Quote Painter

By Zi Jian

1. Project Overview

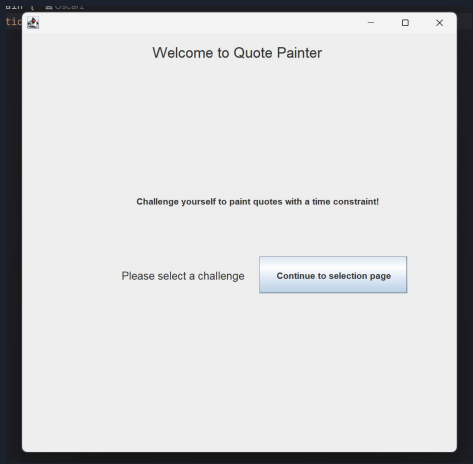
My program functions as a basic drawing application which also provides a challenge in the format of a timed challenge to draw a random prompt (pulled from zen quotes).

A problem that this program solves is providing fun prompt to draw when you're bored and have no idea what to draw.

The LaunchPage Class

```java
public class LaunchPage extends JFrame implements ActionListener {  4 usages  Oscarz
    JFrame frame = new JFrame();  10 usages
    int pageHorizontal = 600;  5 usages
    int pageVertical = 600;  4 usages
    JLabel title = new JLabel( text: "Welcome to Quote Painter");  3 usages
    JLabel options = new JLabel( text: "Please select a challenge");  3 usages
    JButton button = new JButton( text: "Continue to selection page");  5 usages

    LaunchPage(){  2 usages  Oscarz
        // Title
        title.setFont(new Font( name: "Arial", Font.PLAIN,  size: 20));
        title.setBounds( x: pageHorizontal / 2 - (500 / 4),  y: 0,  width: 500 ,  height: 50);

        // Explanation
        JLabel explanation = new JLabel( text: "Challenge yourself to paint quotes with a time constraint!");
        explanation.setBounds( x: pageHorizontal / 2 - (500 / 3) + 20,  y: (pageVertical / 2) - 100,  width: 500 ,  height: 50);
        frame.add(explanation);

        //Option
        options.setFont(new Font( name: "Helvetica", Font.PLAIN,  size: 15));
        options.setBounds( x: pageHorizontal / 2 - (500 / 3), (pageVertical / 2),  width: 500 ,  height: 50);

        //Button
        button.setBounds( x: pageHorizontal / 2 + 20, (pageVertical / 2),  width: 200,  height: 50);
        button.setFocusable(false);
        button.addActionListener( l: this);

        //Frame
        frame.setSize(pageHorizontal , pageVertical);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);
        frame.add(title);
        frame.add(options);
        frame.add(button);
        frame.setVisible(true);
    }

    @Override  Oscarz
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == button){
            frame.dispose();
            SelectionPage page = new SelectionPage();
        }
    }
}
```

The LaunchPage class isn't very exciting. It a JFrame that contains mainly just JLabels as text to introduce the user to the application. It contains an button which leads to another JFrame or window: The SelectionPage Object.

