this.x = x;
        this.y = y;
        this.r = r;
    }

    public int countContainedPoints(int[] xCords, int[] yCords) {
        this.numberOfContainedPoints = 0;
        if (xCords != null) {
            if (xCords.length > 0) {
                if (yCords != null) {
                    if (yCords.length > 0) {
                        if (xCords.length == yCords.length) {
                            for (int i = 0; i < xCords.length; ++i) {
                                contains(xCords, yCords, i);
                            }
                        } else {
                            throw new RuntimeException("Not every provided x coordinate " +
                                    "has a matching y coordinate");
                        }
                    } else {
                        throw new RuntimeException("y coordinates are empty");
                    }
                } else {
                    throw new RuntimeException("y coordinates are empty");
                }
            } else {
                throw new RuntimeException("x coordinates are empty");
            }
        } else {
            throw new RuntimeException("x coordinates are empty");
        }
        return numberOfContainedPoints;
    }

    public boolean contains(int[] xCords, int[] yCords, int i) {
        var result = (xCords[i] - this.x) * (xCords[i] - this.x)
                + (yCords[i] - this.y) * (yCords[i] - this.y) <= r * r;
        if (result) {
            this.numberOfContainedPoints++;
        }
        return result;
    }

    public void moveTo(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void resize(int r) {
        this.r = r;
    }

    @Override
    public String format() {
        return "circle: {" +
                "\n\tcenter: (" + this.x + "," + this.y + ") " +
                "\n\tradius: " + this.r +
                "\n\tcolor: " + this.color.getColorFormatted(false)
                + "\n}";
    }

}

```