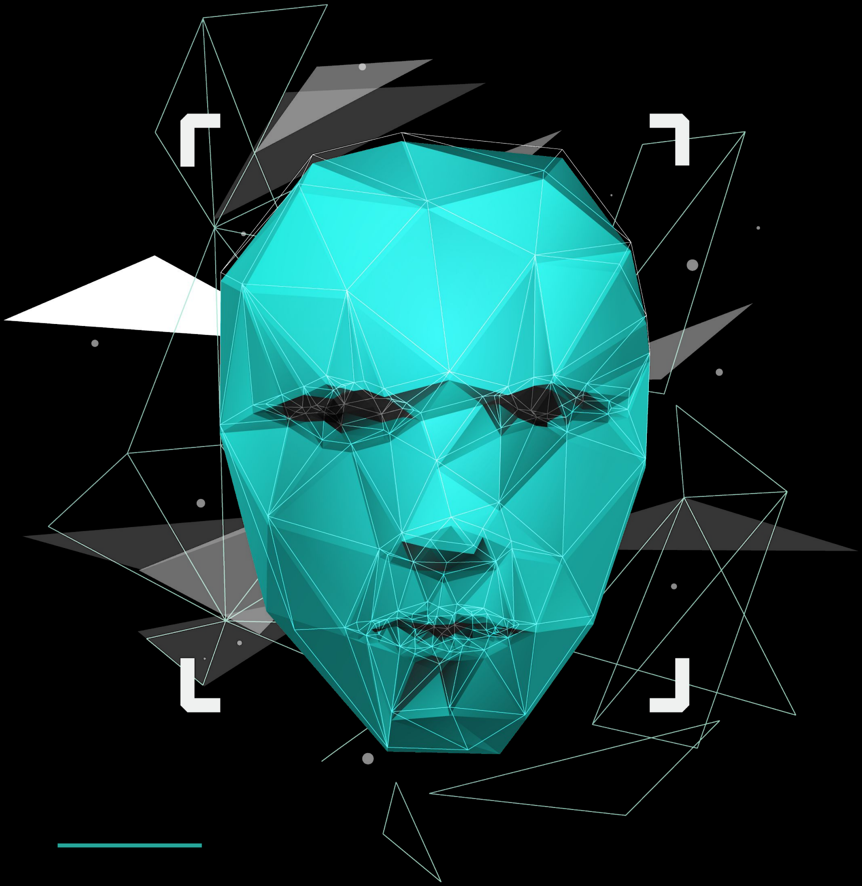
FACE DETECTION

Raunak Singh
Akhil Sharma
Ishan Bansal
Antas
Vipul Wasnik



Introduction to Face Detection using OpenCV

Face detection -- also called facial detection -- is an artificial intelligence (AI) based computer technology used to find and identify human faces in digital images. Face detection technology can be applied to various fields -- including security, biometrics, law enforcement, entertainment and personal safety -- to provide surveillance and tracking of people in real time.

OpenCV is a software toolkit for processing real-time image and video, as well as providing analytics, and machine learning capabilities.

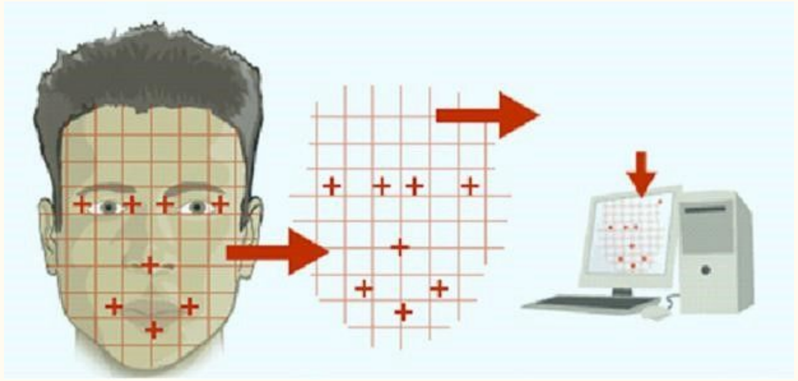
Face Detection Methods

EigenFaces Method

An eigenface is the name given to a set of eigenvectors when used in the computer vision problem of human face recognition. The approach of using eigenfaces for recognition was developed by Sirovich and Kirby (1987) and used by Matthew Turk and Alex Pentland in face classification. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images. The eigenfaces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images. Classification can be achieved by comparing how faces are represented by the basis set.

