

```
/**
 * FColorGrabber - a color grabber in java.
 *
 * Author: Fredrik Fornwall <fredrikfornwall@gmail.com>
 */
package net.fornwall.fcolorgrabber;

import java.awt.AWTException;
import java.awt.Color;
import java.awt.Cursor;
import java.awt.Graphics;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.image.BufferedImage;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JButton;
import javax.swing.JColorChooser;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.UIManager;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;

public class FColorGrabber {

    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (Exception e) {
        }

        JFrame picker = new JFrame();
        picker.setTitle("FColorGrabber");
        picker.setContentPane(new MainPanel());
        picker.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        picker.pack();
        picker.setVisible(true);
    }
}

class FullScreenPicker extends JFrame implements MouseListener {

    private GraphicsDevice device;

    private GraphicsEnvironment env;

    private BufferedImage image;

    private MainPanel panel;

    private Robot robot;

    public FullScreenPicker(MainPanel panel) {
        this.panel = panel;
        setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR_CURSOR));

        setContentPane(new JPanel() {
            public void paint(Graphics g) {
                g.drawImage(image, 0, 0, null);
            }
        });
    }
}
```

```

        addMouseListener(this);

        env = GraphicsEnvironment.getLocalGraphicsEnvironment();
        device = env.getDefaultScreenDevice();
        setUndecorated(true);
    }

    public void mouseClicked(MouseEvent e) {
        panel.setNewColor(new Color(image.getRGB(e.getX(), e.getY())));
        dispose();
    }

    public void mouseEntered(MouseEvent arg0) {
    }

    public void mouseExited(MouseEvent arg0) {
    }

    public void mousePressed(MouseEvent arg0) {
    }

    public void mouseReleased(MouseEvent arg0) {
    }

    public void paint(Graphics g) {
        g.drawImage(image, 0, 0, null);
    }

    public void start() throws AWTException {
        if (robot == null) {
            robot = new Robot();
        }
        image = robot
            .createScreenCapture(new Rectangle(Toolkit.getDefaultToolkit().getScreenSize()));

        setVisible(true);

        if (this != device.getFullScreenWindow()) {
            device.setFullScreenWindow(this);
        }
    }
}

class MainPanel extends JPanel implements ActionListener, DocumentListener, PropertyChangeListener {

    private JColorChooser colorChooser = new JColorChooser();

    private JDialog colorChooserDialog;

    private JTextField colorField = new JTextField(8);

    private JButton editButton = new JButton("Edit");

    private FullScreenPicker fullScreen = new FullScreenPicker(this);

    private JButton exitButton = new JButton("Exit");

    private JButton selectButton = new JButton("Select");

    public MainPanel() {
        editButton.addActionListener(this);
        selectButton.addActionListener(this);
        exitButton.addActionListener(this);

        editButton.setMnemonic(KeyEvent.VK_E);
        selectButton.setMnemonic(KeyEvent.VK_S);
        exitButton.setMnemonic(KeyEvent.VK_X);

        colorChooser.addPropertyChangeListener(this);
        colorChooserDialog = JColorChooser.createDialog(null, "Edit Color", true, colorChooser,
            null, null);

        colorField.getDocument().addDocumentListener(this);

        setLayout(new GridBagLayout());
        GridBagConstraints gc = new GridBagConstraints();

```

```
gc.insets = new Insets(5, 5, 5, 5);

gc.gridx = gc.gridy = 0;
add(colorField, gc);

gc.gridx++;
add(selectButton, gc);

gc.gridx++;
add(editButton, gc);

gc.gridx++;
add(exitButton, gc);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == exitButton) {
        System.exit(0);
    } else if (e.getSource() == editButton) {
        colorChooser.setColor(colorChooser.getColor());
        colorChooserDialog.setVisible(true);
    } else if (e.getSource() == selectButton) {
        try {
            fullScreen.start();
        } catch (AWTException ee) {
            ee.printStackTrace();
        }
    }
}

public void keyPressed(KeyEvent arg0) {
}

public void keyReleased(KeyEvent arg0) {
}

public void keyTyped(KeyEvent e) {
    String text = colorField.getText() + e.getKeyChar();
    if (text.length() == 7) {
        try {
            String colorString = text.substring(1);
            setNewColor(new Color(Integer.parseInt(text.substring(1), 16)));
            e.consume();
            return;
        } catch (NumberFormatException ee) {
        }
    }
    colorField.setBackground(Color.RED);
}

private void validateColorField() {
    String text = colorField.getText();
    if (text.length() == 7) {
        try {
            String colorString = text.substring(1);
            int rgb = Integer.parseInt(text.substring(1), 16);
            setNewColor(new Color(rgb), false);
            return;
        } catch (NumberFormatException ee) {
        }
    }
    colorField.setBackground(Color.RED);
}

public void propertyChange(PropertyChangeEvent arg0) {
    setNewColor(colorChooser.getColor());
}

void setNewColor(Color newColor) {
    setNewColor(newColor, true);
}

void setNewColor(Color newColor, boolean updateField) {
    colorChooser.setColor(newColor);
    setBackground(newColor);
}
```

```
    colorField.setBackground(Color.WHITE);
    if (updateField) {
        String colorString = Integer.toHexString(colorChooser.getColor().getRGB()).substring(2)
            .toUpperCase();
        colorField.setText('#' + colorString);
    }
}

public void insertUpdate(DocumentEvent e) {
    validateColorField();
}

public void removeUpdate(DocumentEvent e) {
    validateColorField();
}

public void changedUpdate(DocumentEvent e) {
    validateColorField();
}
}
```