SWE1010- DIGITAL IMAGE PROCESSING

J-COMPONENT FINAL REVIEW:

TITLE: FACE RECOGNITION USING OPENCV

FACULTY: DR. GEETA S

TEAM MEMBERS:

DESHIREDDY KRISHNAREDDY (20MIS1069)

APPIREDDY GOWTHAMREDDY (20MIS1175)

EXECUTABLE CODES:

1) Main.java

```
package application;


import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.fxml.FXML;
```

```
import javafx.fxml.FXMLLoader;


public class Main extends Application
@Override
	public void start(Stage primaryStage) {
		try {
			BorderPane root =
(BorderPane)FXMLLoader.load(getClass().getResource("Sample.fxml"));
			Scene scene = new Scene(root,1350,720);



	scene.getStylesheets().add(getClass().getResource("application.css").toE
xternalForm());
			primaryStage.getIcons().add(new Image("logo.png"));
		primaryStage.setTitle("e x o V i s i x | Smart & Intelligent Computer
Vision Solution ");


			primaryStage.setScene(scene);
			primaryStage.show();
		} catch(Exception e) {
			e.printStackTrace();
		}
	}


	public static void main(String[] args) {
		launch(args);
```

```
        }
}
```

2) FaceDetector.java

```
package application;

import application.FaceRecognizer;
import java.awt.BasicStroke;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.sql.SQLException;
import java.time.Instant;
import java.util.ArrayList;

import javax.imageio.ImageIO;

import org.bytedeco.javacpp.FlyCapture2.ImageMetadata;
import org.bytedeco.javacpp.Loader;
import org.bytedeco.javacpp.opencv_objdetect;
import org.bytedeco.javacpp.helper.opencv_core;
```

```
import org.bytedeco.javacpp.opencv_core.Mat;
import org.bytedeco.javacv.CanvasFrame;
import org.bytedeco.javacv.Frame;
import org.bytedeco.javacv.FrameGrabber;
import org.bytedeco.javacv.Java2DFrameConverter;
import org.bytedeco.javacv.OpenCVFrameConverter;
import org.bytedeco.javacv.OpenCVFrameGrabber;
import static org.bytedeco.javacpp.opencv_core.*;
import static org.bytedeco.javacpp.opencv_imgproc.*;
import static org.bytedeco.javacpp.opencv_imgcodecs.*;
import static org.bytedeco.javacpp.opencv_objdetect.*;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.embed.swing.SwingFXUtils;
import javafx.fxml.FXML;
import javafx.scene.chart.PieChart.Data;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.image.ImageView;
import javafx.scene.image.WritableImage;
import application.Database;
import application.MotionDetector;
import application.ColoredObjectTracker;
import application.SquareDetector;
```

```
public class FaceDetector implements Runnable {


        Database database = new Database();
        ArrayList<String> user;


        FaceRecognizer faceRecognizer = new FaceRecognizer();
        MotionDetector motionDetector = new MotionDetector();
        OpenCVFrameConverter.ToIplImage grabberConverter = new
OpenCVFrameConverter.ToIplImage();
        Java2DFrameConverter paintConverter = new Java2DFrameConverter();
        ArrayList<String> output = new ArrayList<String>();


        @FXML
        public Label ll;
        private Exception exception = null;


        private int count = 0;
        public String classiferName;
        public File classifierFile;



        public boolean saveFace = false;
        public boolean isRecFace = false;
        public boolean isOutput = false;
        public boolean isOcrMode = false;
        public boolean isMotion = false;
        public boolean isEyeDetection = false;
```

```
        public boolean isSmile = false;
        public boolean isUpperBody = false;
        public boolean isFullBody = false;
        private boolean stop = false;

        private CvHaarClassifierCascade classifier = null;
        private CvHaarClassifierCascade classifierEye = null;
        private CvHaarClassifierCascade classifierSideFace = null;
        private CvHaarClassifierCascade classifierUpperBody = null;
        private CvHaarClassifierCascade classifierFullBody = null;
        private CvHaarClassifierCascade classifierSmile = null;
        private CvHaarClassifierCascade classifierEyeglass = null;

        public CvMemStorage storage = null;
        private FrameGrabber grabber = null;
        private IplImage grabbedImage = null, temp, temp2, grayImage = null,
smallImage = null;
        public ImageView frames2;
        public ImageView frames;

        private CvSeq faces = null;
        private CvSeq eyes = null;
        private CvSeq smile = null;
        private CvSeq upperBody = null;
        private CvSeq sideface = null;
```

```
private CvSeq fullBody = null;



int recogniseCode;
public int code;
public int reg;
public int age;

public String fname; //first name
public String Lname; //last name
public String sec; //section
public String name;

public void init() {
        faceRecognizer.init();

        setClassifier("haar/haarcascade_frontalface_alt.xml");
        setClassifierEye("haar/haarcascade_eye.xml");

setClassifierEyeGlass("haar/haarcascade_eye_tree_eyeglasses.xml");
        setClassifierSideFace("haar/haarcascade_profileface.xml");
        setClassifierFullBody("haar/haarcascade_fullbody.xml");
        setClassifierUpperBody("haar/haarcascade_upperbody.xml");
        setClassifierSmile("haar/haarcascade_smile.xml");

}
```

```
        public void start() {
                try {
                        new Thread(this).start();
                } catch (Exception e) {
                        if (exception == null) {
                                exception = e;

                        }
                }
        }

        public void run() {