Geometric Method

A face can be recognized even when the details of the individual features (such as eyes, nose and mouth) are no longer resolved. The remaining information is, in a sense, purely geometrical and represents what is left at a very coarse resolution. The idea is to extract relative position and other parameters of distinctive features such as eyes, mouth, nose and chin. This was the first approach towards an automated recognition of faces



Haar Cascade Method

It is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them. The model created from this training is available at the OpenCV GitHub repository:

https://github.com/opencv/opencv/tree/master/data/haarcascades

The repository has the models stored in XML files, and can be read with the OpenCV methods. These include models for face detection, eye detection, upper body and lower body detection, license plate detection etc.

Haar Cascade Method

This method works by scanning a picture from the top left corner and moving down across small blocks of data while scanning them.

The OpenCv cascade works by breaking the problem of detecting faces into multiple stages. For each block it encounters, it performs a rough and quick test.

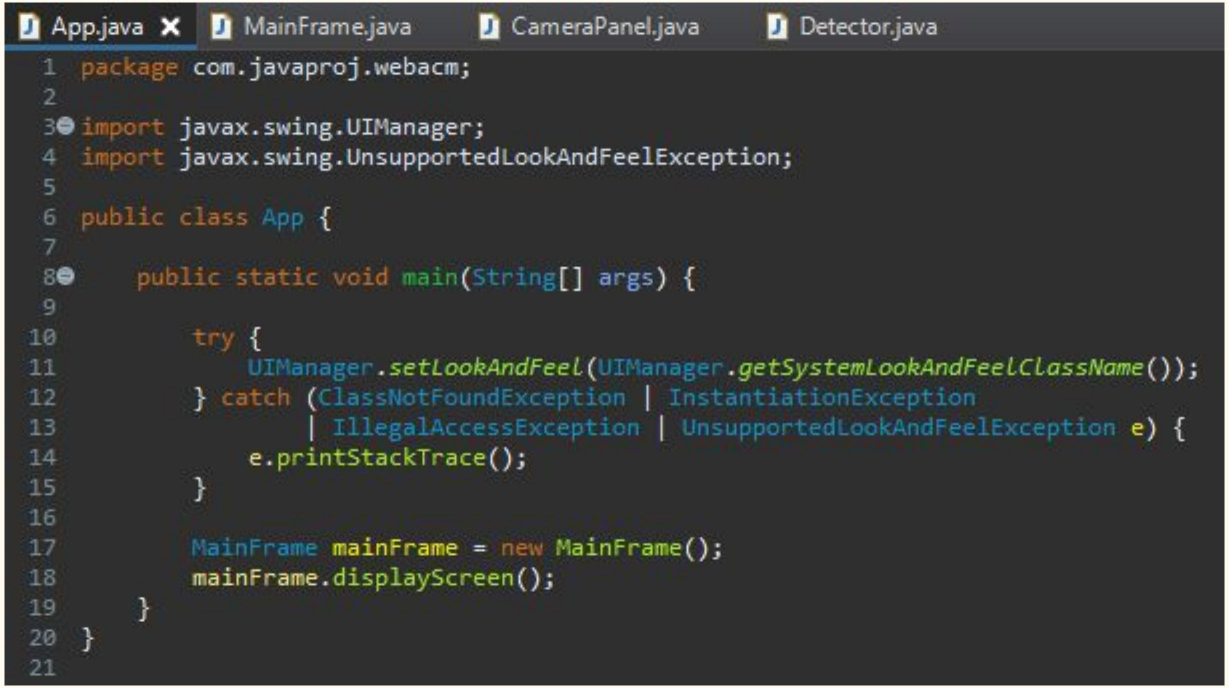
If the test is successful, that is, if the block passes the test, a more sophisticated test is performed.

CASCADE CLASSIFIERS

Cascading classifiers are trained with several hundred "positive" sample views of a particular object and arbitrary "negative" images of the same size. After the classifier is trained it can be applied to a region of an image and detect the object in question. To search for the object in the entire frame, the search window can be moved across the image and check every location for the classifier. This process is most commonly used in image processing for object detection and tracking, primarily facial detection and recognition.

