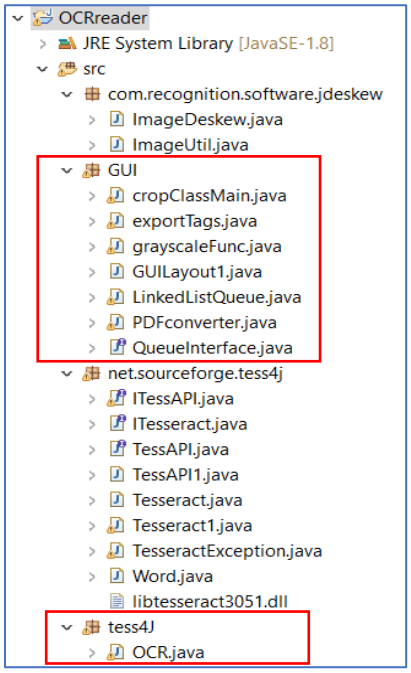
Candidate #: hpg293

Criterion C – Development

Classes and Interfaces*



*Only packages and classes within the red boxes were designed by myself.

Important Techniques and Skills

1. Advanced external libraries and methods
    a. Tesseract Library[1]
    b. Apache POI[2]
    c. Apache PDFBox[3]
2. Fully implemented queue
3. Multi-dimensional array
4. Encapsulation
5. Inheritance
6. Merging of sorted/stored data
7. File reading and outputting
8. Iteration
9. Advanced loops (i.e. nested for-each loop)

[1] (Apache POI - the Java API for Microsoft Documents, 2020)

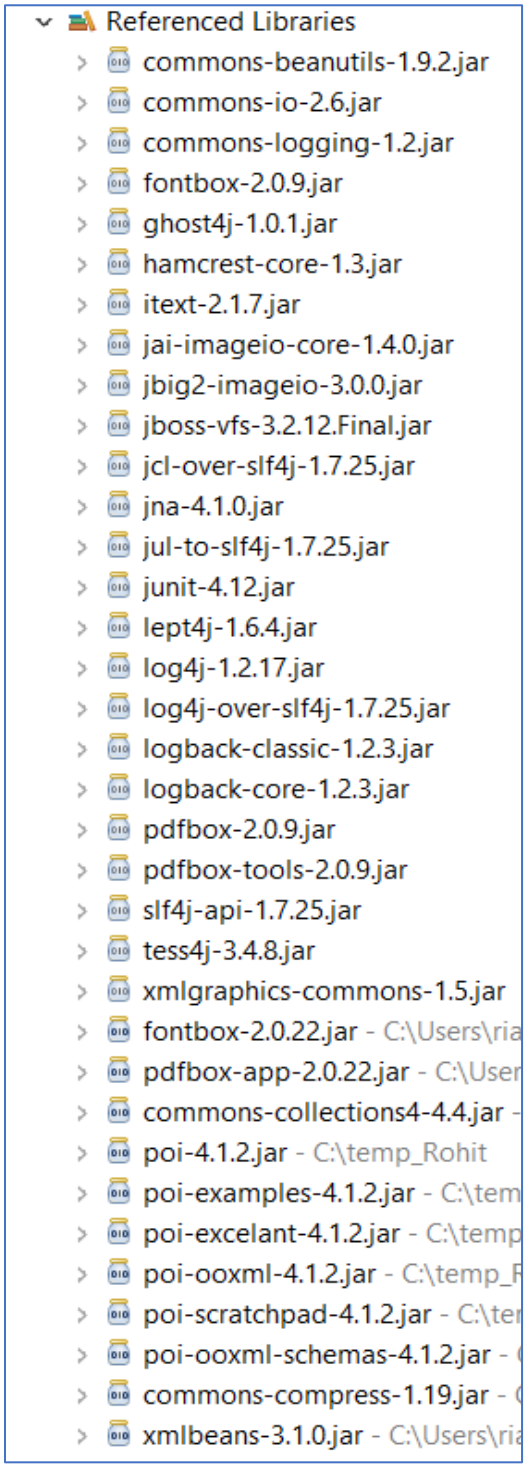
[2] (GitHub Releases · tesseract-ocr/tesseract, 2020)

[3] (Apache PDFBox | Download, 2020)

Candidate #: hpg293

External Libraries

All Imported JAR Libraries



PDFConverter class (utilizes PDFBox)

Sample packages:

```
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.encryption.InvalidPasswordException;
import org.apache.pdfbox.rendering.ImageType;
import org.apache.pdfbox.rendering.PDFRenderer;
```

Primary method utilizing PDFBox methods (code and explanation):

```
public void generateImageFromPDF(String convert) throws InvalidPasswordException, IOException {
    PDDocument doc = PDDocument.load(new File(convert));
    PDFRenderer pdfRender = new PDFRenderer(doc);
    pageCount = doc.getNumberOfPages();
    for (int page = 0; page < pageCount; ++page) {
        img = pdfRender.renderImageWithDPI(page, 300, ImageType.RGB);
        storageList.add(img);
    }
    doc.close();
}
```

Within the PDFConverter Class, the *doc* object contains the PDF file loaded in using the parameter *convert* from the main *GUILayout1* class. The *pdfRender* object can be used to render the PDF pages to a bufferedImage. The for-loop efficiently converts each page to an image with 300 dots per inch (good resolution)[4] and stores each converted page in an arrayList called *storageList*.[5]

[4] (All About Digital Photos - What is DPI?, 2021)

[5] (How to convert PDF to image in Java, 2021)

Candidate #: hpg293

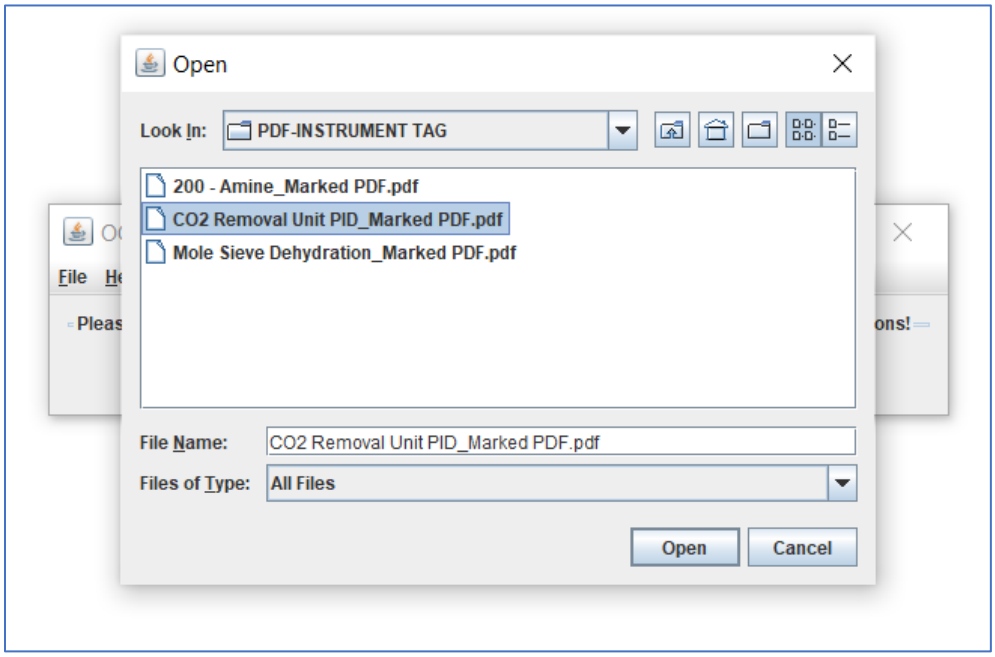
Methods (code and explanation):

```
protected int listSize() {
    return storageList.size();
}

protected void clearList() {
    storageList.clear();
}

protected BufferedImage generateImage() {
    return storageList.get(0);
}

protected void removeModImage(int nextOrPrev, BufferedImage backupImage) {
    storageList.remove(nextOrPrev);
    storageList.add(nextOrPrev, backupImage);
}

protected BufferedImage nextOrPrev(int nextOrPrev) {
    return storageList.get(nextOrPrev);
}
```

These methods are used by the main class to:

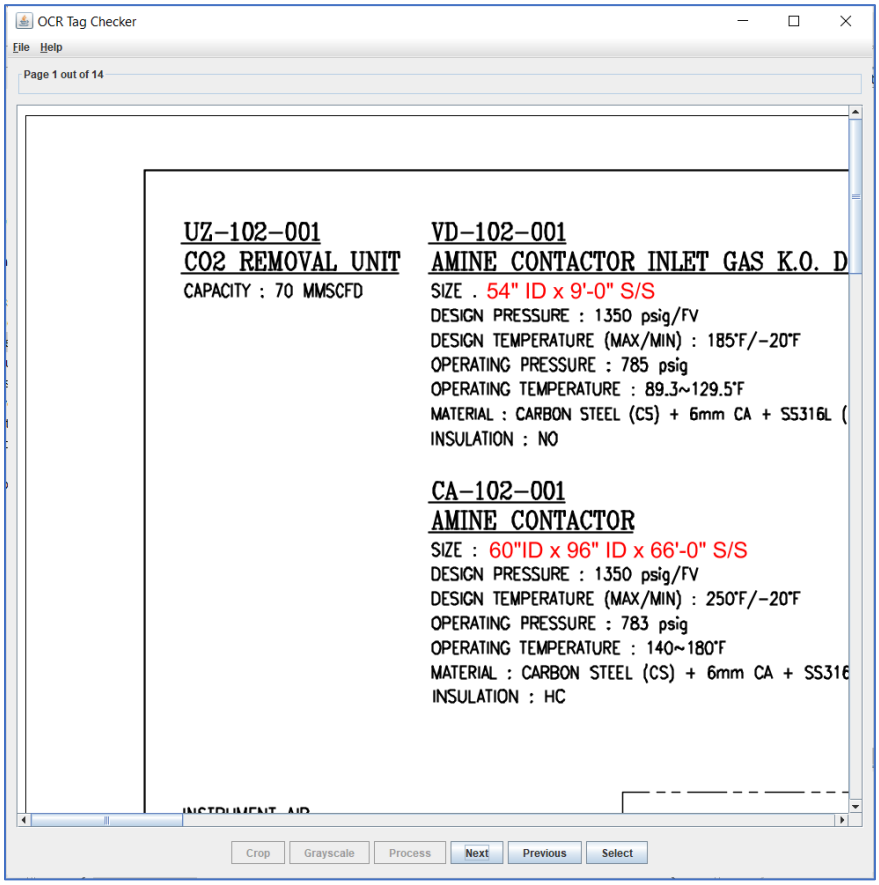
1. Allow the user to cycle through pages (through main class)
2. Reset the image (through main class)
3. Determine if there are multiple pages in the PDF