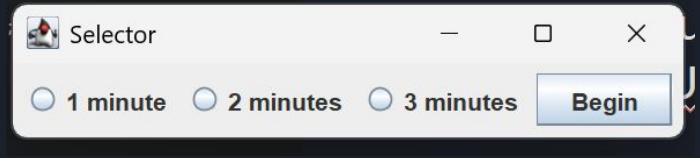


The SelectionPage Class

```java
package com.example;

import ...

public class SelectionPage extends JFrame implements Action
    JFrame frame = new JFrame();  11 usages
    JRadioButton shortTime;  6 usages
    JRadioButton midTime;  6 usages
    JRadioButton longTime;  6 usages
    JButton button;  5 usages
    int time = 0;  5 usages

    public SelectionPage(){  1 usage  Oscarz
        //Buttons
        shortTime = new JRadioButton( text: "1 minute");
        midTime = new JRadioButton( text: "2 minutes");
        longTime = new JRadioButton( text: "3 minutes");
        button = new JButton( text: "Begin");
        shortTime.setFocusable(false);
        midTime.setFocusable(false);
        longTime.setFocusable(false);
        button.setFocusable(false);

        //Group
        ButtonGroup group = new ButtonGroup();
        group.add(shortTime);
        group.add(midTime);
        group.add(longTime);

        //Event Listener
        shortTime.addActionListener( l: this);
        midTime.addActionListener( l: this);
        longTime.addActionListener( l: this);
        button.addActionListener( l: this);

        //Frame
        frame.add(shortTime);
        frame.add(midTime);
```

```java
        frame.add(longTime);
        frame.add(button);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setTitle("Selector");
        frame.setLayout(new FlowLayout());
        frame.pack();
        frame.setVisible(true);
    }

    @Override  Oscarz
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == shortTime){
            time = 1;
        }
        else if(e.getSource() == midTime){
            time = 2;
        }
        else if(e.getSource() == longTime){
            time = 3;
        }

        if(e.getSource() == button){
            if (time != 0){
                frame.dispose();
                paint();
            }
        }
    }

    public void paint(){  1 usage  Oscarz
        SwingUtilities.invokeLater(new Runnable() {  Oscarz
            @Override  Oscarz
            public void run() { new PaintGUI(time).setVisible(true); }
        });
    }
```

This class is a JFrame which contains 3 radial buttons that correspond to the minutes you want draw for. The value of minutes gets passed onto a new PaintGUI object.



PaintGUI: constructor

```
public class PaintGUI extends JFrame {  3 usages  Oscarz
    JFrame frame = new JFrame();  no usages
    private int startTime;  2 usages
    MenuBar menuBar;  3 usages
    Menu fileMenu;  4 usages
    MenuItem saveItem;  3 usages
    MenuItem importItem;  3 usages
    private Canvas canvas;  13 usages

    PaintGUI(int time){  1 usage  Oscarz

        //Menu
        menuBar = new MenuBar();
        fileMenu = new Menu( label: "File");
        saveItem = new MenuItem( label: "Save");
        importItem = new MenuItem( label: "Load");
        fileMenu.add(saveItem);
        fileMenu.add(importItem);

        menuBar.add(fileMenu);
        this.setMenuBar(menuBar);

        //Frame
        this.setPreferredSize(new Dimension( width: 1800,  height: 1000));
        this.pack();
        this.setFocusable(true);
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        startTime = time * 60;
        addGUI();
    }
```

The PaintGUI class is the backbone of the UI

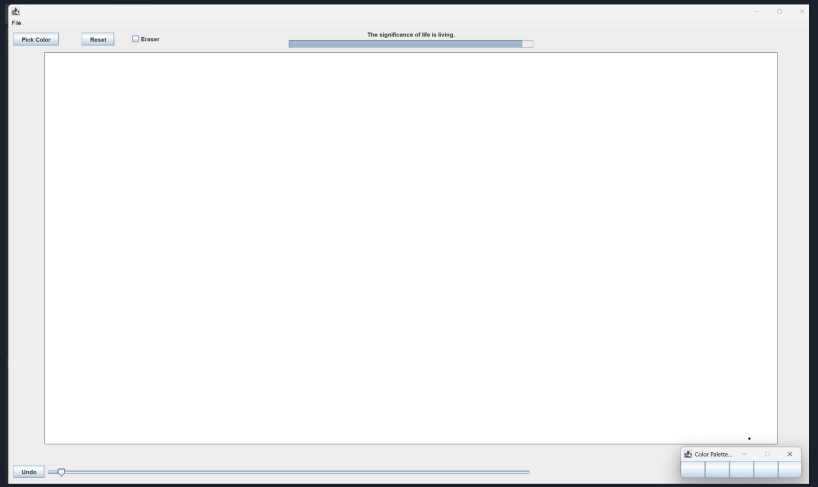
This is the constructor:

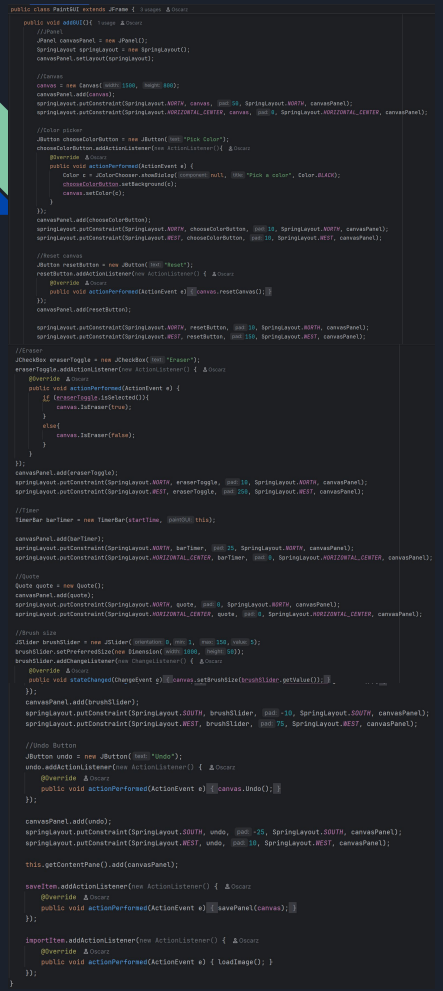
-sets the time

-sets up the menu items

-sets up the JFrame

-and calls a very important method addGUI





```
public class PaintGUI extends JFrame {
    public void addGUI(){
        //JPanel
        JPanel canvasPanel = new JPanel();
        SpringLayout springLayout = new SpringLayout();
        canvasPanel.setLayout(springLayout);

        //Canvas
        canvas = new Canvas( width: 1500,  height: 800);
        canvasPanel.add(canvas);
        springLayout.putConstraint(SpringLayout.NORTH, canvas,  pad: 50, SpringLayout.NORTH, canvasPanel);
        springLayout.putConstraint(SpringLayout.HORIZONTAL_CENTER, canvas,  pad: 0, SpringLayout.HORIZONTAL_CENTER, canvasPanel);

        //Color picker
        JButton chooseColorButton = new JButton( text: "Pick Color");
        chooseColorButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e) {
                Color c = JColorChooser.showDialog( component: null,  title: "Pick a color", Color.BLACK);
                chooseColorButton.setBackground(c);
                canvas.setColor(c);
            }
        });
        canvasPanel.add(chooseColorButton);
        springLayout.putConstraint(SpringLayout.NORTH, chooseColorButton,  pad: 10, SpringLayout.NORTH, canvasPanel);
        springLayout.putConstraint(SpringLayout.WEST, chooseColorButton,  pad: 10, SpringLayout.WEST, canvasPanel);

        //Reset canvas
        JButton resetButton = new JButton( text: "Reset");
        resetButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { canvas.resetCanvas(); }
        });
        canvasPanel.add(resetButton);

        springLayout.putConstraint(SpringLayout.NORTH, resetButton,  pad:10, SpringLayout.NORTH, canvasPanel);
        springLayout.putConstraint(SpringLayout.WEST, resetButton,  pad:150, SpringLayout.WEST, canvasPanel);

//Eraser
JCheckBox eraserToggle = new JCheckBox( text: "Eraser");
eraserToggle.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        if (eraserToggle.isSelected()){
            canvas.IsEraser(true);
        }
