```java
//Simon Girndt
//Hamdi Ghalil
public class Color {

    //Festlegen der h?ufig verwendeten Farben als nicht ver?nderliche Variablen
    static final Color BLACK = new Color (0, 0, 0);
    static final Color WHITE = new Color (255 ,255, 255);
    static final Color GRAY = new Color (128 ,128, 128);
    static final Color RED = new Color (255, 0, 0);
    static final Color GREEN = new Color(0, 255, 0);
    static final Color BLUE = new Color(0, 0, 255);

    private int rgb;
    //Konstruktor speichert int-Darstellung der Farbe in rgb
    public Color(int rgb) {
        this.rgb = rgb;
    }
    //Konstruktor pr?ft ob Werte f?r red green und blue zwischen 0 und 255 liegen
    public Color(int red, int green, int blue) {
        if (red > 255) {
            System.err.println("the value has to be in range 0, 255!");
            red = 255;
        }
        if (red < 0) {
            System.err.println("the value has to be in range 0, 255!");
            red = 0;
        }
        if (green > 255) {
            System.err.println("the value has to be in range 0, 255!");
            green = 255;
        }
        if (green < 0) {
            System.err.println("the value has to be in range 0, 255!");
            green = 0;
        }
        if (blue > 255) {
            System.err.println("the value has to be in range 0, 255!");
            blue = 255;
        }
        if (blue < 0) {
            System.err.println("the value has to be in range 0, 255!");
            blue = 0;
        }
        this.rgb = (red << 16) | (green << 8) | blue;


    }
    //Konstruktor f?r Farbe Schwarz
    public Color() {

        this.rgb = 0;
    }

    public int getRgb() {
    //?ffentliche getter Methoden um Werte f?r red green und blue einzeln auszulesen
        return rgb;
    }

    public int getRed(){
        int red = rgb >> 16 & 0xFF;        //setzt die unteren 8 bits auf 0
        return red;
    }
    public int getGreen(){
        int green = rgb >> 8 & 0b11111111;
        return green;
    }
    public int getBlue(){
        int blue = rgb & 0xFF;
        return blue;
    }
    //Methode zum zur?ckgeben der Hex-Darstellung einer Farbe
    public String getHex(){
        String rgbHex= "#" + Integer.toHexString(rgb);
        return rgbHex.toUpperCase();
    }

    public Color(String rgbHex){
        this.rgb = Integer.parseInt(rgbHex);
    }

    //Methode zur bildung der Komplement?rfarbe
    public Color complementaryColor(){
        int red = 255 - getRed();
        int green = 255 - getGreen();
        int blue = 255 - getBlue();
        return new Color (red, green, blue);
    }

    //Methode zum mischen zweier Farben
    public Color  mixColor(Color color){
        int rneu = (color.getRed() + getRed()) / 2;
        int gneu = (color.getGreen() + getGreen()) / 2;
        int bneu = (color.getBlue() + getBlue()) / 2;

        return new Color(rneu, gneu, bneu);
    }



    public static void main(String[] args) {
        //Initialisierung einiger Farben
        Color c = new Color(255, 16, 0);
        Color orange = new Color(255, 160, 0);
        Color aqua = new Color (0, 220, 255);
        Color floralwhite = new Color ( 240, 255, 255);
        Color firebrick = new Color (178, 34, 34);
        Color lightseagreen = new Color(32, 178, 170);
        System.out.println("RGB" + orange.getRgb());
        System.out.println("Red" + orange.getRed());
        System.out.println("Green" + orange.getGreen());
        System.out.println("Blue" + orange.getBlue());
        System.out.println("Hex" + orange.getHex());
        //Color Visualizer
        new ColorVisualizer(floralwhite);
        new ColorVisualizer(aqua);
        new ColorVisualizer(firebrick);
        new ColorVisualizer(lightseagreen);
        new ColorVisualizer(RED.mixColor(BLUE));
        new ColorVisualizer(GREEN.complementaryColor());
        int rgb2 = c.mixColor(new Color()).getRgb();

    }

}
```