LOVELY PROFESSIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE

Mini Project Report on

Image Editor

Course: - Advance Java (CSE406)

Submitted to: - Lovely Professional University

Faculty Name: - Dr Om Prakash Yadav

Uid: - 26121

Submitted By: -

Name: - Md Amjad Ansari

Reg No: - 12114768

Roll No: - RKE093A29

Section: - KE093

IMAGE EDITOR

In today's digital age, images play a significant role in various aspects of our lives, from personal photography and social media to professional graphic design and advertising. As such, the ability to manipulate and enhance digital images has become increasingly important. The Image Editor project aims to address this need by developing a versatile and user-friendly software application that empowers users to edit, enhance, and transform their digital images with ease.

Purpose: The purpose of the Image Editor project is to provide users with a powerful yet accessible tool for performing a wide range of image editing tasks. Whether it's retouching photos, creating digital art, or designing graphics, the Image Editor offers a comprehensive set of features and functionalities to cater to diverse user needs and preferences.

Scope: The scope of the Image Editor project encompasses the development of a feature-rich software application capable of performing various image editing operations. This includes fundamental tasks such as cropping, resizing, and rotating images, as well as more advanced functionalities like applying filters, effects, and adjustments to enhance the visual appeal of images. Additionally, the project may include tools for drawing, text overlay, and batch processing to further extend its capabilities.

Objectives: The key objectives of the Image Editor project include:

- **Versatility:** Develop a wide range of editing tools and functionalities to accommodate different types of images editing tasks and creative expressions.
- **Usability:** Design an intuitive and user-friendly interface that simplifies the image editing process, making it accessible to users of all skill levels.

- **Performance:** Ensure smooth and responsive performance, even when working with large or high-resolution images, through optimization and efficient algorithms.
- **Compatibility:** Build the Image Editor as a cross-platform application, compatible with major operating systems, to maximize accessibility and usability for users.
- **Customization:** Provide users with options for customization and personalization, allowing them to tailor the editing environment to their preferences and workflow.

Conclusion: The Image Editor project endeavours to empower users to unleash their creativity and imagination through digital image editing. By offering a comprehensive set of tools, intuitive interface, and robust performance, the Image Editor aims to become a valuable asset for photographers, graphic designers, artists, and anyone else who works with digital images.

Implementation

1. ImagePackage:

➢ FilterClass

```
package ImagePackage;


import java.awt.Color;

import java.awt.Image;

import java.awt.image.*;



public class FiltersClass {
```

```
public BufferedImage blurImage(BufferedImage BI)
{
   Kernel k = new Kernel(3, 3, new float[]{1f/(3*3),1f/(3*3),1f/(3*3),
         1f/(3*3),1f/(3*3),1f/(3*3),
         1f/(3*3),1f/(3*3),1f/(3*3)});
   ConvolveOp op = new ConvolveOp(k);
   return op.filter(BI, null);
}

/**
 *
 * @param BI
 * @return
 */
public Image lightenImage(BufferedImage BI)
{
   RescaleOp op = new RescaleOp(2f, 0, null);
   return op.filter(BI, null);
}
public Image darkenImage(BufferedImage BI)
{
   RescaleOp op = new RescaleOp(.5f, 0, null);
   return op.filter(BI, null);
}
public Image invertImage(BufferedImage BI)
{
```

```
//        AffineTransform tx = AffineTransform.getScaleInstance(-1, 1);
//        tx.translate(-BI.getWidth(null), 0);
//        AffineTransformOp op = new AffineTransformOp(tx,
AffineTransformOp.TYPE_NEAREST_NEIGHBOR);
//        return op.filter(BI, null);
        for (int x = 0; x < BI.getWidth(); x++) {
            for (int y = 0; y < BI.getHeight(); y++) {
                int rgba = BI.getRGB(x, y);
                Color col = new Color(rgba, true);
                col = new Color(255 - col.getRed(),
                        255 - col.getGreen(),
                        255 - col.getBlue());
                BI.setRGB(x, y, col.getRGB());
            }
        }
        return BI;
    }
}
```