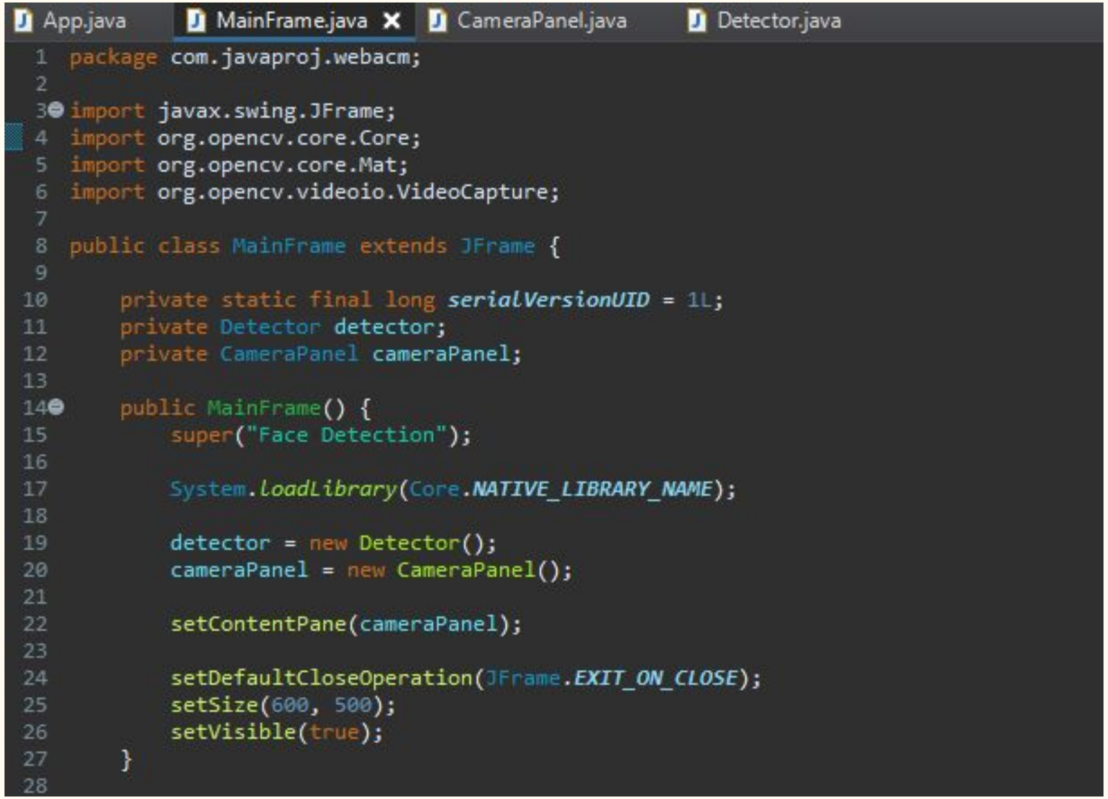
Java Program

App.java



```java
package com.javaproj.webacm;

import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;

public class App {

    public static void main(String[] args) {

        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (ClassNotFoundException | InstantiationException
                | IllegalAccessException | UnsupportedLookAndFeelException e) {
            e.printStackTrace();
        }

        MainFrame mainFrame = new MainFrame();
        mainFrame.displayScreen();
    }
}