        else{
            canvas.IsEraser(false);
        }
    }
});
canvasPanel.add(eraserToggle);
springLayout.putConstraint(SpringLayout.NORTH, eraserToggle,  pad: 10, SpringLayout.NORTH, canvasPanel);
springLayout.putConstraint(SpringLayout.WEST, eraserToggle,  pad: 250, SpringLayout.WEST, canvasPanel);

//Timer
TimerBar barTimer = new TimerBar(startTime,  parentJLC: this);

canvasPanel.add(barTimer);
springLayout.putConstraint(SpringLayout.WORTH, barTimer,  pad: 25, SpringLayout.WORTH, canvasPanel);
springLayout.putConstraint(SpringLayout.HORIZONTAL_CENTER, barTimer,  pad: 0, SpringLayout.HORIZONTAL_CENTER, canvasPanel);

//Quote
Quote quote = new Quote();
canvasPanel.add(quote);
springLayout.putConstraint(SpringLayout.NORTH, quote,  pad: 0, SpringLayout.NORTH, canvasPanel);
springLayout.putConstraint(SpringLayout.HORIZONTAL_CENTER, quote,  pad: 0, SpringLayout.HORIZONTAL_CENTER, canvasPanel);

//Brush size
JSlider brushSlider = new JSlider( orientation: 0,  min: 1,  max: 150,  value: 5);
brushSlider.setPreferredSize(new Dimension( width: 1000,  height: 50));
brushSlider.addChangeListener(new ChangeListener() {
    @Override
    public void stateChanged(ChangeEvent e) { canvas.setBrushSize(brushSlider.getValue()); }
});
canvasPanel.add(brushSlider);
springLayout.putConstraint(SpringLayout.SOUTH, brushSlider,  pad: -10, SpringLayout.SOUTH, canvasPanel);
springLayout.putConstraint(SpringLayout.WEST, brushSlider,  pad: 75, SpringLayout.WEST, canvasPanel);

//Undo Button
JButton undo = new JButton( text: "Undo");
undo.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { canvas.Undo(); }
});

canvasPanel.add(undo);
springLayout.putConstraint(SpringLayout.SOUTH, undo,  pad: -25, SpringLayout.SOUTH, canvasPanel);
springLayout.putConstraint(SpringLayout.WEST, undo,  pad: 10, SpringLayout.WEST, canvasPanel);

this.getContentPane().add(canvasPanel);

saveItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { savePanel(canvas); }
});

importItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { loadImage(); }
});
}
```