➢ ImageClass:

```
package ImagePackage;
```

```
import DebuggingPackage.DebuggingClass1;
import javax.swing.*;
import javax.imageio.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.util.*;
import MainPackage.MainClass;
//import java.util.logging.Level;
//import java.util.logging.Logger;

public class ImageClass {
   static JFrame imageFrame;
   static JPanel imagePanel;
   static Image loadedImage = null;
   static JLabel toolsBoxLabel, filtersBoxLabel;
   static JComboBox toolsBox, filtersBox;
   static JButton imageCropBtn, filtersApplyBtn, saveBtn, backBtn, exitBtn,
undoBtn, redoBtn;
   static ActionListenerClass listener;
   static Stack undoStack, redoStack;
   static boolean editFlag = false, imageCropped = false;
   //    static MouseListenerClass mouseListener;
   static ToolsClass drawTool;
   static Dimension screenSize;
   static int screenHeight, screenWidth;
```

```
    public ImageClass(String filepath)
    {
        SwingUtilities.invokeLater(() -> {
            try {
                undoStack = new Stack();
                redoStack = new Stack();
                imageFrame = new JFrame("Image Editor");
                imageFrame.pack();
                imageFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
//              imageFrame.setIconImage(ImageIO.read(new
File("G:\\Jobs\\Projects\\Image Editing Program\\ImageEditorIcon.png")));
                imageFrame.setBackground(Color.BLACK);
                screenSize = Toolkit.getDefaultToolkit().getScreenSize();
                screenHeight = (int) screenSize.getHeight();
                screenWidth = (int) screenSize.getWidth();
                int w = 5*screenWidth/100;
                int h = 5*screenHeight/100;
                imageFrame.setLocation(w, h);
                imageFrame.setLayout(null);
                imageFrame.setResizable(false);
                imageFrame.setVisible(true);
                imageFrame.setSize(900, 600);

                imagePanel = new JPanel();
                imagePanel.setBounds(0, 0, 500, 500);
                imageFrame.add(imagePanel);

                loadedImage = ImageIO.read(new File(filepath));
                loadedImage = loadedImage.getScaledInstance(500, 500,
Image.SCALE_DEFAULT);
                drawTool = new ToolsClass(loadedImage);
                imagePanel.add(drawTool);



                toolsBoxLabel = new JLabel("Tool: ");
                toolsBoxLabel.setBounds(550, 10, 150, 30);
                imageFrame.add(toolsBoxLabel);

                String[] tools = {"None", "Rectangle", "Circle"};
                toolsBox = new JComboBox(tools);
                toolsBox.setSelectedIndex(0);
                toolsBox.setBounds(590, 10, 150, 30);
                imageFrame.add(toolsBox);
                listener = new ActionListenerClass();
                toolsBox.addActionListener(listener);

                imageCropBtn = new JButton("Crop Image");
                imageCropBtn.setBounds(750, 10, 110, 30);
                imageFrame.add(imageCropBtn);
                imageCropBtn.addActionListener(listener);

                filtersBoxLabel = new JLabel("Filter: ");
                filtersBoxLabel.setBounds(550, 100, 150, 30);
                imageFrame.add(filtersBoxLabel);

                String[] filters = {"None", "Light", "Dark", "Blur", "Invert"};
```

```
filtersBox = new JComboBox(filters);
filtersBox.setSelectedIndex(0);
filtersBox.setBounds(590, 100, 150, 30);
imageFrame.add(filtersBox);

filtersApplyBtn = new JButton("Apply Filter");
filtersApplyBtn.setBounds(750, 100, 110, 30);
imageFrame.add(filtersApplyBtn);
filtersApplyBtn.addActionListener(listener);

saveBtn = new JButton("Save");
saveBtn.setBounds(750, 280, 90, 30);
imageFrame.add(saveBtn);
saveBtn.addActionListener(listener);
drawTool.setImage(loadedImage);
undoBtn = new JButton("Undo");
undoBtn.setBounds(600, 200, 90, 30);
imageFrame.add(undoBtn);
undoBtn.addActionListener(listener);

redoBtn = new JButton("Redo");
redoBtn.setBounds(750, 200, 90, 30);
imageFrame.add(redoBtn);
redoBtn.addActionListener(listener);

backBtn = new JButton("Back");
backBtn.setBounds(600, 500, 90, 30);
```

```
            imageFrame.add(backBtn);
            backBtn.addActionListener(listener);

            exitBtn = new JButton("Exit");
            exitBtn.setBounds(750, 500, 90, 30);
            imageFrame.add(exitBtn);
            exitBtn.addActionListener(listener);

          }
          catch(HeadlessException | IOException ex){
            DebuggingClass1 err =  new DebuggingClass1();
            err.logException(ex.toString()); //Store exception in error ArrayList
          }
        });

    }
    public static BufferedImage toBufferedImage(Image img)
    {
        // Create a buffered image with transparency
        BufferedImage bimage = new BufferedImage(img.getWidth(null),
img.getHeight(null), BufferedImage.TYPE_3BYTE_BGR);

        // Draw the image on to the buffered image
        Graphics2D bGr = bimage.createGraphics();
        bGr.drawImage(img, 0, 0, null);
        bGr.dispose();

        // Return the buffered image
    return bimage;
}

static class ActionListenerClass implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if (actionEvent.getSource() == toolsBox)
        {
            try
            {
                if(toolsBox.getSelectedIndex() != 0)
                    drawTool.setToolType(toolsBox.getSelectedIndex());
            }
            catch(Exception ex)
            {
                DebuggingClass1 err =  new DebuggingClass1();
                err.logException(ex.toString()); //Store exception in error ArrayList
            }
        }
        if (actionEvent.getSource() == imageCropBtn)
        {
            try
            {
                if(toolsBox.getSelectedIndex() != 0)
                {
                    undoStack.push(drawTool.getImage());
```

```
drawTool.setImage(drawTool.cropImage(toBufferedImage(drawTool.getImage(
))));
                        toolsBox.setSelectedIndex(0);
                        editFlag = true;
                    }
                    else
                        JOptionPane.showMessageDialog(imageFrame, "Please Choose
a tool to cut with");
                }
                catch(HeadlessException | RasterFormatException ex)
                {
                    DebuggingClass1 err =  new DebuggingClass1();
                    err.logException(ex.toString()); //Store exception in error ArrayList
                }
            }
            else if (actionEvent.getSource() == filtersApplyBtn)
            {
                FiltersClass filterTool = new FiltersClass();
                try
                {


                    switch (filtersBox.getSelectedIndex()) {
                        case 1:
                            undoStack.push(drawTool.getImage());
                            loadedImage =
filterTool.lightenImage(toBufferedImage(drawTool.getImage()));
//                          loadedImage = loadedImage.getScaledInstance(500, 500,
Image.SCALE_DEFAULT);
```

```
                drawTool.setImage(loadedImage);
                redoStack.clear();
                filtersBox.setSelectedIndex(0);
                editFlag = true;
                break;
            case 2:
                undoStack.push(drawTool.getImage());
                loadedImage =
filterTool.darkenImage(toBufferedImage(drawTool.getImage()));
//                    loadedImage = loadedImage.getScaledInstance(500, 500,
Image.SCALE_DEFAULT);
                drawTool.setImage(loadedImage);
                redoStack.clear();
                filtersBox.setSelectedIndex(0);
                editFlag = true;
                break;
            case 3:
                undoStack.push(drawTool.getImage());
                loadedImage =
filterTool.blurImage(toBufferedImage(drawTool.getImage()));
//                    loadedImage = loadedImage.getScaledInstance(500, 500,
Image.SCALE_DEFAULT);
                drawTool.setImage(loadedImage);
                redoStack.clear();
                filtersBox.setSelectedIndex(0);
                editFlag = true;
                break;
            case 4:
                undoStack.push(drawTool.getImage());
```

```
                        loadedImage =
filterTool.invertImage(toBufferedImage(drawTool.getImage()));
//                              loadedImage = loadedImage.getScaledInstance(500, 500,
Image.SCALE_DEFAULT);
                        drawTool.setImage(loadedImage);
                        redoStack.clear();
                        filtersBox.setSelectedIndex(0);
                        editFlag = true;
                        break;
                    default:
                        JOptionPane.showMessageDialog(imageFrame, "Please
Choose a filter to apply");
                        break;
                }

            }
            catch(HeadlessException ex)
            {
                DebuggingClass1 err =  new DebuggingClass1();
                err.logException(ex.toString()); //Store exception in error ArrayList
            }
        }
        else if (actionEvent.getSource() == saveBtn)
        {
            try
            {
                if(editFlag)
                {
```

```
                String filePath = null;
                JFileChooser fileChooser = new JFileChooser(filePath);
                imageFrame.setVisible(false);
                int choosenBtn = fileChooser.showSaveDialog(fileChooser);
                if(choosenBtn == JFileChooser.APPROVE_OPTION)
                {
                  File tempFile = new 
File(fileChooser.getSelectedFile().toString()+".png");
                  ImageIO.write(toBufferedImage(drawTool.getImage()), "png", 
tempFile);
                  imageFrame.setVisible(true);


                } else {
                  imageFrame.setVisible(true);
                }
              }
              else JOptionPane.showMessageDialog(imageFrame, "You can 
NOT do this right now!");
            }
            catch(HeadlessException | IOException ex)
            {
              DebuggingClass1 err =  new DebuggingClass1();
              err.logException(ex.toString()); //Store exception in error ArrayList
            }
          }
          else if (actionEvent.getSource() == undoBtn)
          {
            try
            {
                if(!undoStack.empty() && editFlag == true)
                {
                    redoStack.push(drawTool.getImage());
                    loadedImage = toBufferedImage((Image)undoStack.pop());
//                      loadedImage = loadedImage.getScaledInstance(700, 700,
Image.SCALE_DEFAULT);
                    drawTool.setImage(loadedImage);
                    drawTool.repaint();
                }
                else
                    JOptionPane.showMessageDialog(imageFrame, "You Can NOT
do this right now!");
            }
            catch(HeadlessException ex)
            {
                DebuggingClass1 err =  new DebuggingClass1();
                err.logException(ex.toString()); //Store exception in error ArrayList
            }
        }
        else if (actionEvent.getSource() == redoBtn)
        {
            try
            {
                if(!redoStack.empty() && editFlag == true)
                {
                    undoStack.push(drawTool.getImage());
                    loadedImage = toBufferedImage((Image)redoStack.pop());
//                    loadedImage = loadedImage.getScaledInstance(700, 700,
Image.SCALE_DEFAULT);
                    drawTool.setImage(loadedImage);
                    drawTool.repaint();
                }
                else
                    JOptionPane.showMessageDialog(imageFrame, "You Can NOT
do this right now!");
            }
            catch(HeadlessException ex)
            {
                DebuggingClass1 err =  new DebuggingClass1();
                err.logException(ex.toString()); //Store exception in error ArrayList
            }
        }
        else if (actionEvent.getSource() == backBtn)
        {
            try
            {
                if (JOptionPane.showConfirmDialog( imageFrame,"Are you
sure?","Query", JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION)
                {
                    imageFrame.setVisible(false);
                    MainClass.main(null); // To back to main frame
                }
            }
            catch(HeadlessException ex)
            {
                DebuggingClass1 err =  new DebuggingClass1();
                err.logException(ex.toString()); //Store exception in error ArrayList
            }
        }
        else if (actionEvent.getSource() == exitBtn)
        {
            try
            {
                if (JOptionPane.showConfirmDialog( imageFrame,"Are you
sure?","Query", JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION)
                {
                    System.exit(0);
                }
            }
            catch(HeadlessException ex)
            {
                DebuggingClass1 err =  new DebuggingClass1();
                err.logException(ex.toString()); //Store exception in error ArrayList
            }
        }
    }
  }
}
```