```

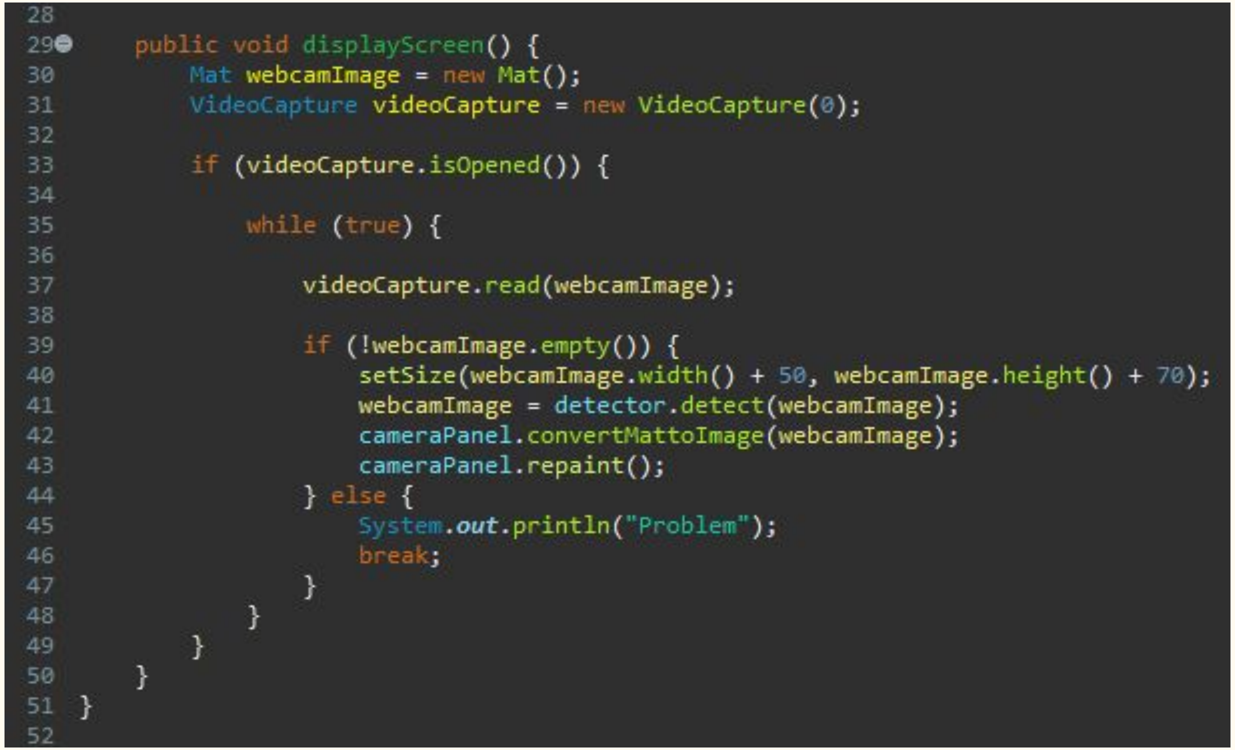
MainFrame.java



App.java MainFrame.java CameraPanel.java Detector.java

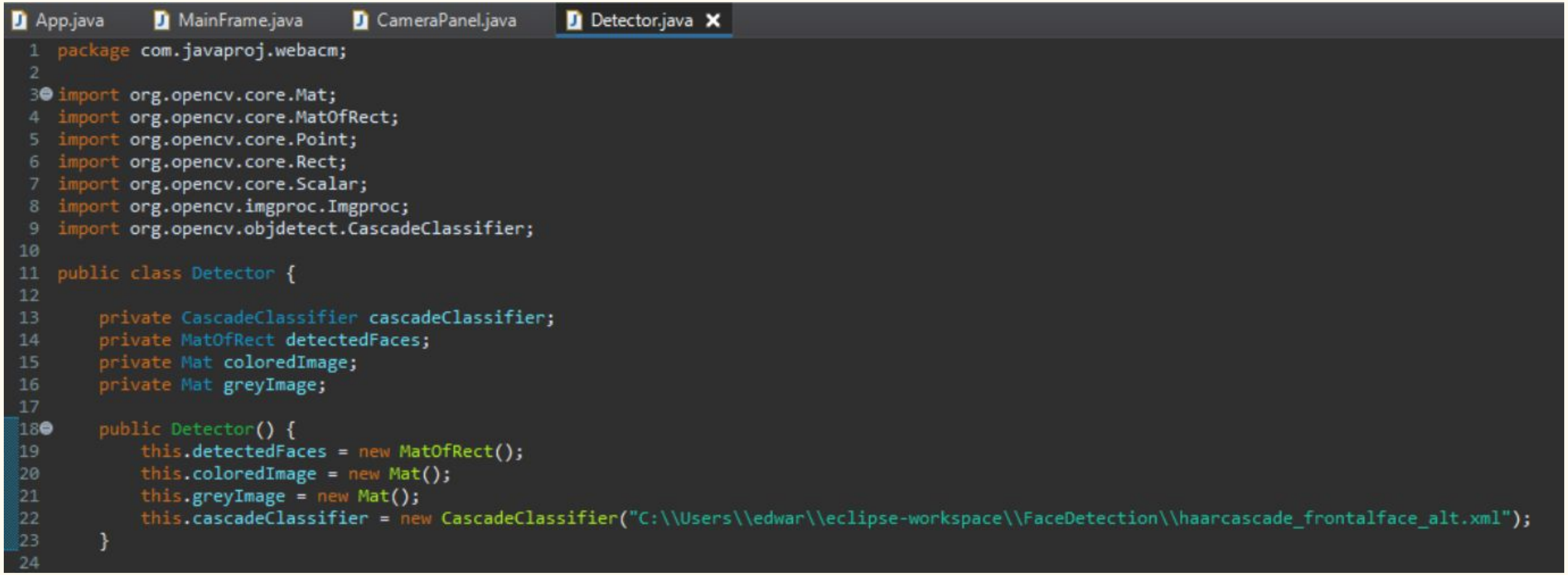
```java
package com.javaproj.webacm;

import javax.swing.JFrame;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.videoio.VideoCapture;

public class MainFrame extends JFrame {

    private static final long serialVersionUID = 1L;
    private Detector detector;
    private CameraPanel cameraPanel;

    public MainFrame() {
        super("Face Detection");

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        detector = new Detector();
        cameraPanel = new CameraPanel();

        setContentPane(cameraPanel);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 500);
        setVisible(true);
    }