PaintGUI: addGUI

The addGUI method creates a JPanel with a SpringLayout. Which acts a container for most of the elements such as other objects of different classes. It creates and adds the Quote object, TimerBar Object, Canvas Object,

Functionality calls on Canvas object's methods

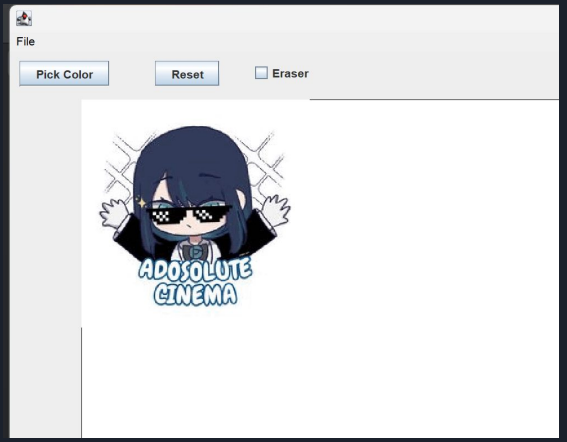
- creates undo button
- creates a Jslider for brush size
- creates a JColorChooser
- eraser checkbox
- undo button
- reset button

PaintGUI: more methods

```
public Canvas getCanvas(){  2 usages  Oscarz
    return canvas;
}

public void savePanel(JPanel panel){  2 usages  Oscarz
    BufferedImage imagebuf=null;
    JFileChooser fileChooser = new JFileChooser();
    int response = fileChooser.showSaveDialog( parent: null);

    if(response == JFileChooser.APPROVE_OPTION){
        try {
            imagebuf = new Robot().createScreenCapture(panel.bounds());
        } catch (AWTException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        Graphics2D graphics2D = imagebuf.createGraphics();
        panel.paint(graphics2D);
        try {
            ImageIO.write(imagebuf, formatName: "jpeg", new File(fileChooser.getSelectedFile().getAbsolutePath()));
        } catch (Exception e) {
            // TODO Auto-generated catch block
            System.out.println("error");
        }
    }
}
```

```
public void loadImage(){  1 usage  Oscarz
    JFileChooser fileChooser = new JFileChooser();
    int response = fileChooser.showOpenDialog( parent: null);

    if(response == JFileChooser.APPROVE_OPTION){
        File selectedFile = fileChooser.getSelectedFile();
        try {
            BufferedImage image = ImageIO.read(new File(selectedFile.getAbsolutePath()));
            canvas.setImage(image);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
}
```

- getCanvas: Just a getter method for the canvas object created
- savePanel: Lets you choose a file location in file explorer and save the canvas' image as a png.
- loadImage: Lets you select a file in file explorer to import into your canvas. Calls on a method in canvas object



Canvas class: constructor

```java
package com.example;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.util.ArrayList;
import java.util.List;

public class Canvas extends JPanel {  5 usages  Oscarz *

    private int strokeSize = 4;  9 usages
    private Color color;  11 usages
    private Color tempColor;  2 usages
    private int x;  5 usages
    private int y;  5 usages
    private int width;  2 usages
    private int height;  2 usages
    private ColorHistory colorHistory;  3 usages
    private Indicator indicator;  5 usages
    private BufferedImage image;  4 usages
```

```java
    public void mousePressed(MouseEvent e) {
        //get mouse location
        x = e.getX();
        y = e.getY();

        //draw in location
        Graphics g = getGraphics();
        g.setColor(color);
        g.fillRect(x, y, strokeSize, strokeSize);
        g.dispose();

        //start path
        currentPath = new ArrayList<>( initialCapacity: 200);
        currentPath.add(new ColorPoints(x, y, color, strokeSize));
    }

    @Override  Oscarz
    public void mouseReleased(MouseEvent e) {
        //reset current path
        addColorHistory();
        allPath.add(currentPath);
        currentPath = null;
    }

    @Override  Oscarz
    public void mouseDragged(MouseEvent e) {
        indicator.setVisible(false);
        //get mouse location
        x = e.getX();
        y = e.getY();

        //
        Graphics2D g2d = (Graphics2D) getGraphics();
        g2d.setColor(color);
        if(!currentPath.isEmpty()){
```

```java
            ColorPoints prevPoint = currentPath.get(currentPath.size() - 1);
            g2d.setStroke(new BasicStroke(strokeSize));

            //connect current to previous point by drawing a line
            g2d.drawLine(prevPoint.getX(), prevPoint.getY(), x, y);
        }
        g2d.dispose();

        //add new point to path
        ColorPoints nextPoint = new ColorPoints(e.getX(), e.getY(), color, strokeSize);
        currentPath.add(nextPoint);
    }

    @Override  Oscarz
    public void mouseMoved(MouseEvent e) {
        indicator.setVisible(true);
        indicator.setBounds(e.getX(), e.getY(),  width: 100,  height: 100);
        indicator.updateIndicator(strokeSize, color);
    }
};
```

ColorPoints Class

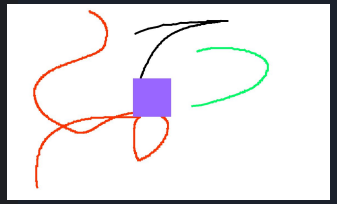
```java
public class ColorPoints {  9 usages  Oscarz
    private Color color;  2 usages
    private int strokeSize;  2 usages
    private int x;  2 usages
    private int y;  2 usages

    public ColorPoints(int x, int y, Color color, int strokeSize){  2 usages  Oscarz
        this.x = x;
        this.y = y;
        this.color = color;
        this.strokeSize = strokeSize;
    }

    public Color getColor() { return color; }

    public int getX() { return x; }

    public int getY() { return y; }

    public int getStrokeSize() { return strokeSize; }
}
```