➢ ToolsClass:

```
package ImagePackage;

import DebuggingPackage.DebuggingClass1;
//import static ImagePackage.ImageClass.toBufferedImage;
import java.awt.Dimension;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Ellipse2D;
import java.awt.image.BufferedImage;
//import java.io.File;
//import java.io.IOException;
//import javax.imageio.ImageIO;
import javax.swing.*;

public class ToolsClass extends JPanel{

   private int x1, y1, x2, y2, toolType;
   private Image loadedImage;
   private MouseListenerClass listener;
   public ToolsClass(Image image){
      try {
         loadedImage = image.getScaledInstance(500, 500,
Image.SCALE_DEFAULT);
         listener = new MouseListenerClass();
         super.addMouseListener(listener);
```

```
        super.addMouseMotionListener(listener);
    } catch (Exception ex) {
        DebuggingClass1 err =  new DebuggingClass1();
        err.logException(ex.toString()); //Store exception in error ArrayList
    }
}
public void setToolType(int toolType) {
    this.toolType = toolType;
}

public void setStartPoint(int x, int y) {
    x1 = x;
    y1 = y;
}

public void setEndPoint(int x, int y) {
    x2 = (x);
    y2 = (y);
}

public int[] getCirclePoints()
{
    int pw = Math.abs(x1-x2);
    int ph = Math.abs(y1-y2);
    return new int[]{x1, y1, x2, y2, pw, ph};
}
```

```
    public void drawPerfectRect(Graphics g, int x, int y, int x2, int y2) {
        int px = Math.min(x,x2);
        int py = Math.min(y,y2);
        int pw=Math.abs(x-x2);
        int ph=Math.abs(y-y2);
        g.drawRect(px, py, pw, ph);
    }

    public void drawPerfectOval(Graphics g, int x, int y, int x2, int y2) {
        int px = Math.min(x,x2);
        int py = Math.min(y,y2);
        int pw=Math.abs(x-x2);
        int ph=Math.abs(y-y2);
        g.drawOval(px, py, pw, ph);
    }

    public Image getImage()
    {
        return loadedImage;
    }

    public void setImage(Image img)
    {
        loadedImage = img;
        toolType = 0;
//          loadImage(img);
        repaint();
    }

    public BufferedImage cropImage(BufferedImage bufferedImage) {
        if(toolType == 2)
        {
            int px = Math.min(x1,x2);
            int py = Math.min(y1,y2);
            int pw = Math.abs(x1-x2);
            int ph = Math.abs(y1-y2);
            BufferedImage circleBuffer = new BufferedImage(500, 500,
BufferedImage.TYPE_3BYTE_BGR);
            Graphics2D g2 = circleBuffer.createGraphics();
            g2.setClip(new Ellipse2D.Float(px, py, pw, ph));
            g2.drawImage(loadedImage, 0, 0, 500, 500, this);
            return circleBuffer.getSubimage(px, py, pw, ph);
        }
        else if(toolType == 1)
        {
            int px = Math.min(x1,x2);
            int py = Math.min(y1,y2);
            int pw=Math.abs(x1-x2);
            int ph=Math.abs(y1-y2);
//            System.out.println(px+" "+py+" "+pw+" "+ph);
//            ImageIO.write(croppedImage,"jpg", new
File("C:\\Users\\egypt2\\Desktop\\test.jpg"));
            return bufferedImage.getSubimage(px, py, pw, ph);
        }
        return null;
    }

//        public BufferedImage loadImage(Image img)
//        {
//            BufferedImage imageBuffer = new BufferedImage(500, 500,
BufferedImage.TYPE_3BYTE_BGR);
//            Graphics2D g2d = imageBuffer.createGraphics();
//            if (loadedImage != null) {
//                g2    d.drawImage(loadedImage, 0, 0, this);
//            }
//            return imageBuffer;
//        }

    @Override
    public Dimension getPreferredSize() {
        return loadedImage == null ? new Dimension(500, 500) : new
Dimension(500, 500);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g.create();
        if (loadedImage != null) {
            g2d.drawImage(loadedImage, 0, 0, this);
        }
        g2d.setColor(Color.BLACK);
        g2d.setStroke(new BasicStroke(3, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_BEVEL, 0, new float[]{9}, 0));
        if(toolType == 1)
        {
            drawPerfectRect(g2d, x1, y1, x2, y2);
        }
        else if (toolType == 2)
        {
            drawPerfectOval(g2d, x1, y1, x2, y2);
        }
        g2d.dispose();
    }

    // Mouse listener class for painting
    private class MouseListenerClass extends MouseAdapter {

        public MouseListenerClass() {
        }

        @Override
        public void mousePressed(MouseEvent e) {
            setStartPoint(e.getX(), e.getY());
        }

        @Override
        public void mouseDragged(MouseEvent e) {
            setEndPoint(e.getX(), e.getY());
            repaint();
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            setEndPoint(e.getX(), e.getY());
            repaint();
        }
    }
}
```