```

MainFrame.java



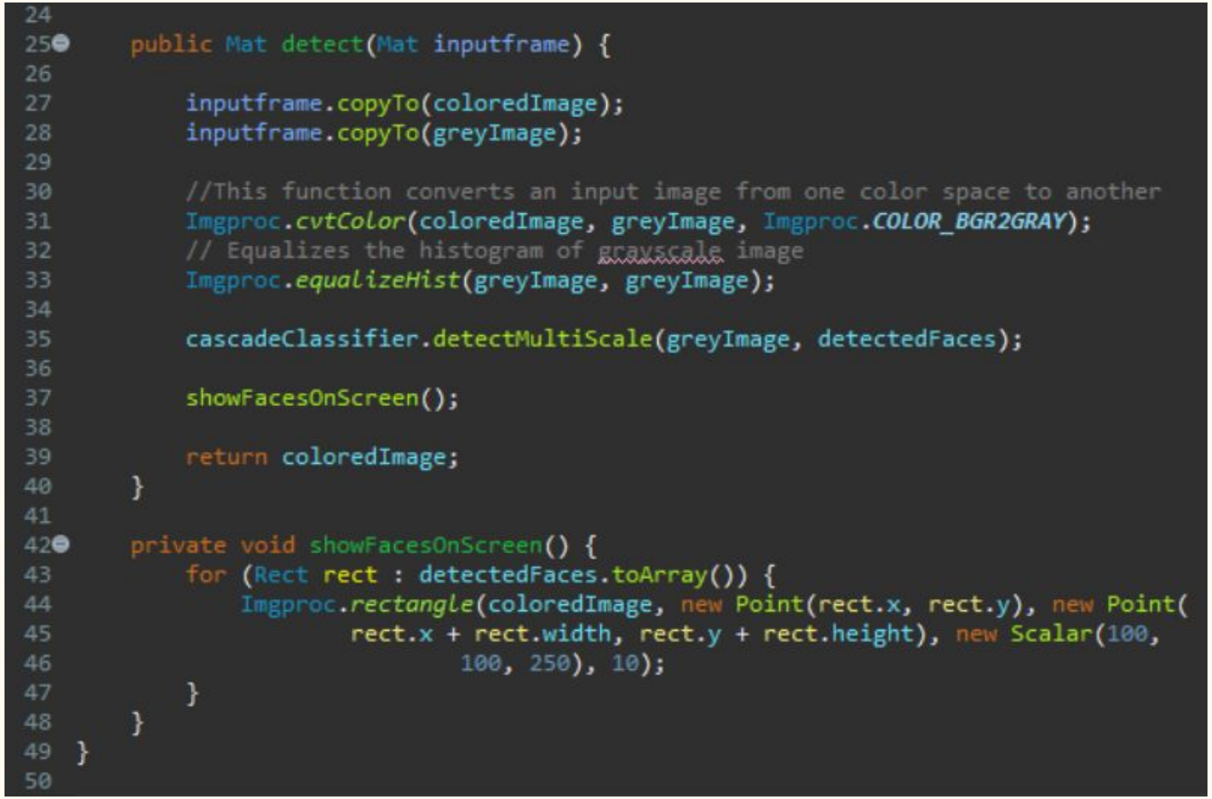
```java

    public void displayScreen() {
        Mat webcamImage = new Mat();
        VideoCapture videoCapture = new VideoCapture(0);

        if (videoCapture.isOpened()) {

            while (true) {

                videoCapture.read(webcamImage);

                if (!webcamImage.empty()) {
                    setSize(webcamImage.width() + 50, webcamImage.height() + 70);
                    webcamImage = detector.detect(webcamImage);
                    cameraPanel.convertMattoImage(webcamImage);
                    cameraPanel.repaint();
                } else {
                    System.out.println("Problem");
                    break;
                }
            }
        }
    }
}

