```java
/*
 Carl Soergel, Matrikelnummer: 22367168
 Carlo Fuchs, Matrikelnummer: 21901371
 */

public class Color {

    private int rgb;
    // Statische, nicht veraenderliche Color-Objekte
    public static final Color BLACK = new Color(0, 0, 0);
    public static final Color WHITE = new Color(255, 255, 255);
    public static final Color GREY = new Color(128, 128, 128);
    public static final Color RED = new Color(255, 0, 0);
    public static final Color GREEN = new Color(0, 255, 0);
    public static final Color BLUE = new Color(0, 0, 255);

    // Ausgewaehlte Farben
    public static final Color ORANGE = new Color(255, 165, 0);
    public static final Color ROYALBLUE = new Color(65, 105, 225);
    public static final Color DARKRED = new Color(139, 0, 0);
    public static final Color FUCHSIA = new Color(255, 0, 255);

    public static void main(String[] args) {
        // Testen des Konstruktors fuer direkte Uebergabe der int-Darstellung
        Color black = new Color(0);
        System.out.println("Black RGB-Wert: " + black.getRgb());

        // Testen des Konstruktors fuer red, green und blue Werte
        Color red = new Color(255, 0, 0);
        System.out.println("Red RGB-Wert: " + red.getRgb());
        ColorVisualizer visualizerred = new ColorVisualizer(red);

        // Testen des Standard-Konstruktors fuer die Farbe Schwarz
        Color defaultColor = new Color();
        System.out.println("Default Color RGB-Wert: " + defaultColor.getRgb());
        ColorVisualizer visualizerdefault = new ColorVisualizer(defaultColor);

        // Testen der Methoden mit weiteren Farbwerte aus der Farbtabelle
        Color complementaryORANGE = ORANGE.complementaryColor();//Test Complement mit Visualisierung
        //System.out.println("Complementary ORANGE RGB-Wert: " + complementaryORANGE.getRgb());
        ColorVisualizer visualizerorange = new ColorVisualizer(ORANGE);
        ColorVisualizer visualizercomporange = new ColorVisualizer(complementaryORANGE);

```

```
44          Color mixedDARKREDandWHITE = DARKRED.mixColor(WHITE); //Test MIX mit Visualisierung
45          //System.out.println("RGB-Wert von Mischung aus DARKRED und WHITE: " + mixedDARKREDandWHITE.getRgb());
46          ColorVisualizer visualizerDARKRED = new ColorVisualizer(DARKRED);
47          ColorVisualizer visualizerWHITE = new ColorVisualizer(WHITE);
48          ColorVisualizer visualizermixedDARKREDandWHITE = new ColorVisualizer(mixedDARKREDandWHITE);
49
50          System.out.println("Fuchsia Hexadezimalwert: " + FUCHSIA.getHex()); //Test HEX
51          System.out.println("RGB-Werte fuer Royalblue: " + "Red: " + ROYALBLUE.getRed() + " Green: "
52                  + ROYALBLUE.getGreen() + " Blue: " + ROYALBLUE.getBlue()); //noch ziemlich dirty sollte Methode sein
53
54          Color crimson = new Color("#DC143C");
55          Color darkCyan = new Color("008B9B");
56          Color darkGoldenRod = new Color("B8860B");
57          Color darkRed = new Color("8B0000");
58
59          /*
60          // Testen der Fehlermeldung.
61          Color notAColor = new Color(-3, 257, 13);
62          // Ausgabe wie erwartet:
63          // Fehler: Der Farbwert darf nicht kleiner als 0 sein. Setze auf 0.
64          // Fehler: Der Farbwert darf nicht groesser als 255 sein. Setze auf 255.
65           */
66      }
67
68      // Konstruktor fuer die direkte Uebergabe der int-Darstellung
69      public Color(int rgb) {
70          this.rgb = rgb;
71      }
72
73      // Konstruktor fuer red, green und blue Werte
74      public Color(int red, int green, int blue) {
75          // ueberpruefen und korrigieren der Werte
76          this.rgb = ((validateAndCreateRGB(red) << 16) & 0xFF0000) |
77                  ((validateAndCreateRGB(green) << 8) & 0x00FF00) |
78                  (validateAndCreateRGB(blue) & 0x0000FF);
79      }
80
81      // Standard-Konstruktor Schwarz
82      public Color() {
83          this.rgb = 0;
84      }
85
86      // Ueberpruefung mit Fehlermeldung der Farbkanalwerte
```

```
87      private int validateAndCreateRGB(int value) {
88          if (value < 0) {
89              System.err.println("Fehler: Der Farbwert darf nicht kleiner als 0 sein. Setze auf 0.");
90              return 0;
91          } else if (value > 255) {
92              System.err.println("Fehler: Der Farbwert darf nicht groesser als 255 sein. Setze auf 255.");
93              return 255;
94          } else {
95              return value;
96          }
97      }
98
99      // Getter-Methode fuer rgb
100     public int getRgb() {
101         return rgb;
102     }
103
104     // Getter-Methode roter Farbkanal
105     public int getRed() {
106         return (rgb >> 16) & 0xFF;
107     }
108
109     // Getter-Methode gruener Farbkanal
110     public int getGreen() {
111         return (rgb >> 8) & 0xFF;
112     }
113
114     // Getter-Methode blauer Farbkanal
115     public int getBlue() {
116         return rgb & 0xFF;
117     }
118
119     // Getter-Methode fuer die 6-stellige Hexadezimaldarstellung
120     public String getHex() {
121         // Die Methode Integer.toHexString() wandelt den int-Wert in einen Hexadezimalstring um
122         String hexString = Integer.toHexString(rgb);
123
124         // Fuelle fuehrende Nullen auf, wenn noetig (um auf 6 Stellen zu kommen)
125         while (hexString.length() < 6) {
126             hexString = "0" + hexString;
127         }
128
129         // Fuege den #-Praefix hinzu und konvertiere den String zu Grossbuchstaben
```

```
130         return "#" + hexString.toUpperCase();
131     }
132
133     // Konstruktor fuer Uebergabe Hexadezimaldarstellung
134     public Color(String hex) {
135         // Entfernen des #-Praefix, falls vorhanden
136         hex = hex.replace("#", "");
137         // Parsen des Hexadezimalstring und speichern in rgb
138         rgb = Integer.parseInt(hex, 16);
139     }
140
141     public String toString() {
142         // Verwendung der getHex()-Methode, um die Hexadezimaldarstellung zu erhalten
143         return getHex();
144     }
145
146     public Color complementaryColor() {
147         int red = 255 - getRed();
148         int green = 255 - getGreen();
149         int blue = 255 - getBlue();
150         return new Color(red, green, blue);
151     }
152
153     public Color mixColor(Color color) {
154         int newRed = (getRed() + color.getRed()) / 2;
155         int newGreen = (getGreen() + color.getGreen()) / 2;
156         int newBlue = (getBlue() + color.getBlue()) / 2;
157         return new Color(newRed, newGreen, newBlue);
158     }
159
160 }
```