Selecting a PDF (GUI):



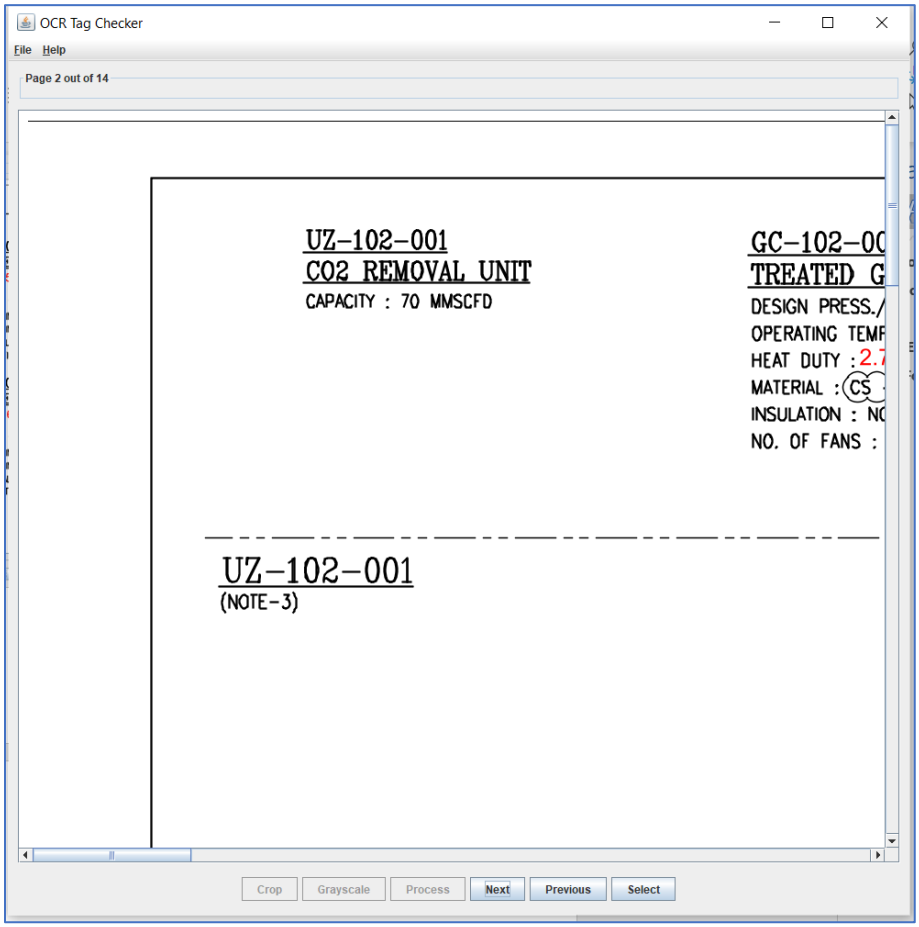
Candidate #: hpg293

Using the *next, previous,* and *select* buttons (GUI):



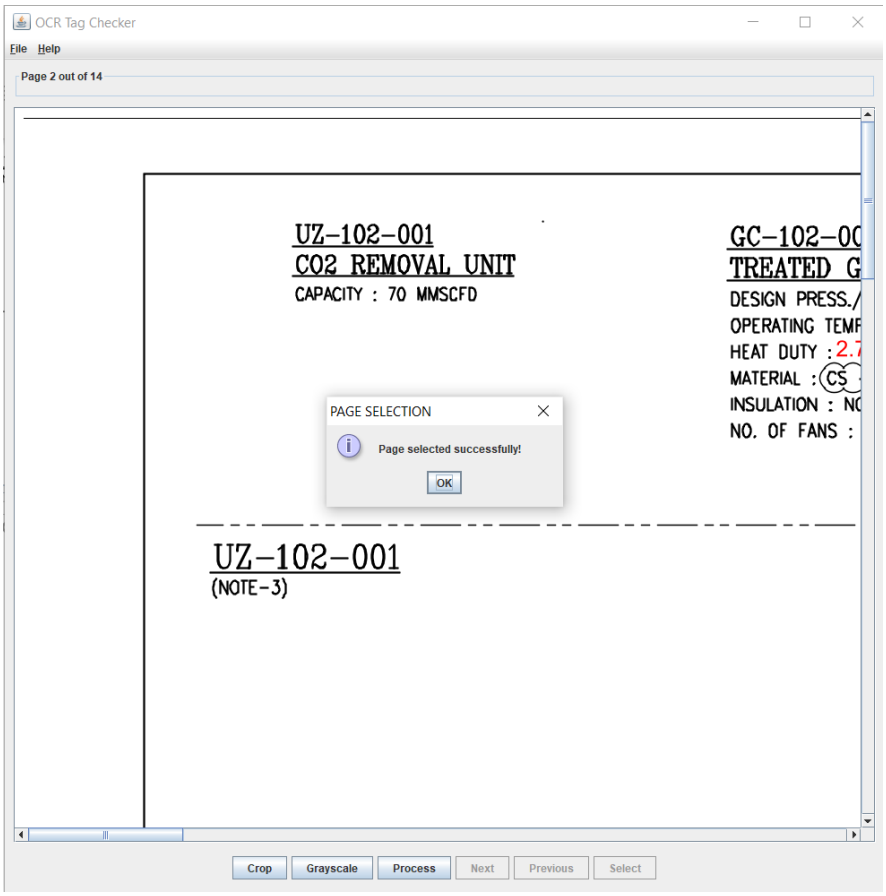
Candidate #: hpg293

After pressing the next button (GUI):



Candidate #: hpg293

After selecting a page (GUI and explanation):



After the desired page has been selected from the arrayList, the page may now be modified for cropping, grayscaling, and output processing. The selection options are grayed out until the file is **reset** or **cleared** to handle potential errors from changing the file during processing.

Candidate #: hpg293

exportTags (utilizes Apache POI)

Sample packages:

```
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.encryption.InvalidPasswordException;
import org.apache.pdfbox.rendering.ImageType;
import org.apache.pdfbox.rendering.PDFRenderer;
```

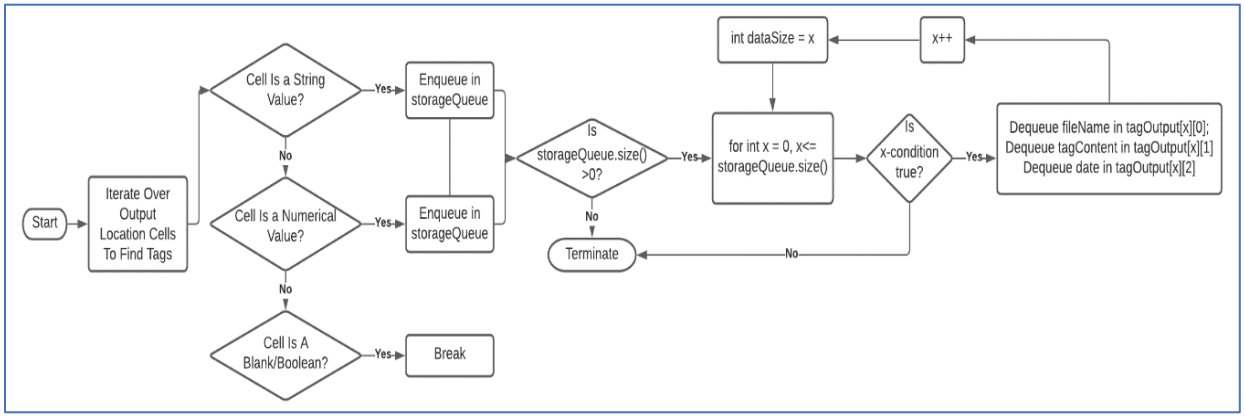
toExcel method objects (code):

```
public void tooExcel(LinkedListQueue queue, String tagFile, String fileLocation) throws IOException {
    try {
        int dataSize = 0;
        LinkedListQueue storageQueue = new LinkedListQueue(); //new queue

        FileInputStream checkValues = new FileInputStream(fileLocation); //object to read and input file contents

        XSSFWorkbook workbook = new XSSFWorkbook(); //new workbook for output
        XSSFSheet sheet = workbook.createSheet("tagOutput"); //new sheet for output

        XSSFWorkbook input = new XSSFWorkbook(fileLocation); //new workbook for input
        XSSFSheet inputSheet = input.getSheetAt(0); //new sheet for input

        LocalDateTime time = LocalDateTime.now(); //time for output
        DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy");

        Object[][] tagData = new Object[1000][3]; //2D array for output
        int rowNum, columnNum;
```

