Practical No. 20

Aim : Mini project - Painting Applet

PAGE NO.
DATE

Practical No. 20

Aim : Mini project → Painting Applet.

Theory:

Painting Applet :

The aim of this project is to enable the user to draw, erase and create, drawings, paintings, art, etc. This applet can also be used as a whiteboard.

The applet consists of 7 color options ~~colors~~ and 1 clear button and a x time indicator in the status bar which is x updated after every 1000 ms, using threads.

The color buttons are rectangles of approximately 12.35 % height of the applet height so that all color rectangles and clear button can fit on the right ~~hand~~ hand side of the applet.

The timer on the bottom-left i.e. the status bar uses the Calendar class to get the current hour of the day, minute of the day, and second of the day. This data is converted into 'HH:MM:SS' format and saved as string in a ~~variable~~ variable called timestring, which is passed to status ~~bar~~ bar, then the thread goes to sleep for 1000 ms, by the use of wait(1000); call.

The drawing on the applet is happening by the use of mousedragged() motion listener, to get the x axis and y axis, this x and y axis is used to draw an oval in the current color and width.

Conclusion:

Hence, we created our mini project and used various java concepts in it to make the paint applet work.

Code:

SimplePaint.java

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

/*<applet code="SimplePaint.class" height="1080" width="1920">  */
public class SimplePaint extends Applet implements MouseListener,
MouseMotionListener, KeyListener,Runnable {
    private int currentColor = BLACK;

    private final static int
                BLACK = 0,
                RED = 1,
                GREEN = 2,
                BLUE = 3,
                CYAN = 4,
                MAGENTA = 5,
                YELLOW = 6,
                 WHITE = 7;

    int STROKE = 2;

    Thread t = null;
    int hours = 0, minutes = 0, seconds = 0;
    String timeString = "";

    private int prevX, prevY; // previous values clicked/ dragged by users

    private boolean dragging;    // checking if the user is currently dragging
the mouse or not

    private Graphics g;

   public void init() {
      addMouseListener(this);
      addMouseMotionListener(this);
      addKeyListener(this);
    }
```

```
public void update(Graphics g) {
   paint(g);
}

public void paint(Graphics g) {

  int width = getSize().width;
  int height = getSize().height;

  int colorSpacing = (height - 56) / 8;


  //drawing screen
  g.setColor(Color.white);
  g.fillRect(3, 3, width - 59, height - 6);


  //backgroundscreen
  g.setColor(Color.gray);
  g.drawRect(0, 0, width-1, height-1);
  g.drawRect(1, 1, width-3, height-3);
  g.drawRect(2, 2, width-5, height-5);


  g.fillRect(width - 56, 0, 56, height);


  g.setColor(Color.white);
  g.fillRect(width-53,  height-53, 50, 50);
  g.setColor(Color.black);
  g.drawRect(width-53, height-53, 49, 49);
  g.drawString("CLEAR", width-48, height-23);

    // For color button colors
  g.setColor(Color.black);
  g.fillRect(width-53, 3 + 0*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.red);
  g.fillRect(width-53, 3 + 1*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.green);
  g.fillRect(width-53, 3 + 2*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.blue);
  g.fillRect(width-53, 3 + 3*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.cyan);
  g.fillRect(width-53, 3 + 4*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.magenta);
  g.fillRect(width-53, 3 + 5*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.yellow);
  g.fillRect(width-53, 3 + 6*colorSpacing, 50, colorSpacing-3);
```

```
  // BOOKMARK
    g.setColor(Color.white);
    g.fillRect(width-53, 3 + 7*colorSpacing, 50, colorSpacing-3);

    //for background of clear button
  g.setColor(Color.white);
  g.drawRect(width-55, 1 + currentColor*colorSpacing, 53, colorSpacing);
  g.drawRect(width-54, 2 + currentColor*colorSpacing, 51, colorSpacing-2);

  g.setColor(Color.BLACK);
  g.drawString("ERASE", width-48, height-105);

}

private void changeColor(int y) {

// border of the selected color
  int width = getSize().width;
  int height = getSize().height;
  int colorSpacing = (height - 56) / 8;
  int newColor = y / colorSpacing;

  if (newColor < 0 || newColor > 7)
     return;

