

Criterion C: Development

The program is a Java program that accepts Color data three ways.

- A user-inputted text field which converts ints into a Color object.
- A user-inputted file chooser which takes in an image and outputs a two-dimensional array of RGB values based median cut algorithm by Sven Woltmann. (<https://github.com/SvenWoltmann/color-thief-java>)
- A colorChooser GUI.

List of Techniques

- Selection sort
- Binary search
- Using Java Color class and its utilities
- Use of BufferedImage to load a file through fileChooser GUI.
- Error handling
- if/else, for loops, toString(), template classes,
- JColorChooser GUI + related functionality
- ArrayList of Palette objects.
- Array of objects
- OOP Features: Encapsulation, polymorphism, inheritance.
- Switch statements
- GUI tabs and popup menus.
- Use of flag values.
- JTable GUI + related functionality
- Two-dimensional and one-dimensional arrays

Structure of the Program

My program is a database program that stores an arrayList of Palette objects. The palette objects consist of its attributes and an array of Color objects. The main classes of my program is the Palette class, the GUI class, and the BinarySearch and SelectionSort classes, which sorts and searches the arrayList of Paletes.

The algorithm that extracts a palette from an image is located in a separate package. By breaking my program into classes and packages, I have taken an Object-Oriented approach. The use of template classes allow me to utilize the OOP features of polymorphism and encapsulation, as well as abstraction.

- **Polymorphism:** when an object can take more than one form. The Palette class contains an overloaded constructor, illustrated below.

```

public Palette(){ //overloaded constructor
}

public Palette(String paletteName, String paletteDescription, boolean
               forWork, int colorsQuantity, Color [] colors){

```

- **Encapsulation:** when the methods and attributes of a class is made private and can only be accessed through public get and set methods. The attributes of the Palette class is made private and made accessible through public get methods. Such as `getPaletteName()`
- **Abstraction:** Sven Woltmann's algorithm is quite complex. However, the use of separate class allows me to only use the features of the algorithm that are relevant to me, such as `getPalette(BufferedImage sourceImage, int colorCount)`

Data Structures Used

Array of Objects

Working with palettes means I have to instantiate and initialize several arrays of Color objects into certain GUI elements. In order to save time, I have created several methods which will allow me to my array of Colors to interact with my GUI elements.

- `displaySwatches(...)` creates an array out of 10 `JTextFields` and displays the colors array in the corresponding text field.
- A static array is appropriate because the size of palettes are fixed to 10 swatches.

ArrayLists

Palettes are stored in an `ArrayList` of palettes. The use of a dynamic structure allows me to save memory to accommodate for the dynamic size of the array.

Main Unique Algorithms

- **Color Thief**
- This program utilizes Sven Woltmann's ColorThief algorithm, which grabs a representative color palette from a given image. It uses the Median Cut sorting algorithm, which sorts data into sets by recursively cutting the data set at a median point. This algorithm is most often used for color quantization, which is the process of reducing the number of colors in an image while retaining the visual qualities of the image.
- **Overview of Median Cut Algorithm**

- For any form of color quantization, a 3d clustering algorithm is ideal because we are dealing with three color channels: Red, Green, and Blue.
 - The process involves putting the RGB values of all pixels into a bucket
 - Find out which RGB color channel has the greatest range.
 - Sort the values inside the bucket.
 - Move the upper half of the bucket into a new bucket and repeat the same process until the desired number of buckets is reached (base case, must be power of two)
- **Buffered Image**
- This program utilizes the `BufferedImage` subclass which extends the `Image` superclass. This allows the user to create a palette from any uploaded image. A the `BufferedImage` was used because it allows the application to access and operate directly on the image. Compared to the image superclass, it is more efficient and useful for the following reasons
 - It generates a `ColorModel` and `Raster` for the image.
 - It stores and manages the image in memory.
 - Allows specific pixels in the image to be accessed.

Pseudocode for buffered image

```
Show fileChooser dialog;
File file = selected file from file chooser
Try{
Image img = read file;
Image resizedImage = scale the image(width of JFrame, height of
JFrame, Smooth scaling);
ImageIcon icon = resizedImage //make displayed image an
ImageIcon;
Set icon of JLabel as icon;
}
Catch(IOException //general error handling){
    Show popup menu "Please select an image"
}
```

- **Java Color Class**
- This program heavily utilizes the `java.awt.color` class. Using this class allows me to use its associated methods, and also saves time because I don't have create a new way to represent color in my IDE.
- The color class has multiple constructors (polymorphism)

- Color(r, g, b)