Iterative input (code):

```
int rowNum, columnNum;
Iterator<Row> rowCheck = inputSheet.iterator();
while (rowCheck.hasNext()) { //iterate through each row
    Row row = rowCheck.next();
    Iterator<Cell> cellIterator = row.cellIterator(); //iterate through each cell
    while (cellIterator.hasNext()) {
        Cell cell = cellIterator.next();
        switch (cell.getCellType()) {
        case NUMERIC:
            storageQueue.enqueue(cell.getNumericCellValue());
            break;
        case STRING:
            storageQueue.enqueue(cell.getStringCellValue());
            break;
        case BLANK:
            break;
        default:
            break;
        }

    }
}
checkValues.close();
input.close();

if (storageQueue.queueSize() > 0) {

    for (int x = 0; x <= (storageQueue.queueSize()); x++) {
        tagData[x][0] = storageQueue.dequeue();
        tagData[x][1] = storageQueue.dequeue();
        tagData[x][2] = storageQueue.dequeue();
        dataSize = x;
    }
}
```

Iterative input (flowchart and explanation):



The iteration algorithm allows the user to only use one Excel output file by re-writing the file contents instead of overwriting them. As demonstrated above, each cell is iterated over, its contents are drawn out, and a switch statement decides if the contents are placed in the new queue (*storageQueue*) based on certain criteria. If the file is empty, nothing is queued in *storageQueue* and the next steps are skipped to save time. If there are contents, they are moved from *storageQueue* to a **2D array** (*tagOutput[][]*). The 2D array is excellent for Excel file formatting thanks to the matrix similarity of the structures[6]. The

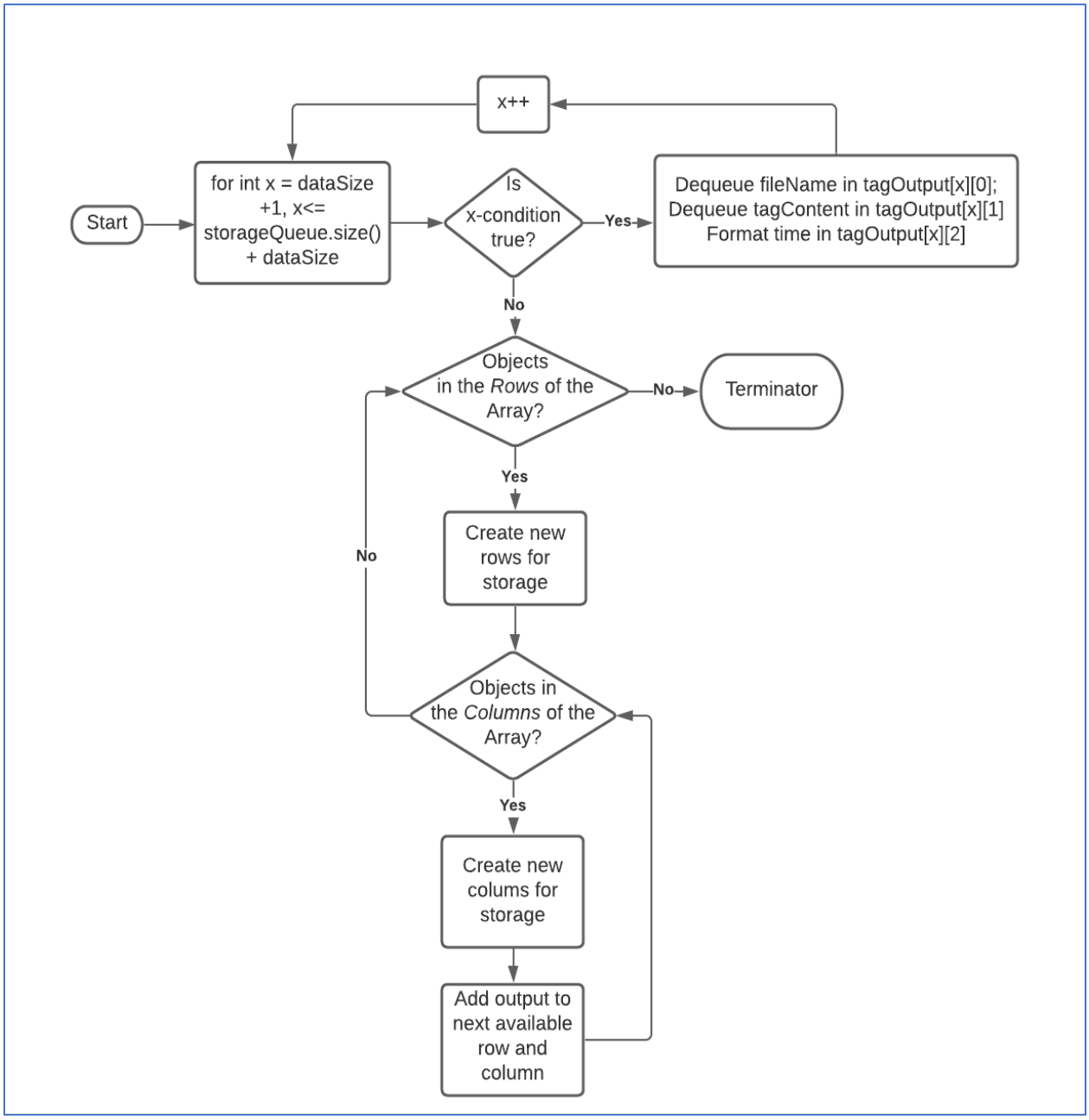
[6] (Excel VBA Programming - Multidimensional Arrays, 2021)

dataSize integer is important for appending the new tags in the *Excel output*. The merging of these two organised data structures was a key goal for usability.

Excel output (code):

```
for (int x = dataSize + 1; x <= ((queue.queueSize())) + dataSize; x++) { // all tags dequeued into the 2D
                                                                         // array
    tagData[x][0] = queue.dequeue();
    tagData[x][1] = queue.dequeue();
    tagData[x][2] = time.format(format);
}

rowNum = 0;

for (Object[] tag : tagData) { // nested for-each loop
    Row row = sheet.createRow(++rowNum); // creates required rows for sheet
    columnNum = 0;

    for (Object field : tag) {
        Cell cell = row.createCell(++columnNum); //creates required columns for sheets
        if (field instanceof String) { //if String, fill sheet with String
            cell.setCellValue((String) field);
        } else if (field instanceof Integer) { //if int, fill cell with int
            cell.setCellValue((Integer) field);
        }
    }

}

try (FileOutputStream outputStream = new FileOutputStream(fileLocation)) {
    workbook.write(outputStream); //output to selected file
}
workbook.close();
```

Excel output (flowchart and explanation):



The first for-loop above employs the *dataSize* integer to start adding new tags to the 2D array *tagOutput[][]* at the next available spot (assuming there were tags in the previous document. Once the x-condition is satisfied, the nested for-each loop creates new rows and columns that align with *tagOutput[][]* and allow for each cell to be properly lined up for efficient and organised output. The queue is beneficial as its fundamental **FIFO** implementation[7] means all the tags can be effectively organised by time and date.

[7] (Cook, 2005)

File (**with stored tags**) before new output (Excel Output):

	A	B	C	D
1				
2		TCB-140-PR-PI-0004 Rev 0B.pdf	TCB-140-PR-PI-0003	2021-01-28 17:47
3				
4				

Same file after output (Excel Output):

	A	B	C	D
1				
2		TCB-140-PR-PI-0004 Rev 0B.pdf	TCB-140-PR-PI-0003	2021-01-28 17:47
3		Mole Sieve Dehydration_Marked PDF.pdf	CX-IOS-ODIA	2021-01-28 17:50
4				

OCR (utilizes Tesseract Library)

Primary method (code and explanation):

```
public class OCR { // OCR Scanner
    private String text;
    Tesseract tesseract = new Tesseract();

    protected String fileName(BufferedImage large) {
        {
            try {
                tesseract.setDatapath("C:\\temp_Rohit\\ia\\Tess4J\\tessdata"); // sets path

                text = tesseract.doOCR(large); // scans the image

                System.out.println(text);

            } catch (TesseractException e) { // catches custom errors related to the Tesseract engine
                e.printStackTrace();
            }
        }
        return text;
    }
}
```

The *GUILayout1* will call on this method once the user presses the process button, at which point the OCR algorithm chops the text into individual characters and uses distinct changes in the shape of words to identify letters and output them as text[8]. All of these operations are run by the Tesseract OCR engine.

[8] (ICDAR, 2007)

Candidate #: hpg293

Queue Interface and Implementation Class

Interface (code):

```
package GUI;

public interface QueueInterface {

    boolean isEmpty(); //determines if queue is empty

    void enqueue (Object obj); //adds object to the end of the queue

    Object dequeue(); //removes and returns first object in the queue

    Object peekFront(); //returns first object in queue without removing it

    Object peekEnd(); //returns last object in queue without removing it

    int queueSize(); //returns size of queue

}
```

Implementation class (code and explanation):

```
public class LinkedListQueue implements QueueInterface {

    public LinkedListQueue() {
        lst = new LinkedList();
    }

    public boolean isEmpty() {
        return lst.isEmpty();
    }

    public void enqueue(Object obj) {
        lst.addLast(obj);
    }

    public Object dequeue() {
        if (queueSize() > 0) {
            return lst.removeFirst();
        } else {
            throw new NullPointerException();
        }
    }

    public Object peekFront() {
        if (queueSize() > 0) {
            return lst.getFirst();
        } else {
            throw new NullPointerException();
        }
    }

    public Object peekEnd() {
        if (queueSize() > 0) {
            return lst.getLast();
        } else {
            throw new NullPointerException();
        }
    }

    public int queueSize() {
        return lst.size();
    }

    private LinkedList lst;


}
```

The queue is primarily used to store the output when the user presses *Process* in the *GUILayout1* class and for storing the contents of the export Excel location as previously seen. The FIFO principle of the queue made it beneficial for organisation, as it permitted the first tag in the queue to be outputted first. Many useful methods were implemented and errors are thrown so they can be handled in other classes. Some methods have been implemented to assist in future development (ex. *peekEnd()* method).

grayscaleFunc Class

Constructor, *getHeight()*, and *getWidth()* methods (code and explanation)

```
public class grayscaleFunc {
    protected grayscaleFunc(BufferedImage image) {
        this.image = image;
    }

    private int getHeight() { //important for the conversion
        return image.getHeight();
    }

    private int getWidth() { //important for the conversion
        return image.getWidth();
    }
```

Since methods like the grayscale would not need to be modified in the future, keeping certain functions from being accessed by unauthorized classes and protecting key variables was a critical goal, achieved through **encapsulation**. For instance, the local *image* variable is kept private so it is not incorrectly modified by another class. The *getHeight()* and *getWidth()* methods are also private for the same purpose, while the constructor is protected to ensure it can be accessed by other subclasses[9].

