```java
package project1_jk;

/**
 * This class represents a color.
 * It uses two equivalent representations:
 * - RGB(red, green, blue) in which each component is represented using a decimal integer
 * in the range 0-255
 * - hexadecimal #RRGGBB in which the hexadecimal symbols and combined into a single string
 * preceded by a pound sign.
 *
 * Optionally, the Color object can store the CSS name of the color (or any other English language
 * name).
 *
 * @author Joanna Klukowska
 *
 */
public class Color implements Comparable<Color> {


    private String hexValue;
    private String colorName;

    /**
     * Constructs a new Color object with specified hex value.
     * @param hexValue hexadecimal value to be used for this Color; should be in the format #RRGGBB
     * @throws IllegalArgumentException  if hexValue parameter is invalid
     */
    public Color (String hexValue) throws IllegalArgumentException {
        this (hexValue, null );
    }

    /**
     * Constructs a new Color object with specified hex value and color name.
     * @param hexValue hexadecimal value to be used for this Color; should be in the format #RRGGBB
     * @param colorName color name to be used for this Color
     * @throws IllegalArgumentException if hexValue parameter is invalid
     */
    public Color ( String hexValue, String colorName )  throws IllegalArgumentException {
        setHexValue( hexValue );
        this.colorName = colorName;
    }

    /**
     * Constructs a new Color object with specified red, green and blue components.
     * @param red red component of this Color object; should be in the range of 0 to 255
     * @param green green component of this Color object; should be in the range of 0 to 255
     * @param blue blue component of this Color object; should be in the range of 0 to 255
     * @throws IllegalArgumentException  if red, gree, or blue parameters are invalid
     */
    public Color (int red, int green, int blue )  throws IllegalArgumentException {
        //validate ranges of the color components
        if (red < 0 || red > 255 )
            throw new IllegalArgumentException("Invalid value for red component. "
                    + "Valid range is 0-255.");
        if (green < 0 || green > 255 )
            throw new IllegalArgumentException("Invalid value for green component. "
                    + "Valid range is 0-255.");
        if (blue < 0 || blue > 255 )
            throw new IllegalArgumentException("Invalid value for blue component. "
                    + "Valid range is 0-255.");

        hexValue = String.format("#%02X%02X%02X" ,red, green, blue );
        colorName = null;

```

```
65      }
66
67      /**
68       * Returns the hexadecimal value representing this Color object.
69       * @return the hex value of this Color object
70       */
71      public String getHexValue () {
72          return hexValue;
73      }
74
75      /**
76       * Returns the English name of this Color object.
77       * @return the English name of this Color object
78       */
79      public String getName () {
80          return colorName;
81      }
82
83      /**
84       * Returns the red component of this Color object.
85       * @return the red component of this Color object
86       */
87      public int getRed( ) {
88          return getComponent(1,2);
89      }
90
91      /**
92       * Returns the green component of this Color object.
93       * @return the green compontent of this Color object
94       */
95      public int getGreen( ) {
96          return getComponent(3,4);
97      }
98
99
100     /**
101      * Returns the blue component of this Color object.
102      * @return the blue component of this Color object
103      */
104     public int getBlue( ) {
105         return getComponent(5,6);
106     }
107
108     /**
109      * Returns the string representation of this Color.
110      * @returns the string representation of this Color object
111      */
112     public String toString () {
113         if (colorName != null) {
114             return String.format("%s, (%3d,%3d,%3d), %s",
115                     hexValue.toUpperCase() , getRed(), getGreen(), getBlue() ,colorName);
116         }
117         else {
118             return String.format("%s, (%3d,%3d,%3d)",
119                     hexValue.toUpperCase() , getRed(), getGreen(), getBlue());
120         }
121     }
122
123
124     /* (non-Javadoc)
125      * @see java.lang.Object#equals(java.lang.Object)
126      */
127     @Override
128     public boolean equals(Object obj) {
```

```
129         if (this == obj)
130             return true;
131         if (obj == null)
132             return false;
133         if (!(obj instanceof Color))
134             return false;
135         Color other = (Color) obj;
136         if (hexValue == null) {
137             if (other.hexValue != null)
138                 return false;
139         } else if (!hexValue.equalsIgnoreCase(other.hexValue))
140             return false;
141         return true;
142     }
143 
144 
145     /* (non-Javadoc)
146      * @see java.lang.Comparable#compareTo(java.lang.Object)
147      */
148     @Override
149     public int compareTo(Color o) {
150         return this.hexValue.compareToIgnoreCase(o.hexValue);
151     }
152 
153 
154     /**
155      * Extracts the component of this Color object represented by the range of indexes
156      * indexStart-indexEng (inclusive).
157      * @param indexStart first index for the character representing a Color component
158      * @param indexEnd  last index for the character representing a Color component
159      * @return an integer representing the component specified by the two index values
160      */
161     private int getComponent (int indexStart, int indexEnd) {
162         String val = hexValue.substring(indexStart, indexEnd+1);
163         return Integer.parseInt(val, 16);
164     }
165 
166 
167     /**
168      * Validates and sets the hex value for this Color object.
169      * @param hexValue hexadecimal value to be examined and set.
170      * @throws IllegalArgumentException if the hexValue is invalid
171      */
172     private void setHexValue(String hexValue)  throws IllegalArgumentException {
173 
174         //validate lenght of the string
175         if (hexValue.trim().length() != 7 )
176             throw new IllegalArgumentException("Invalid length. String format should be #RRGGBB.");
177 
178         //check the first character
179         if ( hexValue.charAt(0) != '#')
180             throw new IllegalArgumentException("Invalid leading character. String format "
181                     + "should be #RRGGBB.");
182 
183         //check symbols at positions 1-7
184         String validSymbols = "0123456789ABCDEFabcdef";
185         for (int i = 1; i < 7; i++ ) {
186             if (! validSymbols.contains( hexValue.charAt(i)+"" )   )
187                 throw new IllegalArgumentException("Invalid symbol found. String format "
188                         + "should be #RRGGBB.\n"
189                         + "Valid symbols are: " + validSymbols + ".");
190         }
191 
192         this.hexValue = hexValue;
```

```
193     }
194
195
196 }
197
```