

```
/**
 * FScreenGrabber - a java application for grabbing screen shots.
 *
 * by Fredrik Fornwall, fredrikfornwall@gmail.com (http://fornwall.net/)
 */
package net.fornwall.fscreengrabber;

import java.awt.AWTException;
import java.awt.Color;
import java.awt.Cursor;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Image;
import java.awt.Insets;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.Transferable;
import java.awt.datatransfer.UnsupportedFlavorException;
import java.awt.dnd.DnDConstants;
import java.awt.dnd.DragGestureEvent;
import java.awt.dnd.DragGestureListener;
import java.awt.dnd.DragSource;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ComponentEvent;
import java.awt.event.ComponentListener;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.UIManager;
import javax.swing.filechooser.FileFilter;

/**
 * @author Fredrik Fornwall
 */
public class FScreenGrabber {
    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (Exception e) {
        }

        JFrame grabber = new JFrame();
        grabber.setTitle("FScreenGrabber");
        grabber.setContentPane(new MainPanel());
        grabber.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        grabber.pack();
        grabber.setVisible(true);
    }
}
```

```
}
}

class ImageTransferable implements Transferable {
    private DataFlavor[] dataFlavors = new DataFlavor[]{DataFlavor.imageFlavor};

    private Image image;

    public ImageTransferable(Image image) {
        this.image = image;
    }

    public DataFlavor[] getTransferDataFlavors() {
        return dataFlavors;
    }

    public boolean isDataFlavorSupported(DataFlavor flavor) {
        return flavor.equals(DataFlavor.imageFlavor);
    }

    public Object getTransferData(DataFlavor flavor) throws UnsupportedFlavorException, IOException {
        if (!flavor.equals(DataFlavor.imageFlavor)) {
            throw new UnsupportedFlavorException(flavor);
        }
        return image;
    }
}

class FullScreenPicker extends JFrame implements MouseListener, MouseMotionListener {

    private ImageDisplay imageDisplay;

    private GraphicsDevice device;

    private GraphicsEnvironment env;

    private BufferedImage image;

    private Robot robot;

    private int pressY = -1;

    private int pressX = -1;

    boolean mousePressed = false;

    public FullScreenPicker(ImageDisplay imageDisplay) {
        this.imageDisplay = imageDisplay;

        addMouseMotionListener(this);
        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) {
    }
}
```

```
public void mouseDragged(MouseEvent arg0) {
}

public void mouseEntered(MouseEvent arg0) {
}

public void mouseExited(MouseEvent arg0) {
}

private int lastX = -1;

private int lastY = -1;

public void mouseMoved(MouseEvent e) {
    if (mousePressed) {
        Graphics g = getGraphics();
        g.setXORMode(Color.WHITE);

        int x = Math.min(pressX, lastX);
        int y = Math.min(pressY, lastY);
        int width = Math.abs(pressX - lastX);
        int height = Math.abs(pressY - lastY);
        if (lastX != -1 && lastY != -1) {
            // Erase old rectangle
            g.setXORMode(Color.WHITE);
            g.drawRect(x, y, width, height);
            g.setXORMode(Color.BLUE);
            g.fillRect(x, y, width, height);
        }

        lastX = e.getX();
        lastY = e.getY();

        x = Math.min(pressX, lastX);
        y = Math.min(pressY, lastY);
        width = Math.abs(pressX - lastX);
        height = Math.abs(pressY - lastY);

        g.setXORMode(Color.WHITE);
        g.drawRect(x, y, width, height);
        g.setXORMode(Color.BLUE);
        g.fillRect(x, y, width, height);
    }

    lastX = e.getX();
    lastY = e.getY();
}

public void mousePressed(MouseEvent e) {
    if (mousePressed) {
        if (e.getX() == pressX || e.getY() == pressY) {
            return;
        }
        int x = Math.min(pressX, lastX);
        int y = Math.min(pressY, lastY);
        int width = Math.abs(pressX - lastX);
        int height = Math.abs(pressY - lastY);
        imageDisplay.setImage(image.getSubimage(x, y, width, height));
        dispose();
    } else {
        mousePressed = true;
        pressX = e.getX();
        pressY = e.getY();
    }
}

public void mouseReleased(MouseEvent arg0) {
```

```
}

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

public void start() throws AWTException {
    mousePressed = false;

    if (robot == null) {
        robot = new Robot();
    }
    image = robot
        .createScreenCapture(new Rectangle(Toolkit.getDefaultToolkit().getScreenSize()));

    setVisible(true);

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

}

class ImageDisplay extends JPanel implements ComponentListener, DragGestureListener {

    BufferedImage image;

    Image scaledImage;

    boolean scale = false;

    public Dimension getPreferredSize() {