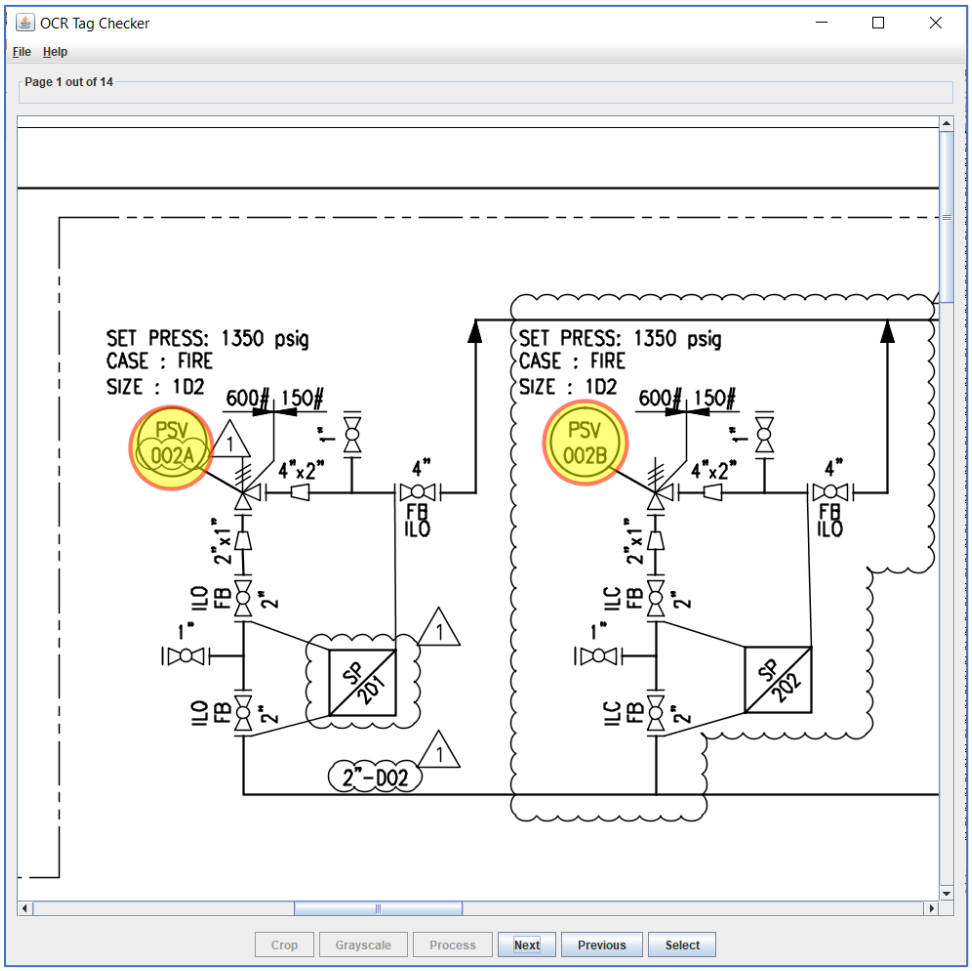
Primary grayscale method (code and explanation)

```
    protected BufferedImage imageGray() { // changes each image pixel to grey using nested for loop and an RGB
                                          // calculation
        if (image == null) {
            return null;
        }
        for (int i = 0; i < getHeight(); i++) {
            for (int j = 0; j < getWidth(); j++) {
                Color imageColor = new Color(image.getRGB(j, i));
                int rgb = ((int) (imageColor.getRed()) + (int) (imageColor.getGreen()) + (int) (imageColor.getBlue()))
                        / 3;
                Color newColor = new Color(rgb, rgb, rgb);
                image.setRGB(j, i, newColor.getRGB());
            }
        }
        return image;
    }

    private BufferedImage image;
```

[9] (What Is an Object? (The Java™ Tutorials > Learning the Java Language > Object-Oriented Programming Concepts), 2021)

With the *getHeight()* and *getWidth()* used to establish parameters, the nested for-loop uses a special algorithm (the **average algorithm**[10]) that divides the average RGB of each pixel to produce a grayscale output. Grayscaling can make a tag easier to read for the OCR software but may consume time so it's an optional choice for the user.

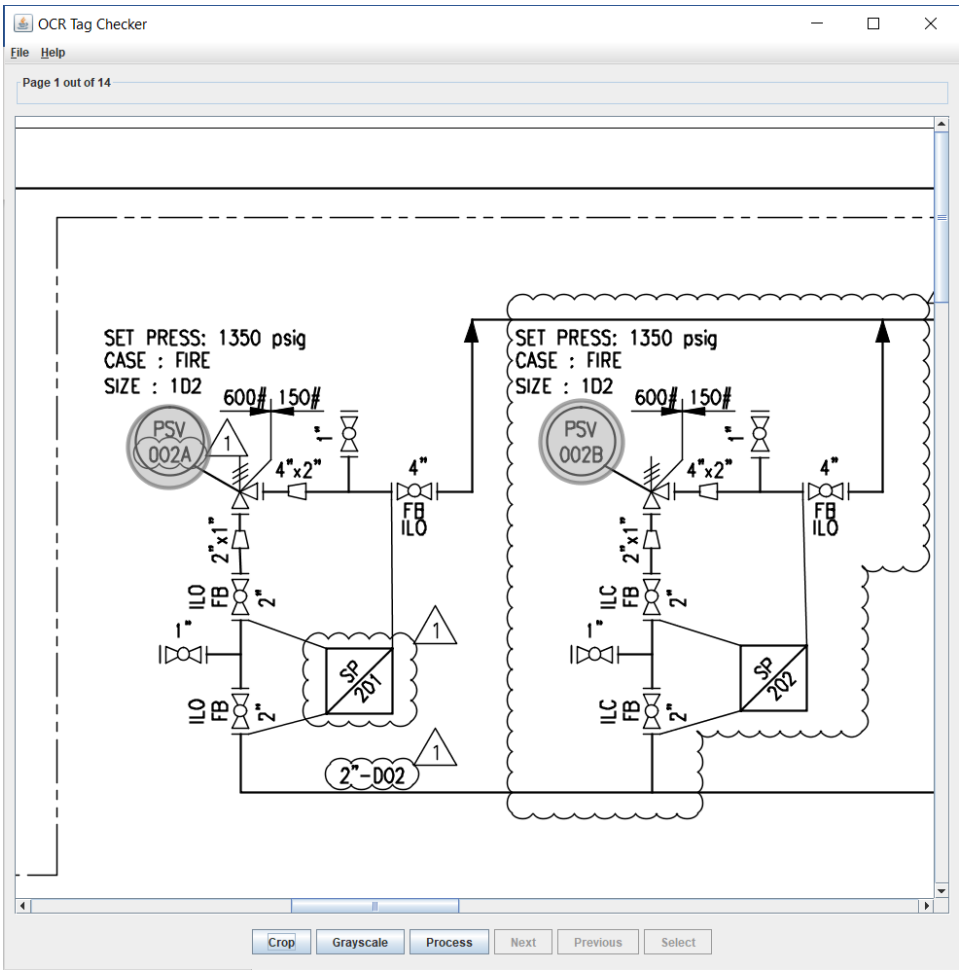
File before grayscale (GUI):



[10] (Converting color to grayscale, 2009)

Candidate #: hpg293

Same file after grayscale (GUI):



cropClassMain

Class variables, inheritance, and constructor (code and explanation)

```
public class cropClassMain extends JPanel {
    private int x, y, w, h;
    private BufferedImage image, image2;
    private Rectangle cropRec;
    private Rectangle finalRect;

    protected cropClassMain(BufferedImage image) {
        this.image = image;
        MouseHandler mouse = new MouseHandler();
        addMouseListener(mouse);
        addMouseMotionListener(mouse);
    }
```