2. DebuggingPackage:

➢ DebuggingClass:

```
package DebuggingPackage;

import java.awt.*;
import java.io.File;
import java.io.IOException;
import java.util.*;
import javax.imageio.ImageIO;
import javax.swing.*;

public class DebuggingClass1 {
   private static final ArrayList<String> errorsList = new ArrayList<String>() ;

   public void DebuggingClass1()
   {


   }
   public void logException(String err)
   {
      errorsList.add(err);
   }
   JFrame exceptionFrame;
   JPanel exceptionPanel;
   JTextArea exceptionTextArea;
   public void openFrame() throws IOException
```

```
    {
        try
        {
            exceptionFrame = new JFrame("Debugging Frame");
            exceptionFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            exceptionPanel = new JPanel();
            exceptionFrame.setIconImage(ImageIO.read(new
File("C:\\Users\\amjad\\Desktop\\6th Semester\\Advance
JAVA\\projects\\download.jpeg")));
            exceptionFrame.add(exceptionTextArea, BorderLayout.WEST);
            exceptionTextArea = new JTextArea(30,72);
            exceptionTextArea.setEditable(false);
            exceptionPanel.add(exceptionTextArea, BorderLayout.PAGE_START);
            String errorText = "";
            for(int i=0;i<errorsList.size();i++)
            {
                errorText += (i+1)+". "+errorsList.get(i)+"\n";
            }
            if(!errorText.equals(""))
            {
                exceptionTextArea.setText(errorText);
                exceptionFrame.pack();
                exceptionFrame.setLocationRelativeTo(null);
                exceptionFrame.setVisible(true);
                exceptionFrame.setResizable(false);
                exceptionFrame.setSize(600, 600);
            }
            else
            JOptionPane.showMessageDialog(exceptionFrame, "No Errors
Yet!");
      }
      catch(NullPointerException ex)
      {
         JOptionPane.showMessageDialog(exceptionFrame, "No Errors Yet!");
      }
      catch(HeadlessException ex)
      {
         logException(ex.toString()); //Store exception in error ArrayList
      }
   }
}
```