```

Detector.java



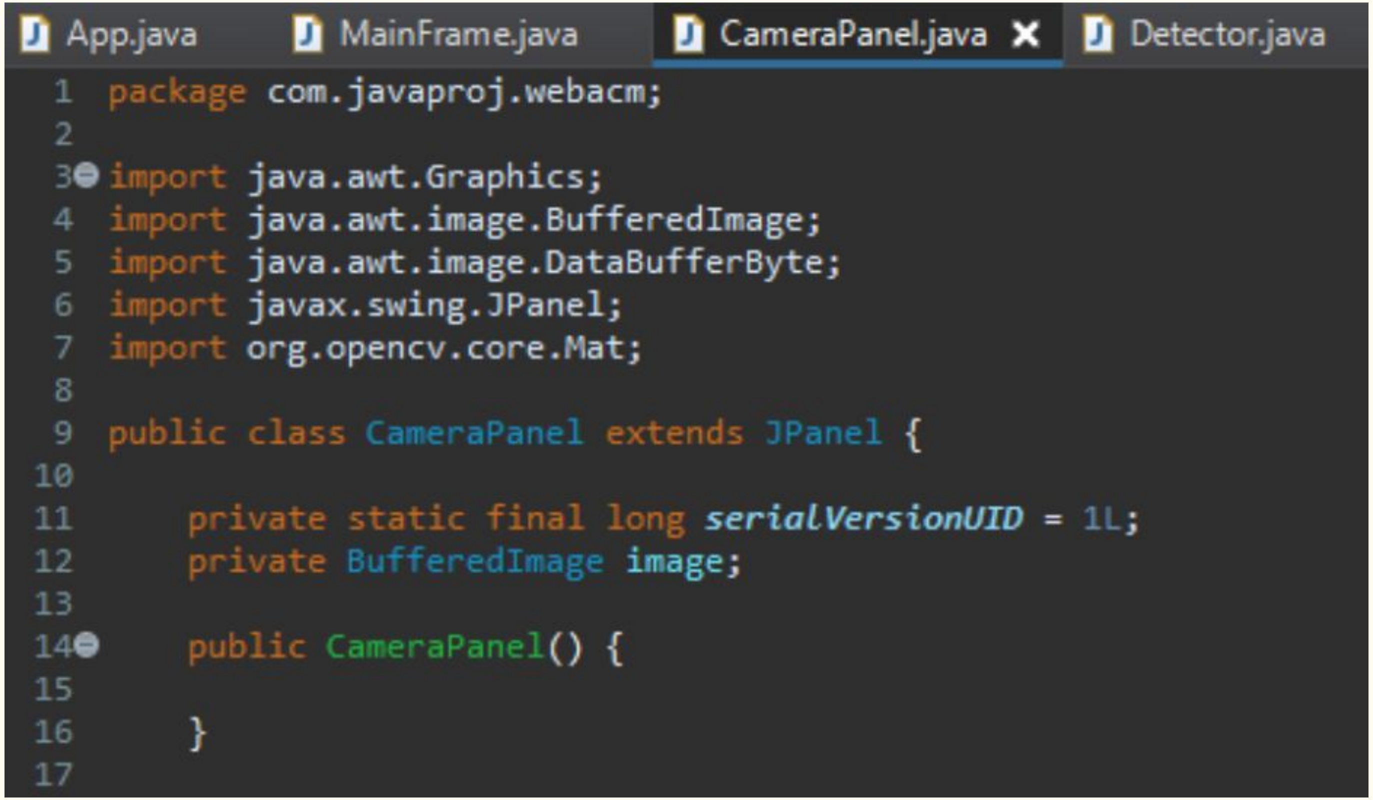
```java
package com.javaproj.webacm;

import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

public class Detector {

    private CascadeClassifier cascadeClassifier;
    private MatOfRect detectedFaces;
    private Mat coloredImage;
    private Mat greyImage;

    public Detector() {
        this.detectedFaces = new MatOfRect();
        this.coloredImage = new Mat();
        this.greyImage = new Mat();
        this.cascadeClassifier = new CascadeClassifier("C:\\Users\\edwar\\eclipse-workspace\\FaceDetection\\haarcascade_frontalface_alt.xml");
    }