Candidate #: hpg293

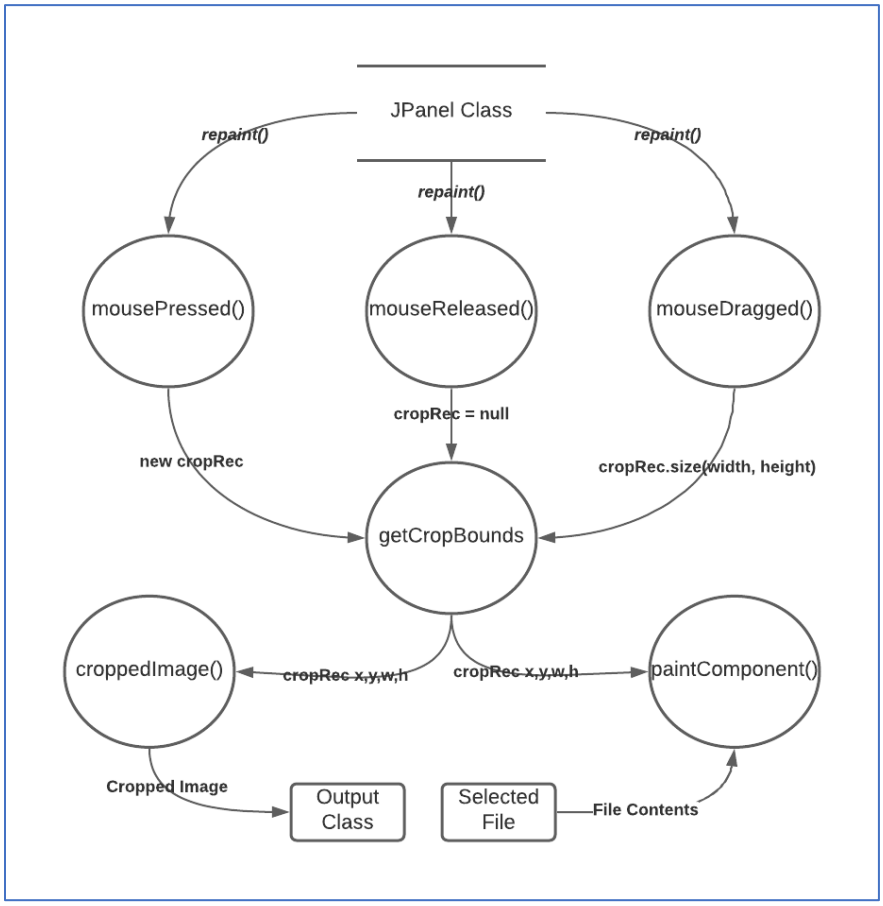
All variables are made private as they only need to be directly accessed by this class (more encapsulation for protection). **Inheritance** of the JPanel is essential, as methods such as the add*MouseListener()* are already in this parent class.

cropClassMain methods (code):

```
private Rectangle getCropBounds() { //returns rectangle with boundaries from the mouse movements
    finalRect = null;
    if (cropRec != null) {
        x = cropRec.x;
        y = cropRec.y;
        w = cropRec.width;
        h = cropRec.height;
    }
    finalRect = new Rectangle(x, y, w, h);
    return finalRect;
}

protected void paintComponent(Graphics g) { //sets image as background and allows for drawing of a cropping rect.
    Rectangle drawCrop = getCropBounds();
    super.paintComponent(g);
    if (drawCrop != null) {
        g.drawImage(image, 0, 0, null);
        g.setColor(Color.red);
        g.drawRect(x, y, w, h);
    }
}

protected class MouseHandler extends MouseAdapter { //allows for mouse input and sets the coordinates
    public void mouseReleased(MouseEvent a) {
        cropRec = null;
        repaint();
    }

    public void mousePressed(MouseEvent a) {
        cropRec = new Rectangle();
        cropRec.setLocation(a.getPoint());
        repaint();
    }

    public void mouseDragged(MouseEvent a) {
        Point point = a.getPoint();
        int recWidth = point.x - cropRec.x;
        int recHeight = point.y - cropRec.y;
        cropRec.setSize(recWidth, recHeight);
        repaint();
    }
}

protected BufferedImage croppedImage() { //uses rectangle coordinates/length/width to return subimage
    image2 = image.getSubimage(Math.abs(x), Math.abs(y), Math.abs(w), Math.abs(h));
    return image2;
}
```

Candidate #: hpg293

cropClassMain methods (data flow diagram and explanation):



As shown above, the JPanel gives all the *MouseListeners()* access to graphics commands. The file is displayed by the paint component in the background of a new tab when initialized in the *GUILayout1* class. When the user presses down, they create a new rectangle (*cropRec*). *cropRec*'s variables (coordinates and size) are filled in when the mouse is dragged. These values are actively displayed by the *paintComponent()* and utilised to create a sub-image with the same coordinates and size variables of the rectangle.

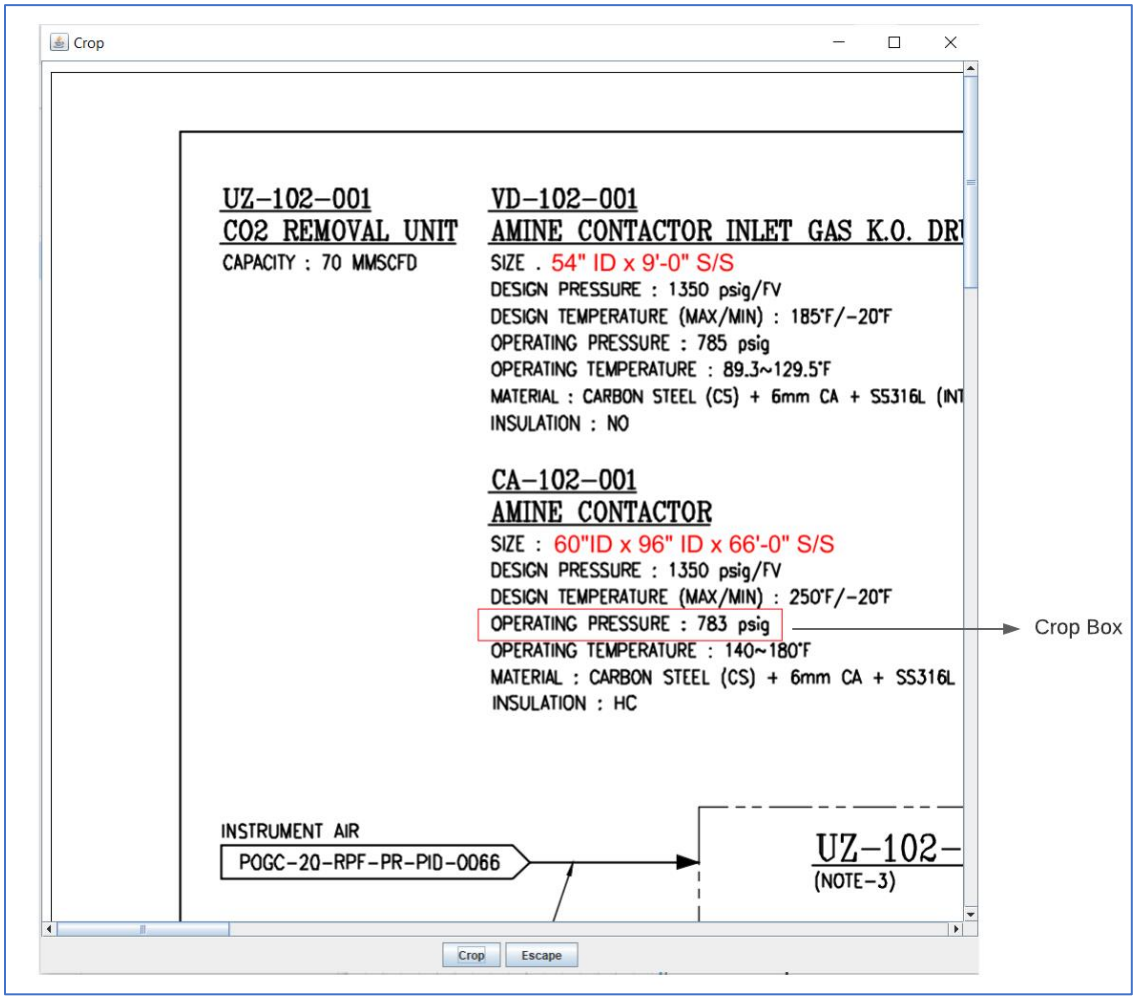
getPreferredSize() override (code and explanation)

```
public Dimension getPreferredSize() { //formats the size of the cropping window (for the JScrollPane)
    return new Dimension(image.getWidth(), image.getHeight());
}
```

The overriding of this method was crucial for implementing a JScrollPane on the graphic when it is displayed. Without it, the JScrollPane could not recognize the image drawing was out of bounds.