3. MainPackage:

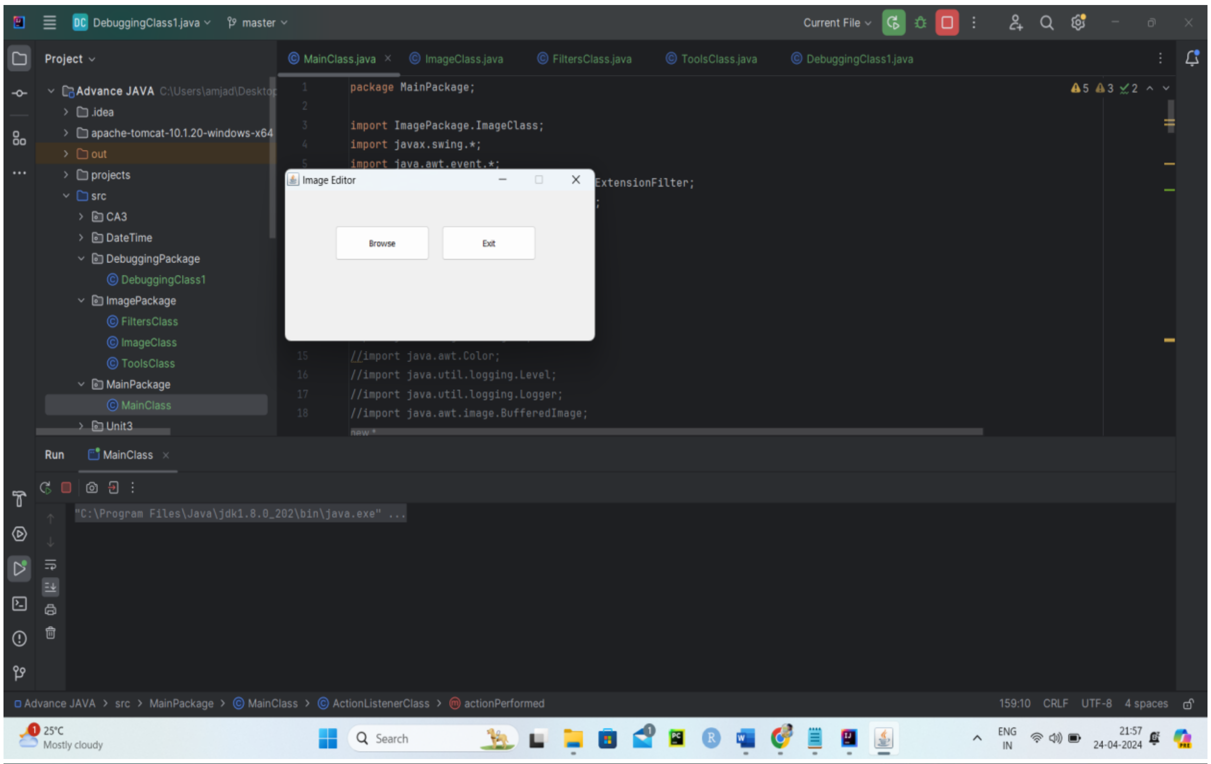
➢ MainClass:

```
package MainPackage;

import ImagePackage.ImageClass;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import DebuggingPackage.DebuggingClass1;

import java.awt.Dimension;
import java.awt.HeadlessException;
import java.awt.Toolkit;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
//import java.awt.Color;
//import java.util.logging.Level;
//import java.util.logging.Logger;
//import java.awt.image.BufferedImage;
public class MainClass {

    private static String getFileExtension(String fileName) {

        int i = fileName.lastIndexOf('.');
        if (i >= 0)
            return fileName.substring(i+1);
        return "";
    }
//    public MainClass() {
//        String[] args = null;
//        MainClass.main(args);
//    }
    /**
     * @param args the command line arguments
     */
    static JFrame mainFrame;
    static JButton browseBtn, debugBtn, exitBtn;
    static ActionListenerClass listener;
    static Dimension screenSize;
    static int screenHeight, screenWidth;
    public static void main(String[] args) {
        // TODO code application logic here
        try
        {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
// Sets the window style like windows operating system.
            //Constructing the main frame
            mainFrame = new JFrame("Image Editor");
            screenSize = Toolkit.getDefaultToolkit().getScreenSize();
            screenHeight = (int) screenSize.getHeight();
            screenWidth = (int) screenSize.getWidth();
            int w = 20*screenWidth/200;
            int h = 15*screenHeight/200;
            mainFrame.setLocation(w,h);
//          mainFrame.setContentPane(new JLabel(new
ImageIcon(ImageIO.read(new File("G:\\Jobs\\Projects\\Image Editing
Program\\Background.jpg")))));
            mainFrame.pack();
            mainFrame.setVisible(true);
            mainFrame.setResizable(false);
            mainFrame.setSize(500, 250);
//          mainFrame.setIconImage(ImageIO.read(new
File("G:\\Jobs\\Projects\\Image Editing Program\\ImageEditorIcon.png")));
            mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            //Creating a panel which contains buttons
            mainFrame.setLayout(null);

            listener = new ActionListenerClass();

            //Browse Button
            browseBtn = new JButton("Browse");
            browseBtn.setBounds(80, 50, 150, 50);
            mainFrame.add(browseBtn);
            browseBtn.addActionListener(listener);

            //Exit Button
            exitBtn = new JButton("Exit");
            exitBtn.setBounds(250, 50, 150, 50);
```

```
            mainFrame.add(exitBtn);
            exitBtn.addActionListener(listener);



            //Debug Button
            debugBtn = new JButton("Debug");
            debugBtn.setBounds(3, 3, 40, 20);
//            mainFrame.add(debugBtn);
            debugBtn.addActionListener(listener);
        }
        catch(HeadlessException | ClassNotFoundException |
IllegalAccessException | InstantiationException |
UnsupportedLookAndFeelException /*| IOException*/ ex){
            DebuggingClass1 err =  new DebuggingClass1();
            err.logException(ex.toString());
        }
    }
    public static boolean contains(String [] stringArray, String comparingString)
    {
        for (String stringInArray : stringArray)
            if (comparingString.equals(stringInArray))
                return true;
        return false;
    }
    static class ActionListenerClass implements ActionListener{

        @Override
        public void actionPerformed(ActionEvent actionEvent) {
            if (actionEvent.getSource() == browseBtn)
            {
               try
               {
                  String filePath = null;
                  JFileChooser fileChooser = new JFileChooser(filePath);
                  String[] arrayOfFormats = new String[]{"jpg", "png", "gif", "bmp",
"jpeg", "tiff", "JPG", "PNG", "GIF", "BMP", "JPEG", "TIFF"};
                  fileChooser.setFileFilter(new FileNameExtensionFilter("Images",
arrayOfFormats)); // Sets selectable formats to images only
                  mainFrame.setVisible(false);
                  int choosenBtn = fileChooser.showOpenDialog(fileChooser);
                  if(choosenBtn == JFileChooser.APPROVE_OPTION)
                  {
                     filePath = fileChooser.getSelectedFile().getAbsolutePath(); // Get
the path of selected file
                  } else {
                     mainFrame.setVisible(true);
                  }
                  if(contains(arrayOfFormats, getFileExtension(filePath)))
                     new ImageClass(filePath); // Invoke ImageClass to run with
filepath got by filechooser
                  else
                  {
                     JOptionPane.showMessageDialog(mainFrame, "This is not an
image, please choose a correct file!");
                     mainFrame.setVisible(true);
                  }
               }
            catch(HeadlessException | NullPointerException ex)
            {
                DebuggingClass1 err =  new DebuggingClass1();
                err.logException(ex.toString()); //Store exception in error ArrayList
            }
        }
        else if (actionEvent.getSource() == debugBtn)
        {
            try
            {
                DebuggingClass1 openDebugFrame = new DebuggingClass1();
                openDebugFrame.openFrame();
            }
            catch(IOException ex)
            {
                DebuggingClass1 err =  new DebuggingClass1();
                err.logException(ex.toString()); //Store exception in error ArrayList
            }
        }
        else if (actionEvent.getSource() == exitBtn)
        {
            try
            {
                if (JOptionPane.showConfirmDialog(mainFrame,"Are you
sure?","Query", JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION)
                {
                    System.exit(0);
```

```
                        }
                    }
                    catch(HeadlessException ex)
                    {
                        DebuggingClass1 err =  new DebuggingClass1();
                        err.logException(ex.toString()); //Store exception in error ArrayList
                    }
                }
            }
        }
    }
```