This constructor does a lot, the first notable part of this is it's mouse listeners which tracks your mouse movement. Whenever you click you create a custom object the ColorPoints object is created with a color and x and y coordinates. These ColorPoints are added to a ColorPoints List called currentPath. This list represents a line (which is made out of smaller lines interpolated between points to make a smooth continuous line like a S curve). And there is a list of currentPaths called allPaths which represent all of the lines on the canvas.

Canvas class: constructor continued

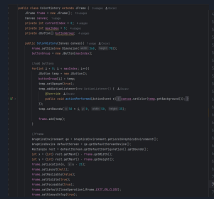
Indicator Class

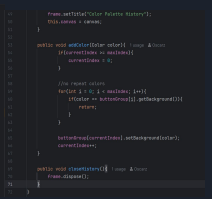
```
public class Indicator extends JPanel {  2 usages  Oscarz
    public int strokeSize;  3 usages
    private Color color;  2 usages

    public Indicator(){  1 usage  Oscarz
        this.setOpaque(false);
        this.setBackground(new Color( r: 0,  g: 0,  b: 0,  a: 0));
        this.setSize(new Dimension( width: 100,  height: 100));
    }

    public void updateIndicator(int strokeSize, Color color) {  1 usage  Oscarz
        this.strokeSize = strokeSize;
        this.color = color;
        repaint();
    }

    @Override  Oscarz
    protected void paintComponent(Graphics g) {
        Graphics2D g2 = (Graphics2D) g.create();
        g2.setColor(color);
        g2.fillRect( x: 0,  y: 0, strokeSize, strokeSize);
        g2.dispose();
    }
}
```

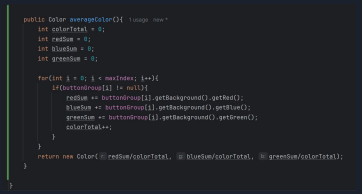


The constructor also creates a Indicator object which is to show your brush size as your mouse pointer. It is a JPanel which follows your mouse' xy. Its transparent and redraw the brush indicator based on color and stroke size. Updates the xy and color (using update indicator) in the mouseMoved listener method.

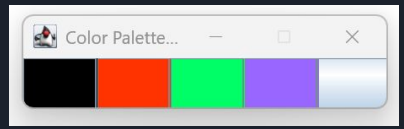
Color history class







The constructor also makes a ColorHistory Object. Which is a small JFrame that cycles between 5 colors which are buttons that can be pressed on to reselect recents colors that have just been used. The average color of this history can be returned too. The mouse released listener adds the color via the addColor method in ColorHistory



Canvas class: More methods

```
public void setColor(Color c) { color = c; }

public void resetCanvas(){  1 usage  Oscarz
    Graphics g = getGraphics();
    g.clearRect( x: 0,  y: 0, width, height);
    g.dispose();

    currentPath = null;
    allPath = new ArrayList<>( initialCapacity: 200);

    repaint();
    revalidate();
}

@Override  Oscarz *
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d= (Graphics2D) g;

    //draws picture
    if(image != null){
        g2d.drawImage(image,  x: 0,  y: 0,  observer: null);
    }

    //redraws lines
    for(List<ColorPoints> path: allPath){
        ColorPoints from = null;
        for(ColorPoints points: path){
            if(path.size() == 1){
                g2d.fillRect(points.getX(), points.getY(), strokeSize, strokeSize);
            }

            if(from != null){
                g2d.setColor(points.getColor());
                g2d.setStroke(new BasicStroke(points.getStrokeSize()));
                g2d.drawLine(from.getX(), from.getY(), points.getX(), points.getY());
            }
            from = points;
        }
    }
}
```

```
public void IsEraser(boolean isEraser){  2 usages  Oscarz
    if(isEraser){
        tempColor = color;
        color = Color.WHITE;
    }
    else{
        color = tempColor;
    }
}

public void setBrushSize(int brushSize) { strokeSize = brushSize; }

public void Undo(){  1 usage  Oscarz
    allPath.remove(allPath.getLast());
    paintComponent(getGraphics());
}

public void setImage(BufferedImage image){  1 usage  Oscarz
    Graphics g = getGraphics();
    g.drawImage(image,  x: 0,  y: 0,  observer: null);
}

public void addColorHistory(){  1 usage  Oscarz
    colorHistory.addColor(color);
}

public ColorHistory returnColorHistory(){  1 usage  Oscarz
    return colorHistory;
}
```