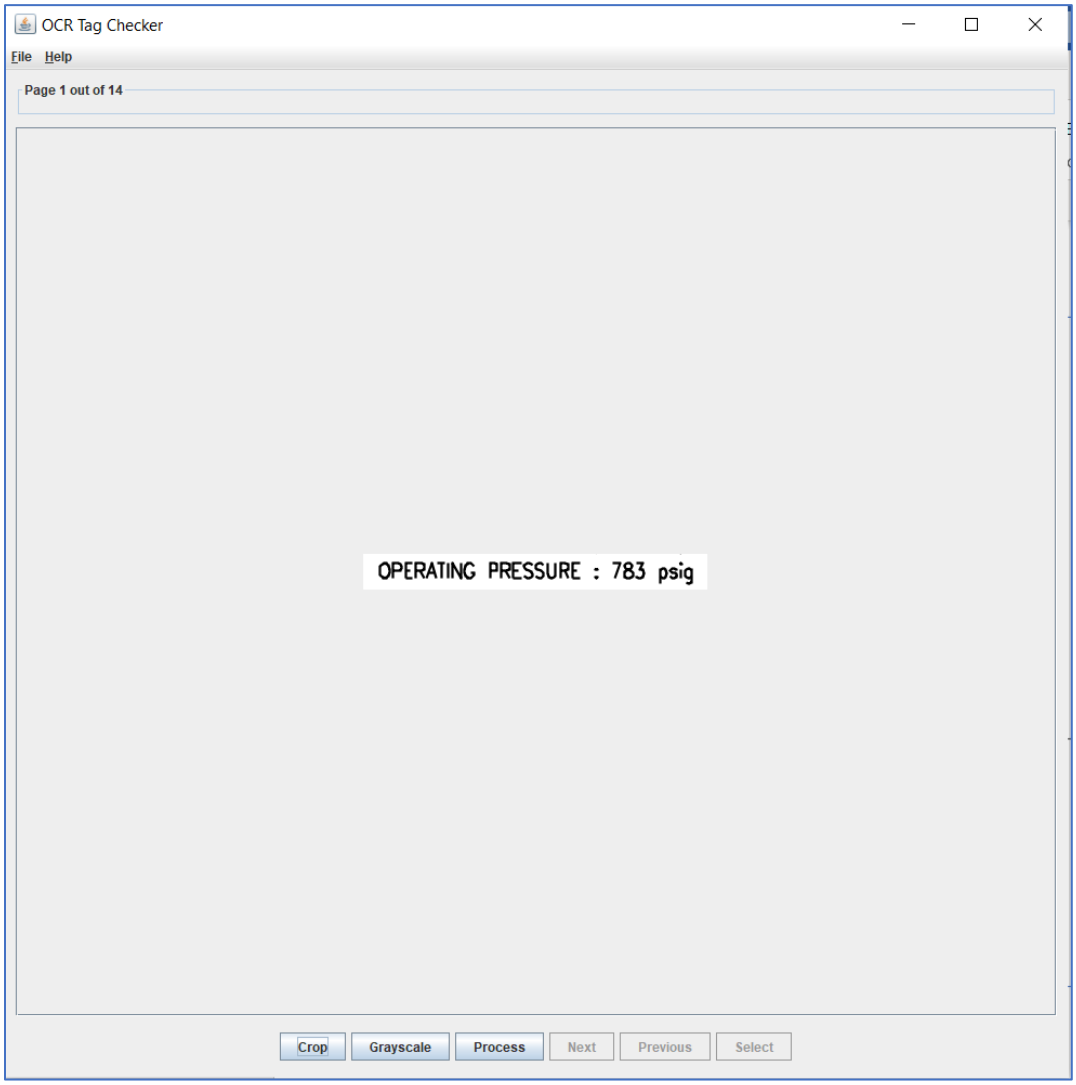
Candidate #: hpg293

File being cropped (GUI):



Candidate #: hpg293

Same file after crop (GUI):



Candidate #: hpg293

GUILayout1 Class

GUILayout1 imports, variables, and main method (code and explanation):

```
import java.awt.BorderLayout;
import java.awt.Event;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.awt.image.RasterFormatException;
import java.io.File;
import java.io.IOException;
import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.ButtonGroup;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.KeyStroke;
import javax.swing.ScrollPaneConstants;
import tess4J.OCR;

public class GUILayout1 extends OCR {
    private JFrame frame, smallFrame; //private variables
    private JPanel contentPane, panel;
    private JButton crop, crop2, escape, grayscale, process, next, previous, select;
    private JLabel imgLabel;
    private ButtonGroup processButtonGroup;
    private static int FRAME_WIDTH = 615;
    private static int FRAME_HEIGHT = 150;
    private boolean isClicked = true;
    private JFileChooser fc = new JFileChooser();
    private File draftDraw;
    private int nextOrPrev = 0;
    private JScrollPane pane;

    public BufferedImage image, backupImage; //public variables and objects
    public String filePath, file_Name;
    public LinkedListQueue queue = new LinkedListQueue();
    public exportTags excel = new exportTags();
    public PDFconverter convert = new PDFconverter();

    public static void main(String[] args) { //main method
        GUILayout1 GUITabs = new GUILayout1();
        GUITabs.start();
    }
```

Most of the imports here are for laying out the GUI. Java Swing was used as it has more sophisticated methods than the older Java AWT, although AWT's *actionListener* interface is used to give functionality to the buttons. All variables only used by this class are private, whereas certain objects and variables are public as they are required for other classes.

Candidate #: hpg293

Sample creation methods (code and explanation):

```
private void start() { // Creates main frame and sets it to a default size
    frame = new JFrame("OCR Tag Checker");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    menuSetUp();
    makeContent();
    frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
    frame.setVisible(true);
    showHelp();
    JOptionPane.showMessageDialog(frame,
            "To see instructions again, refer to the 'USER GUIDE' menu option under the 'HELP' menu (top LEFT)",
            "HELP MENU", JOptionPane.INFORMATION_MESSAGE);
}

private void menuSetUp() { // Adds menu bars to main frame
    JMenuBar menuMain;
    menuMain = new JMenuBar();
    frame.setJMenuBar(menuMain);
    menuMain.add(dropdownMenu1());
    menuMain.add(dropdownMenu2());

}

private void makeContent() { // Adds first content pane and sets path for image
    contentPane = (JPanel) frame.getContentPane();
    contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.Y_AXIS));
    contentPane.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
    ImageIcon icon = new ImageIcon("path");
    JLabel label = new JLabel(icon);
    frame.add(label);
    SouthRegion();

}
```

Each section of the code is extremely organised, with the creation of each component of the GUI (ex. menu, regions) being broken up into individual methods. These methods call upon each other in chronological order to create the GUI. This type of code organisation will assist future developers in optimizing these components of the program.

Candidate #: hpg293

Creation of menus with mnemonics and accelerators (code):

```
private JMenu dropdownMenu1() { // creates items on the first menu bar
    JMenu menu;
    JMenuItem item;

    menu = new JMenu("File");
    menu.setMnemonic(KeyEvent.VK_F);

    item = new JMenuItem("Select");
    item.setMnemonic(KeyEvent.VK_S);
    item.addActionListener(new selectListener());
    item.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S, Event.ALT_MASK));
    menu.add(item);

    item = new JMenuItem("Export");
    item.setMnemonic(KeyEvent.VK_P);
    item.addActionListener(new exportListener());
    item.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_P, Event.ALT_MASK));
    menu.add(item);

    menu.addSeparator();

    item = new JMenuItem("Exit");
    item.setMnemonic(KeyEvent.VK_X);
    item.addActionListener(new exitListener());
    item.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q, Event.CTRL_MASK));
    menu.add(item);

    return menu;
}

private JMenu dropdownMenu2() { // creates items on the second menu bar
    JMenu menu;
    JMenuItem item;

    menu = new JMenu("Help");
    menu.setMnemonic(KeyEvent.VK_H);

    item = new JMenuItem("User Guide");
    item.setMnemonic(KeyEvent.VK_G);
    item.addActionListener(new helpListener());
    item.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_G, Event.ALT_MASK));
    menu.add(item);
    menu.addSeparator();

    item = new JMenuItem("Reset");
    item.setMnemonic(KeyEvent.VK_R);
    item.addActionListener(new resetListener());
    item.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_R, Event.ALT_MASK));
    menu.add(item);

    item = new JMenuItem("Close");
    item.setMnemonic(KeyEvent.VK_C);
    item.addActionListener(new clearListener());
    item.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C, Event.ALT_MASK));
    menu.add(item);

    return menu;
}
```

Candidate #: hpg293

SouthRegion() buttons (code):

```
private void SouthRegion() { // creates south region where all of the JButtons will be stored
    contentPane = (JPanel) frame.getContentPane();
    contentPane.setLayout(new BorderLayout(10, 10));
    panel = new JPanel();
    panel.setBorder(BorderFactory.createTitledBorder(
            "Please refer to the 'USER GUIDE' under the 'HELP' menu for additional assistance and instructions!"));
    JPanel buttonPanel = new JPanel();

    processButtonGroup = new ButtonGroup();

    crop = new JButton("Crop");
    processButtonGroup.add(crop);
    crop.addActionListener(new cropButton());
    buttonPanel.add(crop);

    grayscale = new JButton("Grayscale");
    processButtonGroup.add(grayscale);
    grayscale.addActionListener(new grayscale());
    buttonPanel.add(grayscale);

    process = new JButton("Process");
    processButtonGroup.add(process);
    process.addActionListener(new process());
    processButtonGroup.add(process);
    buttonPanel.add(process);

    next = new JButton("Next");
    processButtonGroup.add(next);
    next.addActionListener(new next());
    processButtonGroup.add(process);
    buttonPanel.add(next);

    previous = new JButton("Previous");
    processButtonGroup.add(previous);
    previous.addActionListener(new previous());
    processButtonGroup.add(process);
    buttonPanel.add(previous);

    select = new JButton("Select");
    processButtonGroup.add(select);
    select.addActionListener(new select());
    processButtonGroup.add(process);
    buttonPanel.add(select);

    contentPane.add(panel, BorderLayout.CENTER);
    contentPane.add(buttonPanel, BorderLayout.SOUTH);

}
```

Candidate #: hpg293

NorthRegion() method (code and explanation):

```
private void NorthRegion(String findFile) throws IOException { // adds file that is supposed to be processed to the
                                                               // north
                                                               // region
    if (findFile.contains("\\")) {
        findFile.replace("\\", "\\\\");
    }

    try {
        if (findFile.contains(".pdf") || findFile.contains(".PDF")) { //checks if image is a PDF
            contentPane.remove(panel);
            frame.setSize(1000, 1000); // frame size adjusts to appropriate size
            convert.generateImageFromPDF(findFile);
            image = convert.generateImage(); //image is received from other class
            imgLabel = new JLabel((new ImageIcon(image)), JLabel.CENTER);
            contentPane.add(imgLabel, BorderLayout.NORTH);
            crop.setEnabled(false);
            grayscale.setEnabled(false);
            process.setEnabled(false);
            isClicked = false;
            panel = new JPanel();
            panel.setBorder(
                    BorderFactory.createTitledBorder("Page " + (nextOrPrev + 1) + " out of " + convert.listSize())); //shows PDF page user is on
            contentPane.add(panel, BorderLayout.NORTH);
        } else {
            JOptionPane.showMessageDialog(frame, "Please ensure you have selected an approved file type",
                    "NO FILE SELECTED", JOptionPane.ERROR_MESSAGE);
        }
    } catch (NullPointerException incorrectFile) {
        JOptionPane.showMessageDialog(frame, "Please ensure you have selected a file", "INCORRECT FILE",
                JOptionPane.ERROR_MESSAGE);
        contentPane.removeAll();
        SouthRegion();
        frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
    } catch (IllegalArgumentException incorrectFile2) {
        contentPane.removeAll();
        JOptionPane.showMessageDialog(frame, "Please ensure you have selected an approved file type",
                "INCORRECT FILE", JOptionPane.ERROR_MESSAGE);
        SouthRegion();
        frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
    } catch (IOException fileOpenProblem) {
        contentPane.removeAll();
        JOptionPane.showMessageDialog(frame,
                "Error opening/reading file. Please ensure the file is not currently OPENED", "OPEN/READ ERROR",
                JOptionPane.ERROR_MESSAGE);
        SouthRegion();
        frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
    }

}
```

The *NorthRegion()* method runs after a file is selected. The file path is used as a parameter and is passed to the *generateImageFromPDF()* method, where each page is converted to a bufferedImage and stored. The first page is displayed as default and the panel adjusts to the correct size. All potential errors in this method are handled and solutions are presented to the user.