Result:

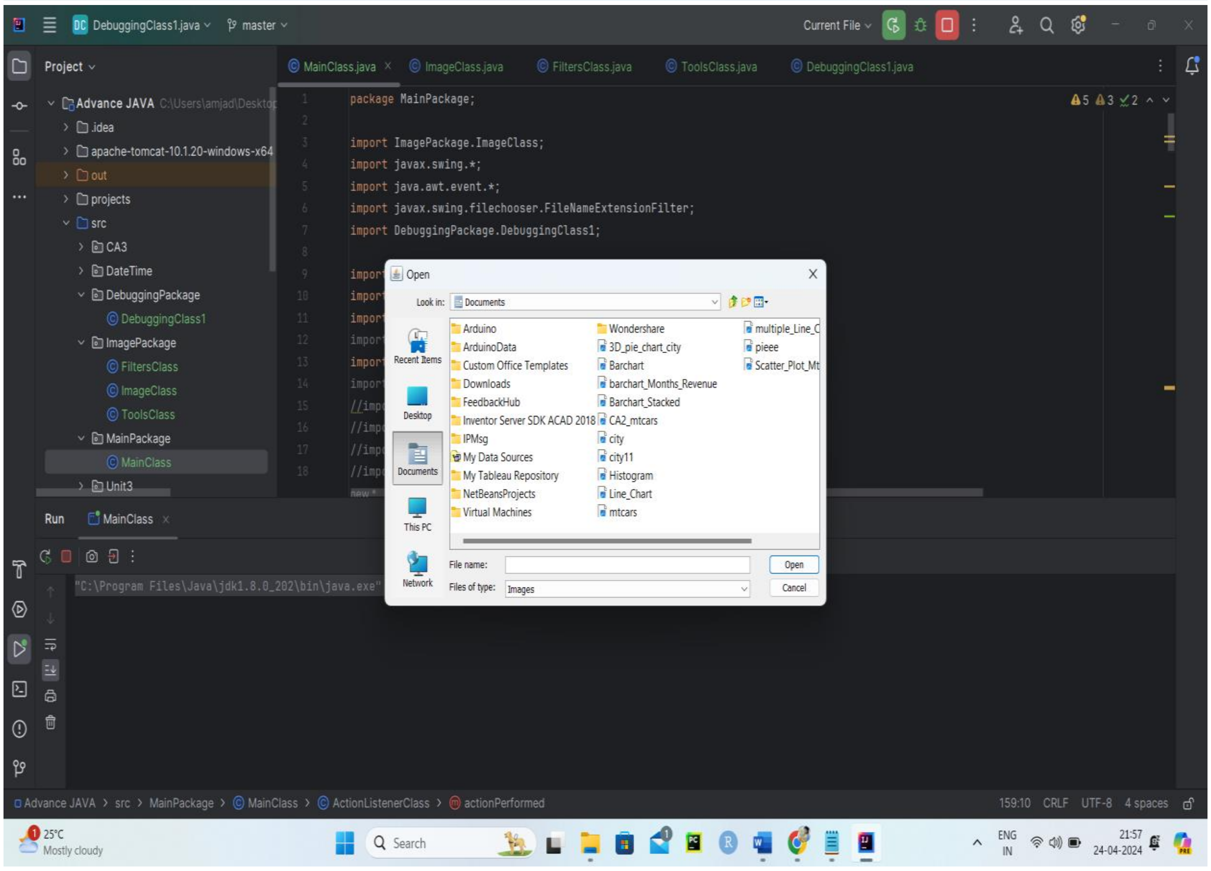


```
package MainPackage;

import ImagePackage.ImageClass;
import javax.swing.*;
import java.awt.event.*;
```

```
//import java.awt.Color;
//import java.util.logging.Level;
//import java.util.logging.Logger;
//import java.awt.image.BufferedImage;
```

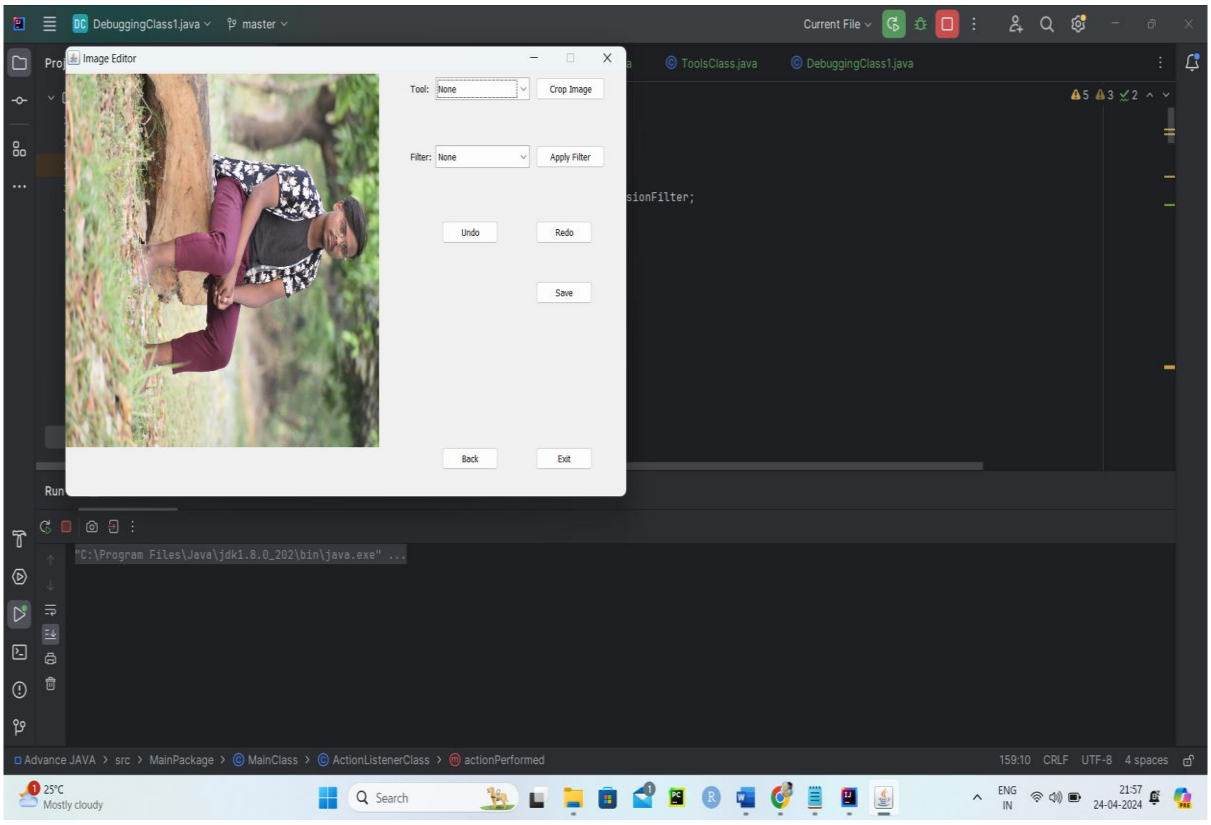
```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
```