```

Detector.java



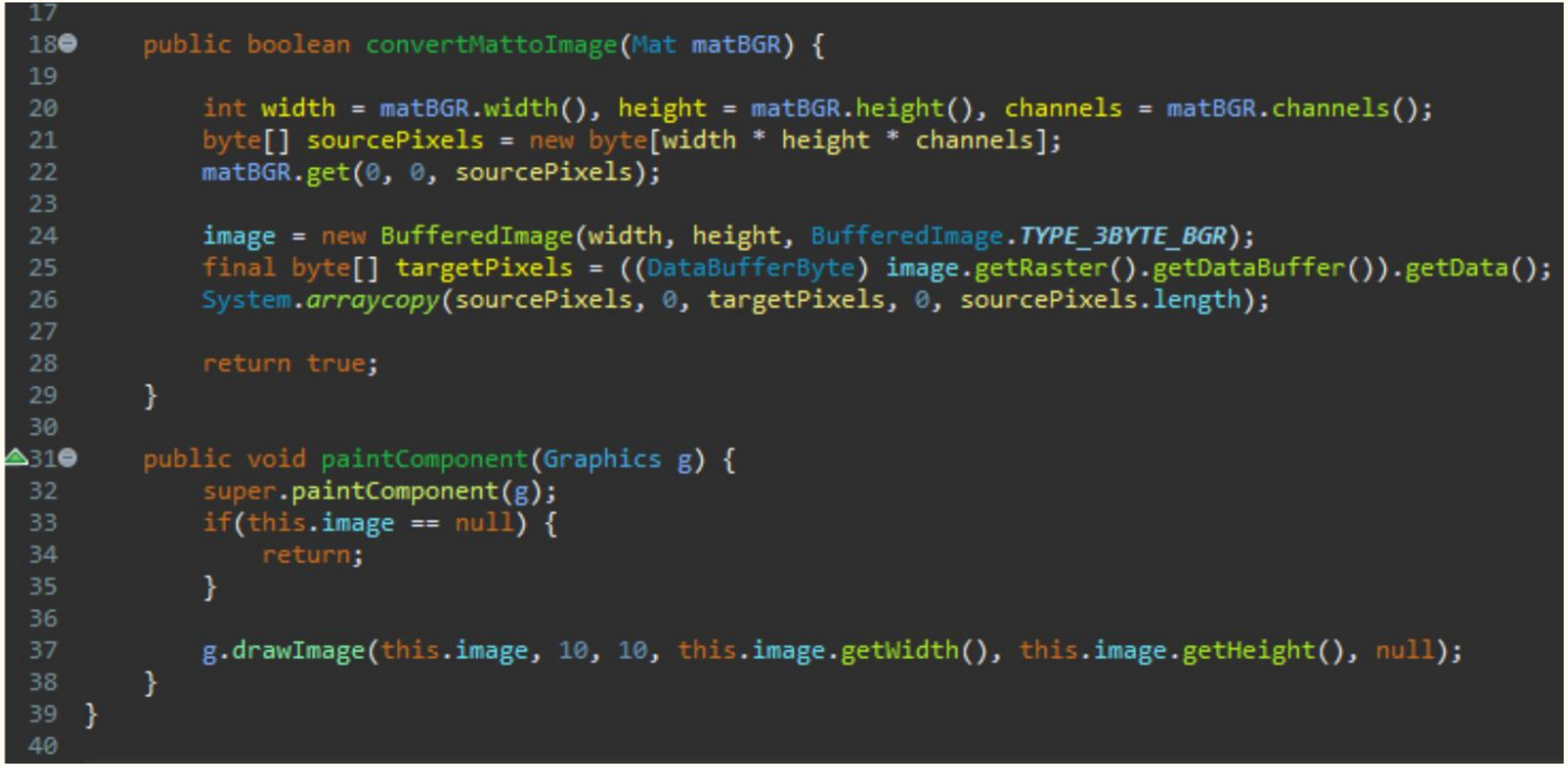
```
    public Mat detect(Mat inputframe) {

        inputframe.copyTo(coloredImage);
        inputframe.copyTo(greyImage);

        //This function converts an input image from one color space to another
        Imgproc.cvtColor(coloredImage, greyImage, Imgproc.COLOR_BGR2GRAY);
        // Equalizes the histogram of grayscale image
        Imgproc.equalizeHist(greyImage, greyImage);

        cascadeClassifier.detectMultiScale(greyImage, detectedFaces);

        showFacesOnScreen();

        return coloredImage;
    }

    private void showFacesOnScreen() {
        for (Rect rect : detectedFaces.toArray()) {
            Imgproc.rectangle(coloredImage, new Point(rect.x, rect.y), new Point(
                    rect.x + rect.width, rect.y + rect.height), new Scalar(100,
                            100, 250), 10);
        }
    }
}
```