Candidate #: hpg293

ShowHelp() instruction guide method (code):

```
private void showHelp() { // creates help menu for user (displayed at the start of the program)
    if (JOptionPane.showConfirmDialog(frame,
            "Welcome to the OCR Tag Reader! To move onto instructions, press 'OK'. \nIf you are ready to begin working, press 'CANCEL'.",
            "WELCOME", JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
        if (JOptionPane.showConfirmDialog(frame,
                "1. To select a file for processing, use the 'SELECT' option under the 'FILE' \nmenu option (top LEFT)."
                + "\n2. Select any PDF file! \n3. Once the file is loaded in, use the 'NEXT' and 'PREVIOUS'\n buttons (BOTTOM) to cycle to the desired page."
                + " \n4. Use the 'SELECT' button to choose the page.",
                "SELECTING FILES", JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
            if (JOptionPane.showConfirmDialog(frame,
                    "1. Use the 'GRAYSCALE' button (BOTTOM) to remove all color from the file and make it easier"
                    + "\nto process (this should be done if the tags font/background is NOT black and white). "
                    + "\n2. Use the 'CROP' button (BOTTOM) to crop the tag by dragging your cursor across it \nand then pressing the 'CROP' button once more.",
                    "PRE-PROCESSING", JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
                if (JOptionPane.showConfirmDialog(frame,
                        "1. Once the tag has been captured, you can press 'PROCESS' (BOTTOM) to read \nand store the contents.\nAt this point, you may: "
                        + "\n - Export the tag to an Excel Spreadsheet using the 'EXPORT' menu option\n under 'FILE' option (top LEFT) "
                        + "\n - Use the 'RESET' menu option under the 'HELP' menu (top LEFT) and continue\n accumulating tags from the SAME DOCUMENT before exporting "
                        + "\n - Use the 'CLEAR' menu option under the 'HELP' menu (top LEFT),\nselect a NEW DOCUMENT, and continue accumulating tags before exporting",
                        "PROCESSING and EXPORTING", JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
                    if (JOptionPane.showConfirmDialog(frame,
                            "1. Located under the 'HELP' menu (top LEFT), 'RESET' will clear all modifications made to a\ndisplayed file, "
                            + "'CLEAR' will remove a specified file and allow for the user to select a new one",
                            "RESET and CLEAR", JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
                        if (JOptionPane.showConfirmDialog(frame,
                                "To report any errors/new ideas for future use, please contact the developer at \n********@gmail.com "
                                + "\nFuture development ideas:",
                                "ERRORS and FUTURE DEVELOPMENT",
                                JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
                        }
                    }
                }

            }
        }
    }
}
```

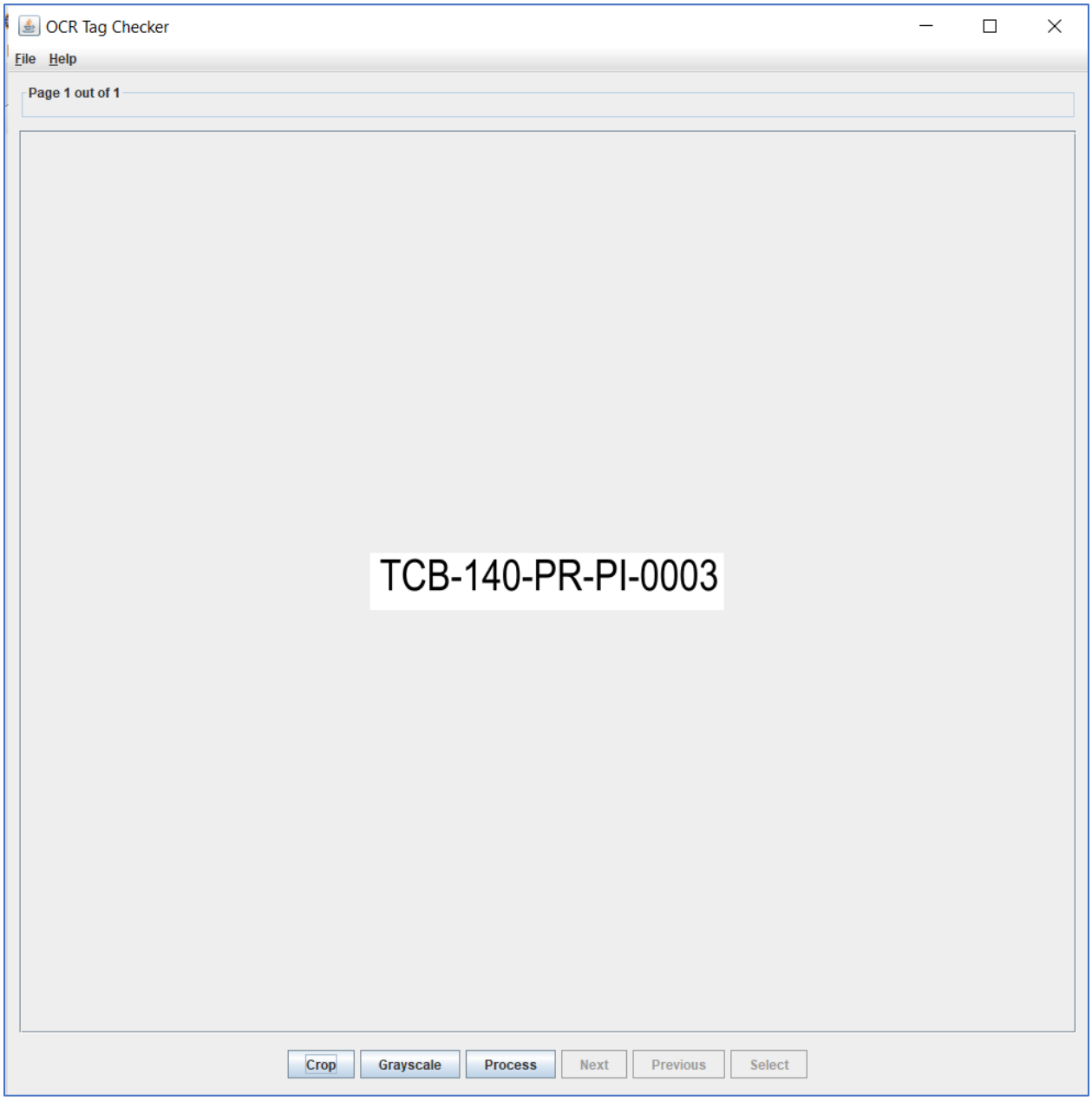
resetListener class (code + explanation):

```
private class resetListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if (image == null) {

            JOptionPane.showMessageDialog(frame, "No file selected, no need to reset!", "RESET NOT NEEDED",
                    JOptionPane.INFORMATION_MESSAGE);
        } else {
            contentPane.removeAll();
            image = null;
            imgLabel = null;
            pane = null;
            SouthRegion();
            grayscale.setEnabled(false);
            crop.setEnabled(false);
            process.setEnabled(false);
            panel = new JPanel();
            panel.setBorder(
                    BorderFactory.createTitledBorder("Page " + (nextOrPrev + 1) + " out of " + convert.listSize()));
            contentPane.add(panel, BorderLayout.NORTH);
            convert.removeModImage(nextOrPrev, backupImage);
            image = convert.generateImage();
            imgLabel = new JLabel(new ImageIcon(image));
            pane = new JScrollPane(imgLabel);
            pane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
            pane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
            contentPane.add(pane);
            frame.setSize(1000, 999);
            frame.setSize(1000, 1000);
            JOptionPane.showMessageDialog(frame, "Image reset!", "RESET SUCCESSFUL",
                    JOptionPane.INFORMATION_MESSAGE);
            isClicked = false;
        }
    }
}
```

The reset button removes all cropping changes to a file to allow for multiple tag captures. Keeps all grayscale changes to save time.