Image Editor
Browse
Exit
Advance JAVA > src > MainPackage > MainClass > ActionListenerClass > actionPerformed
159:10 CRLF UTF-8 4 spaces



```
package MainPackage;

import ImagePackage.ImageClass;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import DebuggingPackage.DebuggingClass1;
```

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe"
```

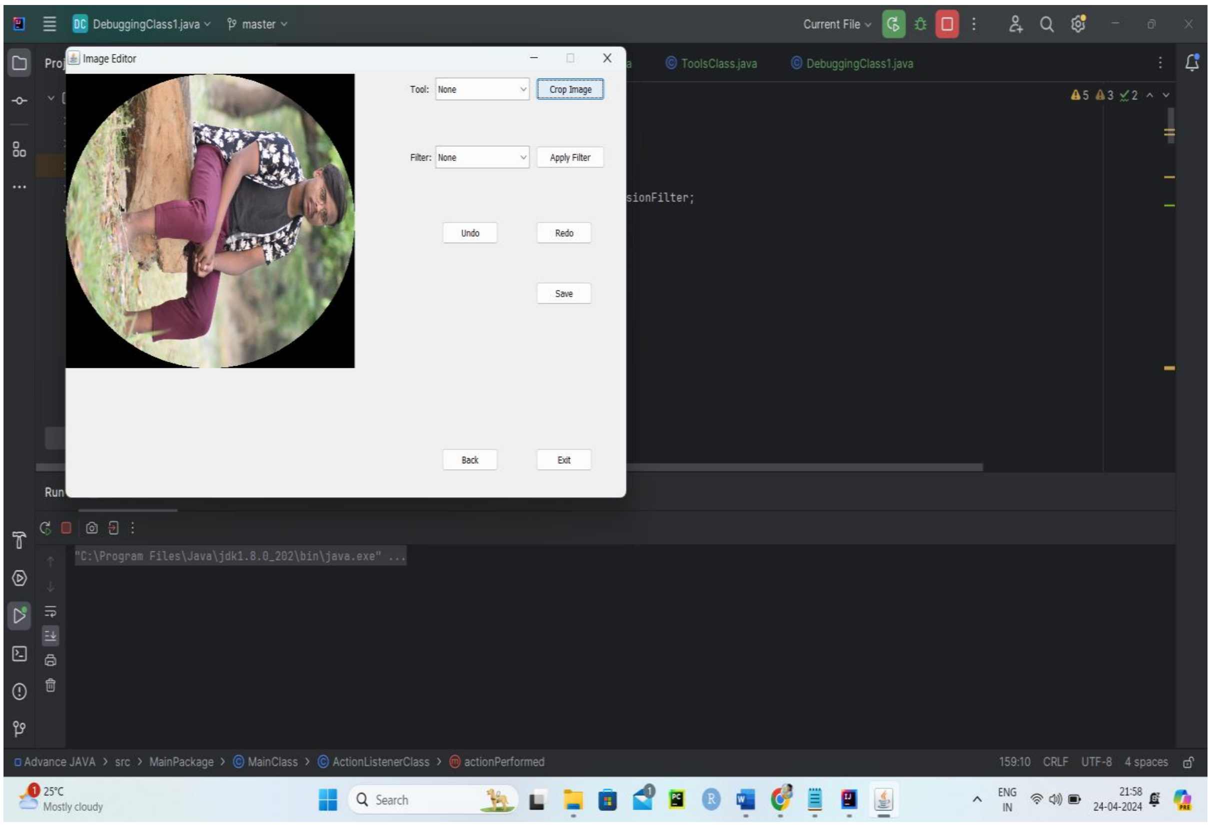


```
sionFilter;
```

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
```

Advance JAVA > src > MainPackage > MainClass > ActionListenerClass > actionPerformed   159:10 CRLF UTF-8 4 spaces