                try {
                        try {
                                grabber = OpenCVFrameGrabber.createDefault(0);
//parameter 0 default camera , 1 for secondary

                                grabber.setImageWidth(700);
                                grabber.setImageHeight(700);
                                grabber.start();

                                grabbedImage =
grabberConverter.convert(grabber.grab());

                                storage = CvMemStorage.create();
                        } catch (Exception e) {
```

```
                    if (grabber != null)
                        grabber.release();
                    grabber = new OpenCVFrameGrabber(0);
                    grabber.setImageWidth(700);
                    grabber.setImageHeight(700);
                    grabber.start();
                    grabbedImage =
grabberConverter.convert(grabber.grab());

                }
                int count = 15;
                grayImage = cvCreateImage(cvGetSize(grabbedImage), 8, 1);
//converting image to grayscale

                //reducing the size of the image to speed up the processing
                smallImage = cvCreateImage(cvSize(grabbedImage.width() /
4, grabbedImage.height() / 4), 8, 1);

                stop = false;

                while (!stop && (grabbedImage =
grabberConverter.convert(grabber.grab())) != null) {

                    Frame frame =
grabberConverter.convert(grabbedImage);
                    BufferedImage image =
paintConverter.getBufferedImage(frame, 2.2 / grabber.getGamma());
                    Graphics2D g2 = image.createGraphics();
```

```
if (faces == null) {
    cvClearMemStorage(storage);

    //creating a temporary image
    temp =
cvCreateImage(cvGetSize(grabbedImage), grabbedImage.depth(),
grabbedImage.nChannels());

    cvCopy(grabbedImage, temp);

    cvCvtColor(grabbedImage, grayImage,
CV_BGR2GRAY);
    cvResize(grayImage, smallImage,
CV_INTER_AREA);

    //cvHaarDetectObjects(image, cascade,
storage, scale_factor, min_neighbors, flags, min_size, max_size)
    faces = cvHaarDetectObjects(smallImage,
classifier, storage, 1.1, 3, CV_HAAR_DO_CANNY_PRUNING);
    //face detection

    CvPoint org = null;
    if (grabbedImage != null) {

        if (isEyeDetection) {          //eye
detection logic
            eyes =
cvHaarDetectObjects(smallImage, classifierEye, storage, 1.1, 3,
```

```
        CV_HAAR_DO_CANNY_PRUNING);

                                        if (eyes.total() == 0) {
                                                eyes =
cvHaarDetectObjects(smallImage, classifierEyeglass, storage, 1.1, 3,
        CV_HAAR_DO_CANNY_PRUNING);

                                        }

                                        printResult(eyes, eyes.total(), g2);

                                }

                                if (isFullBody) { //full body detection logic
                                        fullBody =
cvHaarDetectObjects(smallImage, classifierFullBody, storage, 1.1, 3,
        CV_HAAR_DO_CANNY_PRUNING);

                                        if (fullBody.total() > 0) {
                                                printResult(fullBody,
fullBody.total(), g2);
                                        }

                                }
```

```
                    if (isUpperBody) {
                        try {
                            upperBody =
cvHaarDetectObjects(smallImage, classifierUpperBody, storage, 1.1, 3,
    CV_HAAR_DO_CANNY_PRUNING);

                            if (upperBody.total() > 0) {
    printResult(upperBody, upperBody.total(), g2);
                            }

                        } catch (Exception e) {

                            e.printStackTrace();
                        }
                    }

                    if (isSmile) {
                        try {
                            smile =
cvHaarDetectObjects(smallImage, classifierSmile, storage, 1.1, 3,
    CV_HAAR_DO_CANNY_PRUNING);

                            if (smile != null) {
                                printResult(smile,
smile.total(), g2);
```

```
                }
            } catch (Exception e) {

                e.printStackTrace();
            }

        }

        if (isOcrMode) {
            try {

                OutputStream os = new
FileOutputStream("captures.png");
                ImageIO.write(image,
"PNG", os);
            } catch (IOException e) {

                e.printStackTrace();
            }
        }

        isOcrMode = false;

        if (faces.total() == 0) {
            faces =
cvHaarDetectObjects(smallImage, classifierSideFace, storage, 1.1, 3,
```

```
        CV_HAAR_DO_CANNY_PRUNING);

                                        }

                                        if (faces != null) {
                                                g2.setColor(Color.green);
                                                g2.setStroke(new BasicStroke(2));
                                                int total = faces.total();

                                                for (int i = 0; i < total; i++) {

                                                        //printing rectange box
where face detected frame by frame
                                                        CvRect r = new
CvRect(cvGetSeqElem(faces, i));
                                                        g2.drawRect((r.x() * 4), (r.y()
* 4), (r.width() * 4), (r.height() * 4));