Pseudocode for Color class example

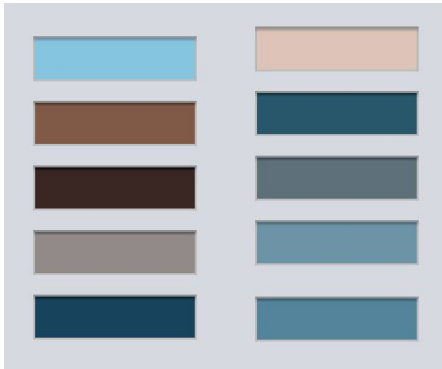
Declare an array of 10 Colors called colorsArray

Loop through a = 0, a < 10

```
Color c = new Color(paletteArray at [a][0] //red component,  
paletteArray at [a][1] //green component, paletteArray at [a][2]  
//blue component)
```

colorsArray at [a] = c

- Color.getRGB
- Color.WHITE
- color.getGreen, color.getRed, color.getBlue
- color.toString()

Sample Input	Sample output	Notes
	<p>RGB Output</p> <pre>Color1: {134 , 197 , 224} Color2: {222 , 195 , 184} Color3: {128 , 90 , 71} Color4: {40 , 86 , 106} Color5: {58 , 39 , 35} Color6: {94 , 113 , 122} Color7: {145 , 138 , 136} Color8: {108 , 148 , 166} Color9: {23 , 68 , 92} Color10: {84 , 132 , 156}</pre>	<p>Using color.getGreen, color.getRed, color.getBlue and color.toString() to display RGB color components as a String</p>

- Display Palette

- This algorithm displays a palette from a table selection using the JTextField GUI Element. Using an ArrayList allows me to easily find and display these palettes.

Pseudocode

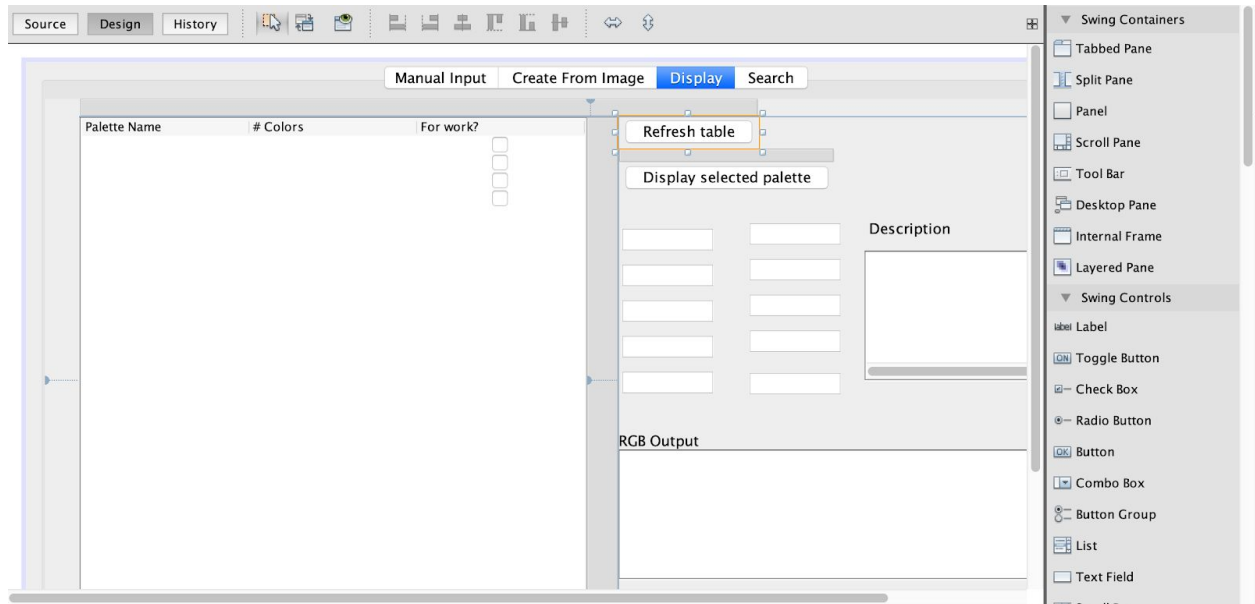
```
Int index = selectedRow of JTable;
```

```
Array of Colors displaySelected = get Palette object from  
paletteList @ index;
```

```
Display swatches (utilizing displaySwatches method. See  
displaySwatches() method);
```

Software Tools used

This program used the NetBeans IDE. Netbeans is ideal because it allows me to work in an Objected oriented way through the creation of classes and packages. It also has a Java library containing useful tools, such as java.awt.color. It also lets me add and program GUI elements. The GUI tab is very simple and easy to use, with several useful GUI features.



Word Count: 824