-set color: sets color

-reset canvas: clears the canvas

-paintComponent: to ensure the lines don't disappear when canvas is updated it redraws all of allPath and imported image

-isEraser: toggles brush color to white

-setBrushSize: sets brush size

-Undo: Delete most recent currentPaths from allPaths and reloads

setImage: draws image of import

addColor history: helps add color history

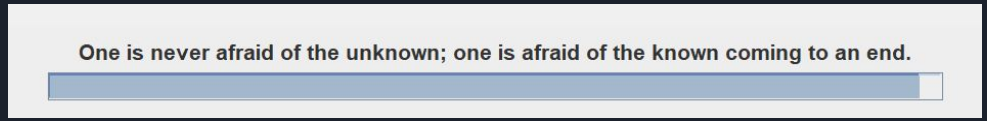
returnColor history: returns Color history

Quote Class

```java
public class Quote extends JPanel {  2 usages  Oscarz
    JSONArray quotes;  2 usages

    public Quote(){  1 usage  Oscarz
        //API quote
        try {
            quotes = new JSONArray(getData( endpoint: "https://zenquotes.io/api/random"));
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        JSONObject obj = quotes.getJSONObject( index: 0);
        String quote = obj.getString( key: "q");
        System.out.println(quote);

        this.setPreferredSize(new Dimension( width: 1000,  height: 50));
        //Jlabel
        JLabel display = new JLabel(quote);
        display.setSize( width: 1000,  height: 50);
        this.add(display);
//          this.setLayout(null);
```

```java
public static String getData(String endpoint) throws Exception {  1 usage  Oscarz
    /*endpoint is a url (string) that you get from an API website*/
    URL url = new URL(endpoint);
    /*connect to the URL*/
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    /*creates a GET request to the API:. Asking the server to retrieve information for our program*/
    connection.setRequestMethod("GET");
    /* When you read data from the server, it wil be in bytes, the InputStreamReader will convert it to text.
    The BufferedReader wraps the text in a buffer so we can read it line by line*/
    BufferedReader buff = new BufferedReader(new InputStreamReader(connection.getInputStream()));
    String inputLine;//variable to store text, line by line
    /*A string builder is similar to a string object but faster for larger strings,
    you can concatenate to it and build a larger string. Loop through the buffer
    (read line by line). Add it to the stringbuilder */
    StringBuilder content = new StringBuilder();
    while ((inputLine = buff.readLine()) != null) {
        content.append(inputLine);
    }
    buff.close(); //close the bufferreader
    connection.disconnect(); //disconnect from server
    return content.toString(); //return the content as a string
}
```

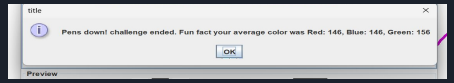
Uses's Ms. Turin's getData method to pull a JSONarray from zen quotes. Stores the first element in a JSONobject and gets the string of the object to input into a JLabel which is by extension put into the PaintGUI as mentioned previously.