  Graphics g = getGraphics();
  g.setColor(Color.gray);
  g.drawRect(width-55, 1 + currentColor*colorSpacing, 53, colorSpacing);
  g.drawRect(width-54, 2 + currentColor*colorSpacing, 51, colorSpacing-2);
  currentColor = newColor;
  //current border set to white
  g.setColor(Color.white);
  g.drawRect(width-105, 1 + currentColor*colorSpacing, 53, colorSpacing);
  g.drawRect(width-54, 2 + currentColor*colorSpacing, 51, colorSpacing-2);
  g.dispose();

}

public void start ()
{
    t = new Thread (this);
    t.start ();
}

public void run ()
{
```

```
        try
        {


            while (true)
            {
                String temp;
                Calendar cal = Calendar.getInstance ();
                hours = cal.get (Calendar.HOUR_OF_DAY);
                minutes = cal.get (Calendar.MINUTE);
                seconds = cal.get (Calendar.SECOND);
                 if (hours > 12){
                    temp = " PM";
                    hours -= 12;
                 }else{
                    temp = " AM";
                 }
                SimpleDateFormat formatter = new SimpleDateFormat("hh:mm:ss");
                Date date = cal.getTime ();
                timeString = formatter.format (date);
                timeString += temp;
                t.sleep (1000); // interval given in milliseconds
                showStatus(timeString);
            }
        }
        catch (Exception e)
        {
        }
    }

    private void setUpDrawingGraphics() {

      g = getGraphics();
      switch (currentColor) {
         case BLACK:
            g.setColor(Color.black);
            break;
         case RED:
            g.setColor(Color.red);
            break;
         case GREEN:
            g.setColor(Color.green);
            break;
         case BLUE:
            g.setColor(Color.blue);
            break;
         case CYAN:
            g.setColor(Color.cyan);
```

```
            break;
         case MAGENTA:
            g.setColor(Color.magenta);
            break;


         case YELLOW:
            g.setColor(Color.yellow);
            break;
          case WHITE:
              g.setColor(Color.white);
              break;
      }
   }

   public void mousePressed(MouseEvent evt) {


      int x = evt.getX();
      int y = evt.getY();

      int width = getSize().width;
      int height = getSize().height;

      if (dragging == true)
          return;

      if (x > width - 53) {

         if (y > height - 53)
            repaint();
         else
            changeColor(y);
      }
      else if (x > 3 && x < width - 56 && y > 3 && y < height - 3) {

         prevX = x;
         prevY = y;
         dragging = true;
         setUpDrawingGraphics();
      }

   }     // end mousePressed()

   public void mouseReleased(MouseEvent evt) {

       if (dragging == false)
          return;  // Nothing to do because the user isn't drawing.
```

```
        dragging = false;
        g.dispose();
        g = null;
    }

    public void mouseDragged(MouseEvent evt) {


        if (dragging == false)
            return;  // Nothing to do because the user isn't drawing.

        int x = evt.getX();   // x-coordinate of mouse.
        int y = evt.getY();   // y=coordinate of mouse.

        if (x < 3)                          // Adjust the value of x,
            x = 3;                          //   to make sure it's in
        if (x > getSize().width - 57)       //   the drawing area.
            x = getSize().width - 57;

        if (y < 3)                          // Adjust the value of y,
            y = 3;                          //   to make sure it's in
        if (y > getSize().height - 4)       //   the drawing area.
            y = getSize().height - 4;

//        g.drawLine(prevX, prevY, x, y);  // Draw the line.
        g.fillOval (prevX, prevY, STROKE, STROKE);
        g.fillOval (x, y, STROKE, STROKE);

        prevX = x;  // Get ready for the next line segment in the curve.
        prevY = y;

    } // end mouseDragged.