CameraPanel.java



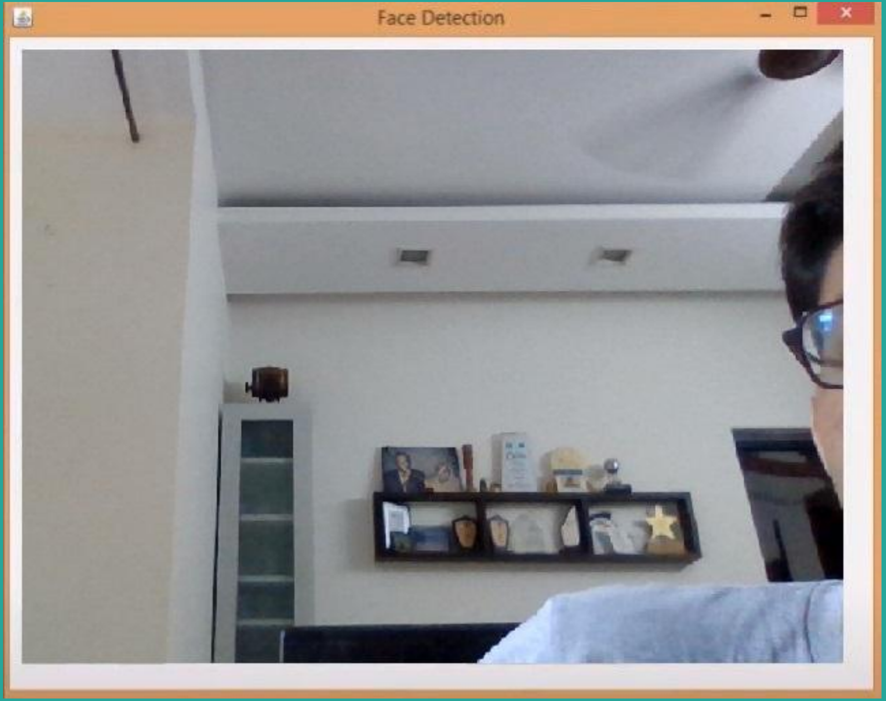
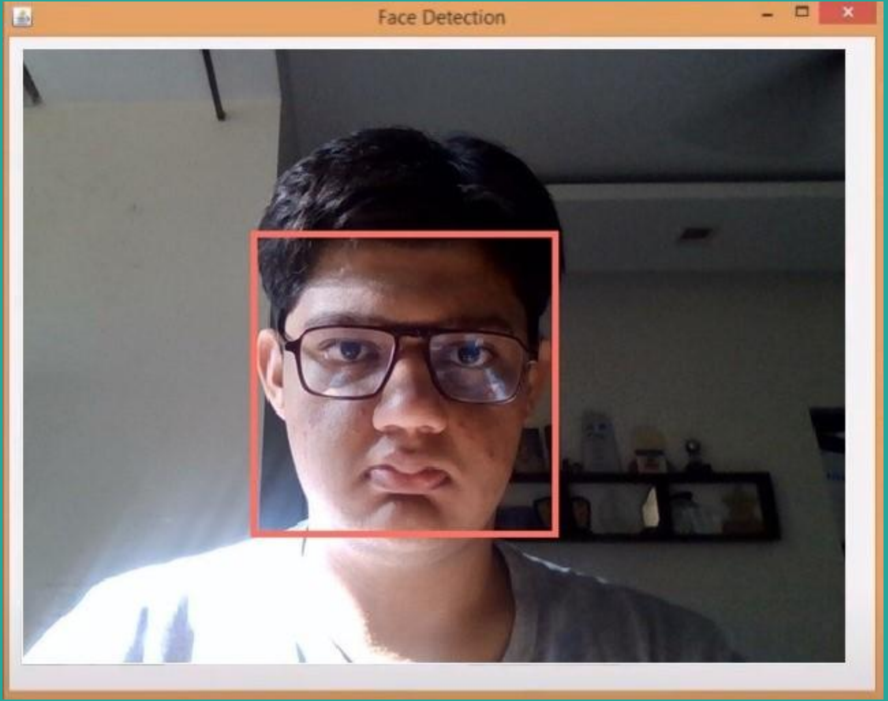
```java
package com.javaproj.webacm;

import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import javax.swing.JPanel;
import org.opencv.core.Mat;

public class CameraPanel extends JPanel {

    private static final long serialVersionUID = 1L;
    private BufferedImage image;

    public CameraPanel() {

    }

```

CameraPanel.java



```java

    public boolean convertMattoImage(Mat matBGR) {

        int width = matBGR.width(), height = matBGR.height(), channels = matBGR.channels();
        byte[] sourcePixels = new byte[width * height * channels];
        matBGR.get(0, 0, sourcePixels);

        image = new BufferedImage(width, height, BufferedImage.TYPE_3BYTE_BGR);
        final byte[] targetPixels = ((DataBufferByte) image.getRaster().getDataBuffer()).getData();
        System.arraycopy(sourcePixels, 0, targetPixels, 0, sourcePixels.length);

        return true;
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        if(this.image == null) {
            return;
        }

        g.drawImage(this.image, 10, 10, this.image.getWidth(), this.image.getHeight(), null);
    }
}

```

Program Output


Face Detection

Face Detection