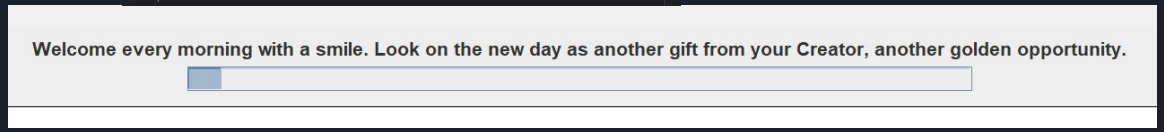


TimerBar Class

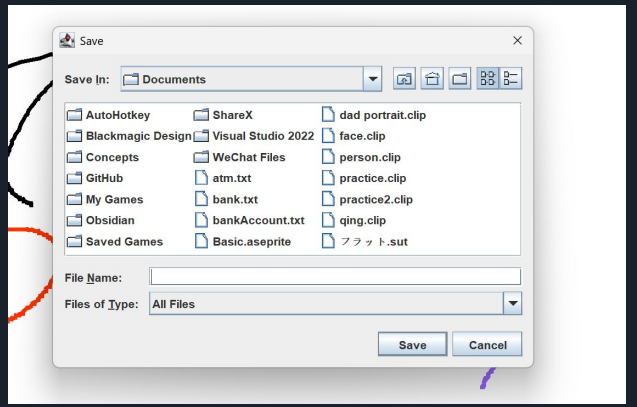
```java
package com.example;

import ...

public class TimerBar extends JProgressBar {  2 usages  Oscarz
    private JProgressBar barTimer = new JProgressBar();  7 usages
    private Timer timer;  2 usages
    private int startTime;  1 usage
    private boolean ended;  3 usages
    private PaintGUI paintGUI;  6 usages

    public TimerBar(int startTime, PaintGUI paintGUI){  1 usage  Oscarz
        this.startTime = startTime;
        ended = false;
        this.setPreferredSize(new Dimension( width: 500,  height: 15));
        barTimer.setBounds( x: 0,  y: 0 ,  width: 500,  height: 15);
        barTimer.setMaximum(startTime);
        barTimer.setValue(startTime);
        this.add(barTimer);
        this.paintGUI = paintGUI;
        countDown();
        timer.start();
    }
```

```java
public void countDown(){  1 usage  Oscarz
    timer = new Timer( delay: 1000, new ActionListener() {  Oscarz
        @Override  Oscarz
        public void actionPerformed(ActionEvent e) {
            barTimer.setValue(barTimer.getValue() - 1);
            if(barTimer.getValue() == 0 && ended == false){
                ended = true;
                Color averageColor = paintGUI.getCanvas().returnColorHistory().averageColor();
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Pens down! challenge ended. Fun fa" +
                        "ct your average color was Red: " + averageColor.getRed() + ", Blue: "
                        + averageColor.getBlue() + ", Green: " + averageColor.getGreen(),  title: "title",
                        JOptionPane.INFORMATION_MESSAGE);
                paintGUI.savePanel(paintGUI.getCanvas());
                paintGUI.getCanvas().returnColorHistory().closeHistory();
                paintGUI.dispose();
                LaunchPage launchPage = new LaunchPage();
            }
        }
    });
}
```





The constructor creates a JProgressbar that starts from the max value. Then in the countDown method it counts the time down. When it reaches 0 it gives you a JOptionPane message (which gives you your average color from color history), asks you to save your image, and closes all windows to loop back into the Launchpage.



Features added

✔Base Project (+88%)

- Drawing app

✔Statistics / ML / Basic Computations (+6%)

- Calculate the average color

✔ GUI (+2%)

- Java Swing to build a drawing app GUI

✔ Save/Load (+2%)

- Ability to save and load a PNG

= 98%

What did I learn?

- I learned how to use Swing for GUI
- Got experience for making larger projects
- Learned about some of the drawing features in Swing
- How to deal with a basic API

Quote: "Don't let the fear of losing be greater than the excitement of winning"

Pick Color Reset Eraser

Great minds are always feared by lesser minds.

Undo

Color Palette...