25°C Mostly cloudy   Search   ENG IN   21:57 24-04-2024

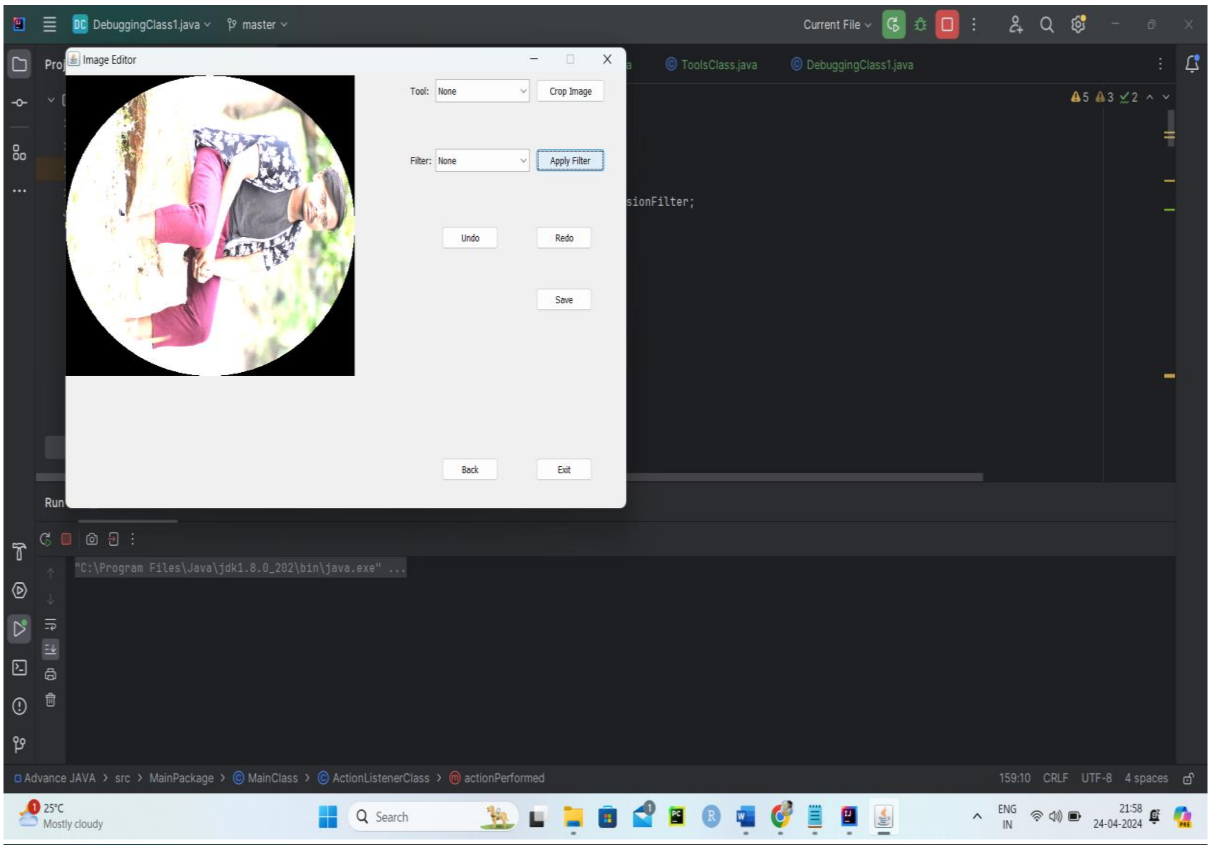


```
sionFilter;
```

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
```

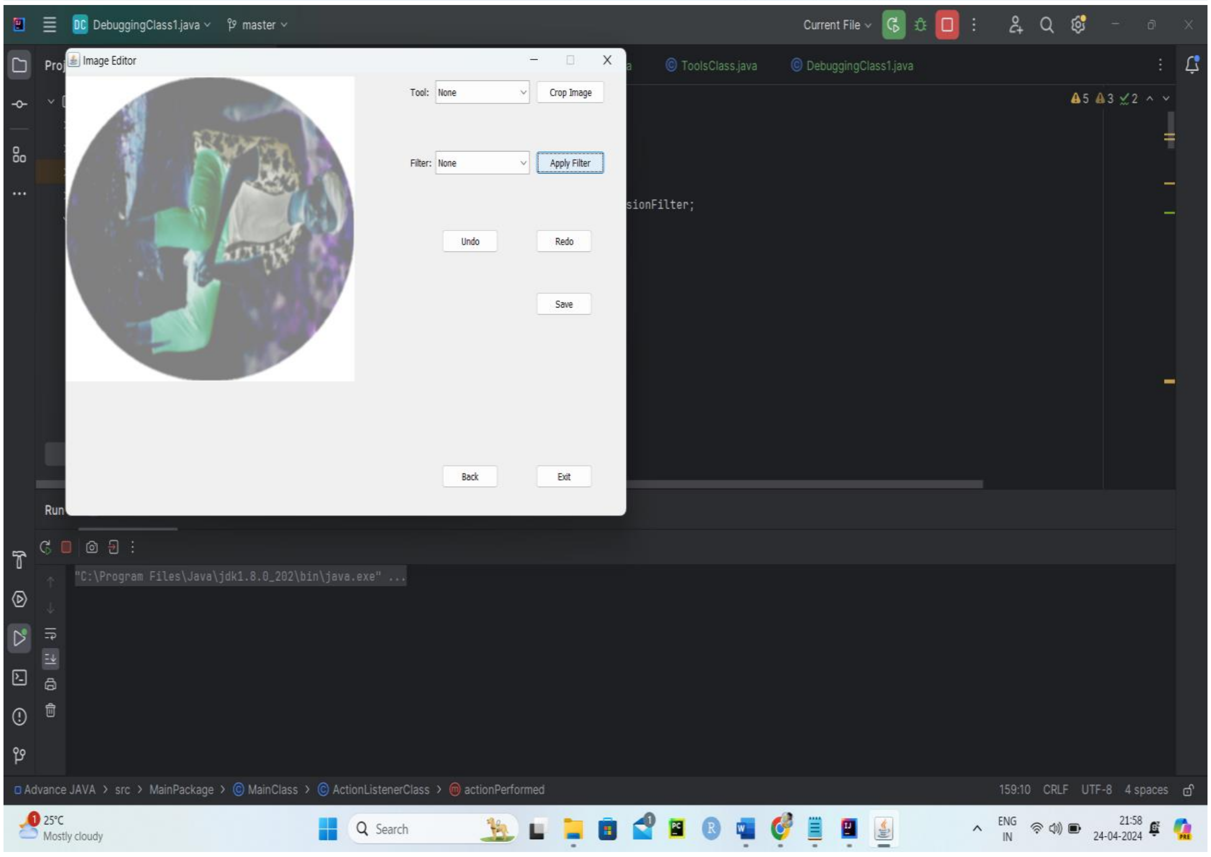
Advance JAVA > src > MainPackage > MainClass > ActionListenerClass > actionPerformed   159:10 CRLF UTF-8 4 spaces

25°C Mostly cloudy   Search   ENG IN   21:58 24-04-2024



```
sionFilter;
```

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
```



```
sionFilter;
```

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
```

Thank You