        if (image == null) {
            return new Dimension(600, 600);
        } else {
            return new Dimension(image.getWidth(), image.getHeight());
        }
    }

    public ImageDisplay() {
        addComponentListener(this);
        DragSource.getDefaultDragSource().createDefaultDragGestureRecognizer(this,
DnDConstants.ACTION_COPY, this);
    }

    public void paint(Graphics g) {
        if (image != null) {
            if (!scale) {
                g.setColor(getBackground());
                g.fillRect(0, 0, getWidth(), getHeight());
            }
            g.drawImage((scale) ? scaledImage : image, 0, 0, null);
        } else {
            g.fillRect(0, 0, getWidth(), getHeight());
        }
    }

    public BufferedImage getImage() {
        if (scale) {
            BufferedImage result = (BufferedImage) createImage(getWidth(), getHeight());
            result.getGraphics().drawImage(scaledImage, 0, 0, null);
            return result;
        } else {
            return image;
        }
    }
}
```

```
public void setImage(BufferedImage image) {
    this.image = image;
    if (scale) {
        doScale();
    } else {
        repaint();
    }
}

public void setScaled(boolean scale) {
    this.scale = scale;
    if (scale) {
        doScale();
    } else {
        repaint();
    }
}

private void doScale() {
    if (image != null) {
        scaledImage = image.getScaledInstance(getWidth(), getHeight(), Image.SCALE_SMOOTH);
        repaint();
    }
}

public void componentResized(ComponentEvent e) {
    if (scale && image != null) {
        doScale();
    }
}

public void componentMoved(ComponentEvent arg0) {
}

public void componentShown(ComponentEvent arg0) {
}

public void componentHidden(ComponentEvent arg0) {
}

public void dragGestureRecognized(DragGestureEvent dge) {
    dge.startDrag(DragSource.DefaultCopyDrop, new ImageTransferable(scale ? getImage(): image));
}

}

class MainPanel extends JPanel implements ActionListener {
    private JButton exitButton = new JButton(new AbstractAction("Exit") {
        {
            putValue(Action.SHORT_DESCRIPTION, "The short description");
            putValue(Action.LONG_DESCRIPTION, "A really long description");
        }

        public void actionPerformed(ActionEvent e) {
            System.exit(0);
        }
    });

    private JButton grabButton = new JButton("Grab");

    private ImageDisplay imageDisplay = new ImageDisplay();

    private FullScreenPicker picker = new FullScreenPicker(imageDisplay);

    private JButton saveButton = new JButton("Save");

    private JCheckBox scaleCheck = new JCheckBox("Scale");
}
```

```

private JFileChooser saveFileChooser = new JFileChooser();

public MainPanel() {
    setLayout(new GridBagLayout());
    GridBagConstraints gc = new GridBagConstraints();
    gc.insets = new Insets(5, 5, 5, 5);

    gc.gridx = gc.gridy = 0;
    gc.weightx = 1.0;
    gc.weighty = 1.0;
    gc.gridwidth = 6;
    gc.fill = GridBagConstraints.BOTH;
    add(imageDisplay, gc);

    gc.anchor = GridBagConstraints.EAST;
    gc.weightx = 1.0;
    gc.weighty = 0.0;
    gc.fill = GridBagConstraints.HORIZONTAL;
    gc.gridy++;
    gc.gridwidth = 1;

    gc.gridx++;
    add(scaleCheck, gc);
    gc.gridx++;
    add(grabButton, gc);
    gc.gridx++;
    add(saveButton, gc);
    gc.gridx++;
    add(exitButton, gc);

    grabButton.addActionListener(this);
    saveButton.addActionListener(this);
    //exitButton.addActionListener(this);
    scaleCheck.addActionListener(this);

    scaleCheck.setMnemonic(KeyEvent.VK_C);
    grabButton.setMnemonic(KeyEvent.VK_G);
    saveButton.setMnemonic(KeyEvent.VK_S);
    exitButton.setMnemonic(KeyEvent.VK_X);

    saveButton.setEnabled(false);

    saveFileChooser.setFileFilter(new FileFilter() {
        public boolean accept(File f) {
            return f.isDirectory() || f.getName().toLowerCase().endsWith(".png");
        }

        public String getDescription() {
            return "png";
        }
    });
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == exitButton) {
        System.exit(0);
    } else if (e.getSource() == grabButton) {
        try {
            picker.start();
            saveButton.setEnabled(true);
        } catch (AWTException ex) {
            JOptionPane.showMessageDialog(this, "Error doing screen grab:\n" + ex.getMessage());
        }
    } else if (e.getSource() == scaleCheck) {
        imageDisplay.setScaled(scaleCheck.isSelected());
    } else if (e.getSource() == saveButton) {
        if (saveFileChooser.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {

```

```
    try {
        ImageIO.write((BufferedImage) imageDisplay.getImage(), "png", saveFileChooser
            .getSelectedFile());
    } catch (IOException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error saving image:\n" + ex.getMessage());
    }
}
}
```