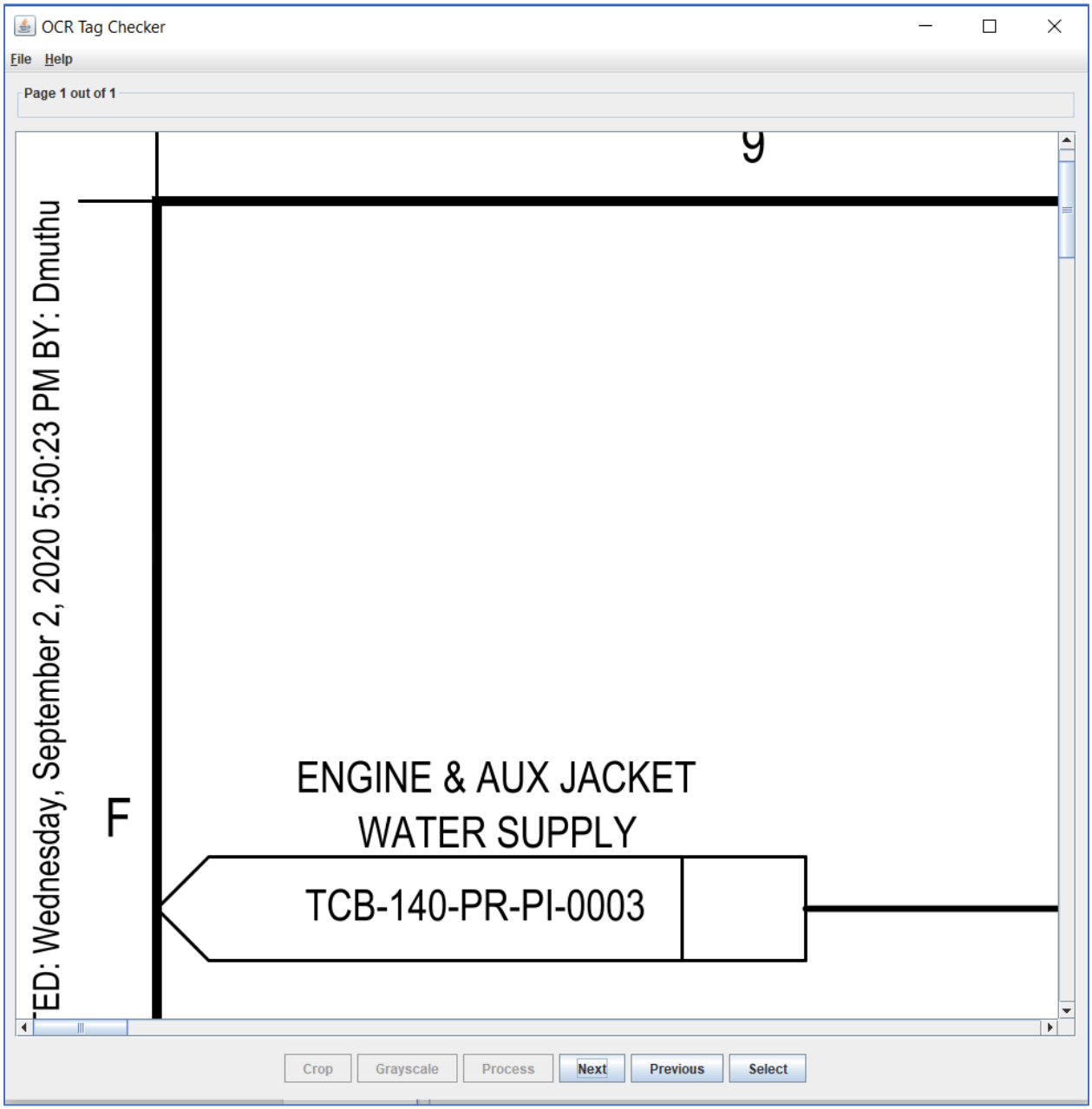
Candidate #: hpg293

File before being reset (GUI):



Candidate #: hpg293

Same file after being reset (GUI):

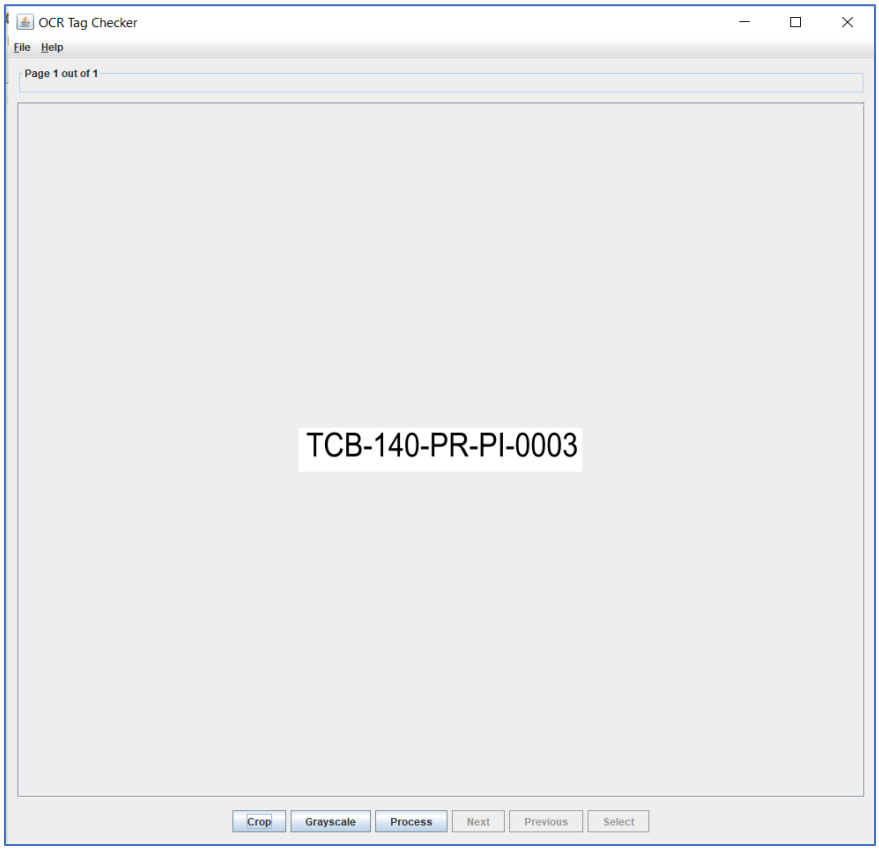


clearListener class (code + explanation):

```
private class clearListener implements ActionListener { //clear all fields, including the currently stored file
    public void actionPerformed(ActionEvent e) {
        if (image == null) {
            JOptionPane.showMessageDialog(frame, "No file selected, no need to clear!", "CLEAR NOT NEEDED",
                    JOptionPane.INFORMATION_MESSAGE);
        } else {
            contentPane.removeAll();
            convert.clearList();
            image = null;
            imgLabel = null;
            pane = null;
            file_Name = null;
            nextOrPrev = 0;
            SouthRegion();
            frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
            JOptionPane.showMessageDialog(frame, "All fields cleared!", "CLEAR SUCCESSFUL",
                    JOptionPane.INFORMATION_MESSAGE);
            isClicked = false;
        }
    }
}
```

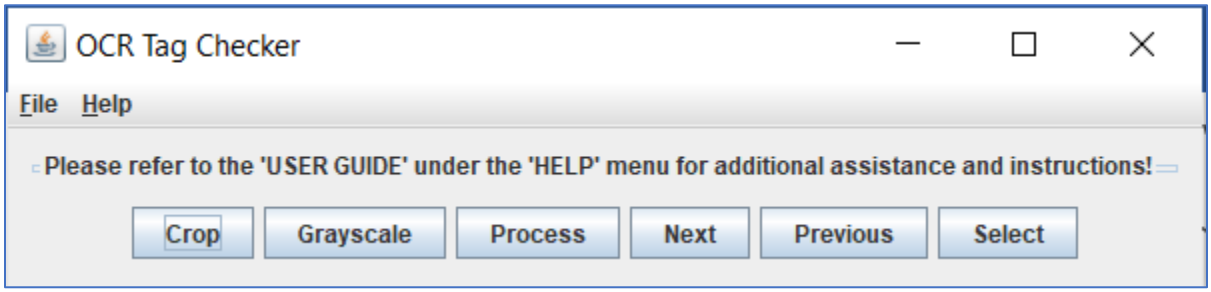
The clear button removes all fields and allows for the selection of a new file.

File before being cleared:



Candidate #: hpg293

Same file after being cleared:



Bibliography:

Sources for explanation:

Rideau-info.com. 2021. *All About Digital Photos - What is DPI?*. [online] Available at: <http://www.rideau-info.com/photos/whatisdpi.html>.

Pdfbox.apache.org. 2021. *Apache PDFBox | Download*. [online] Available at: <https://pdfbox.apache.org/download.cgi>.

Poi.apache.org. 2021. *Apache POI - the Java API for Microsoft Documents*. [online] Available at: <https://poi.apache.org/>.

John D. Cook | Applied Mathematics Consulting. 2021. *Converting color to grayscale*. [online] Available at: <https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/>.

Cook, C., 2005. *Blue Pelican Java*. College Station: Virtualbookworm.com.

Homeandlearn.org. 2021. *Excel VBA Programming - Multidimensional Arrays*. [online] Available at: <https://www.homeandlearn.org/multidimensional_arrays.html>.

R. Smith, 2007, "An Overview of the Tesseract OCR Engine," *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*

DEV Community. 2021. *How to convert PDF to image in Java*. [online] Available at: <https://dev.to/eiceblue/how-to-convert-pdf-to-image-in-java-491g>.

GitHub. 2021. *Releases · tesseract-ocr/tesseract*. [online] Available at: <https://github.com/tesseract-ocr/tesseract/releases>.

Docs.oracle.com. 2021. *What Is an Object? (The Java™ Tutorials > Learning the Java Language > Object-Oriented Programming Concepts)*. [online] Available at: <https://docs.oracle.com/javase/tutorial/java/concepts/object.html>.

Sources for all Diagrams

Lucidchart. 2021. *Online Diagram Software & Visual Solution | Lucidchart*. [online] Available at: <https://www.lucidchart.com/pages/>.

Tutorialspoint.com. 2021. *UML - Class Diagram - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/uml/uml_class_diagram.htm>.

1063 Words