     public void keyTyped(KeyEvent evt) {
        char c = evt.getKeyChar();
        switch (c){
            case '1':
                STROKE = 1;
                break;
            case '2':
                STROKE = 2;
                break;
            case '3':
                STROKE = 3;
                break;
            case '4':
                STROKE = 4;
                break;
```

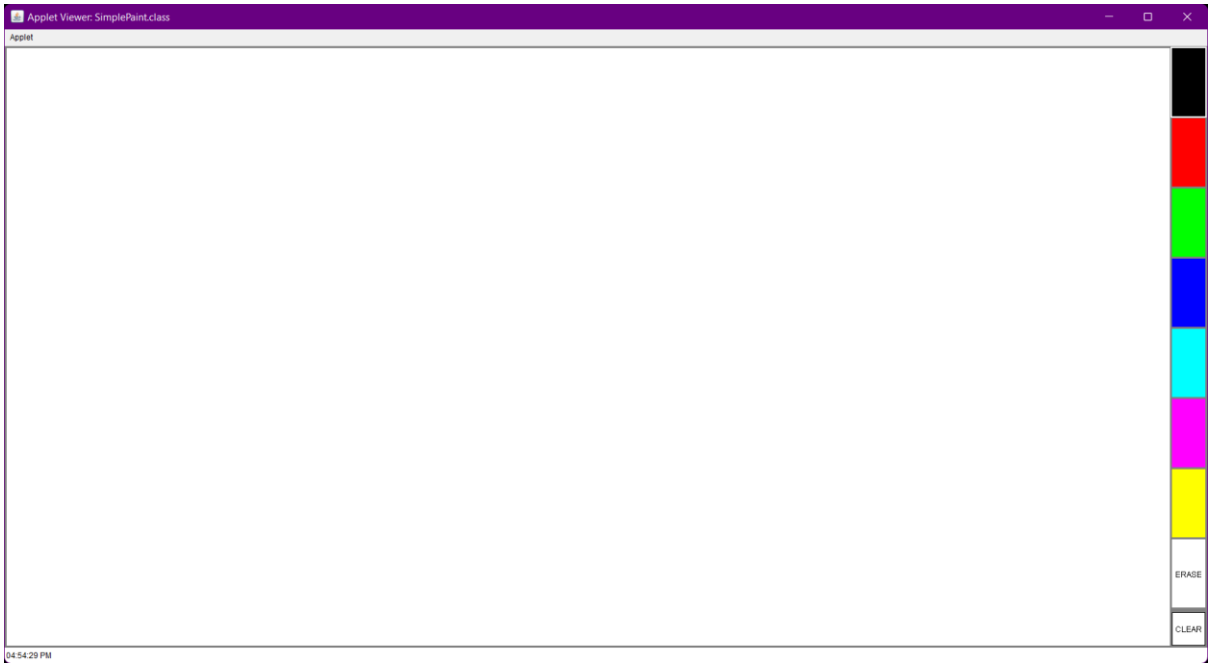
```
            case '5':
                STROKE = 5;
                break;
            case '6':
                STROKE = 6;
                break;


            case '7':
                STROKE = 7;
                break;
            case '8':
                STROKE = 8;
                break;
            case '9':
                STROKE = 9;
                break;
            case 'a':
                if(STROKE==1){}
                else {
                    STROKE -= 1;
                }
                break;
            case 'd':
                STROKE += 1;
                break;
            case 'w':
                if(currentColor <= 0 ){
                    currentColor=6;
                }else{
                    currentColor-=1;
                }
                break;
            case 's':
                currentColor = (currentColor+1) % 7;
                break;
            default:
                return;
        }
    }

    public void keyPressed(KeyEvent evt) {
        char c = evt.getKeyChar();
        switch (c){
            case 'a':
                if(STROKE==1){}
                else {
                    STROKE -= 1;
```

```
                }
                break;
            case 'd':
                STROKE += 1;
                break;
            default:
                return;
        }


        System.out.println(STROKE);
    }

    public void keyReleased(KeyEvent evt) { }
    public void mouseEntered(MouseEvent evt) { }   // Some empty routines.
   public void mouseExited(MouseEvent evt) { }      //  (Required by the
MouseListener
   public void mouseClicked(MouseEvent evt) { }   //    and
MouseMotionListener
   public void mouseMoved(MouseEvent evt) { }     //    interfaces).
} // end class SimplePaint
```

SimplePaint.html

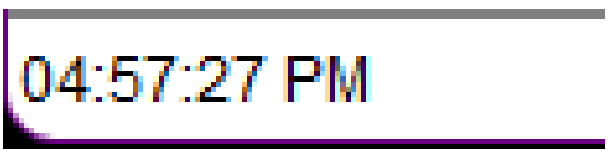
```
<html>

<applet code="SimplePaint.class" height="900" width="1800" >  </applet>

</html>
```

Output:



Painting Applet First View

Drawing in the applet



Live time in the applet implemented by using multi-threading concept

PAGE NO.
DATE

Conclusion:

Hence, we created our miniproject and used various java projects concepts in it to make the paint applet work.