                                                        CvRect re = new
CvRect((r.x() * 4), r.y() * 4, (r.width() * 4), r.height() * 4);

                                                        cvSetImageROI(temp, re);

                                                        // File f = new
File("captures.png");
```

```
org = new CvPoint(r.x(),
r.y());


if (isRecFace) {
        String
names="Unknown Person!";
        this.recogniseCode =
faceRecognizer.recognize(temp);

        //getting recognised
user from the database

        if(recogniseCode != -
1)
        {
            database.init();
            user = new
ArrayList<String>();
            user =
database.getUser(this.recogniseCode);
            this.output =
user;

            names =
user.get(1) + " " + user.get(2);
        }

        //printing recognised
person name into the frame
```

```
        g2.setColor(Color.WHITE);
                                                    g2.setFont(new
Font("Arial Black", Font.BOLD, 20));

                                                    g2.drawString(names,
(int) (r.x() * 6.5), r.y() * 4);

                                                }

                                                if (saveFace) { //saving
captured face to the disk
                                                    //keep it in mind that
face code should be unique to each person
                                                    String fName =
"faces/" + code + "-" + fname + "_" + Lname + "_" + count + ".jpg";
                                                    cvSaveImage(fName,
temp);
                                                    count++;

                                                }

                                            }
                                            this.saveFace = false;
                                            faces = null;
                                        }

                                        WritableImage showFrame =
SwingFXUtils.toFXImage(image, null);
```

```
                                        javafx.application.Platform.runLater(new
Runnable(){

                                                @Override
                                                 public void run() {
                                                frames.setImage(showFrame);
                                                 }
                                                 });

                                        if (isMotion) {
                                                new Thread(() -> {

                                                        try {

        motionDetector.init(grabbedImage, g2);

                                                        } catch
(InterruptedException ex) {
                                                        } catch (Exception e) {

                                                                e.printStackTrace();
                                                        }

                                                }).start();
```

```
                                        }
                                        isMotion = false;

                                }
                                cvReleaseImage(temp);
                        }

                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }
}

public void stop() {
        stop = true;

        grabbedImage = grayImage = smallImage = null;
        try {
                grabber.stop();
        } catch (org.bytedeco.javacv.FrameGrabber.Exception e) {

                e.printStackTrace();
```

```
            }
            try {
                grabber.release();
            } catch (org.bytedeco.javacv.FrameGrabber.Exception e) {

                e.printStackTrace();
            }
            grabber = null;
        }

        public void setClassifier(String name) {

            try {

                setClassiferName(name);
                classifierFile = Loader.extractResource(classiferName, null,
"classifier", ".xml");

                if (classifierFile == null || classifierFile.length() <= 0) {
                    throw new IOException("Could not extract \"" +
classiferName + "\" from Java resources.");
                }

                // Preload the opencv_objdetect module to work around a
known bug.
                Loader.load(opencv_objdetect.class);
```

```
                classifier = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath()));
                classifierFile.delete();
                if (classifier.isNull()) {
                        throw new IOException("Could not load the classifier
file.");
                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }

    }

    public void setClassifierEye(String name) {

        try {

                classiferName = name;
                classifierFile = Loader.extractResource(classiferName, null,
"classifier", ".xml");

                if (classifierFile == null || classifierFile.length() <= 0) {
```

```
                        throw new IOException("Could not extract \"" +
classiferName + "\" from Java resources.");
                    }

                    // Preload the opencv_objdetect module to work around a
known bug.
                    Loader.load(opencv_objdetect.class);
                    classifierEye = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath()));
                    classifierFile.delete();
                    if (classifier.isNull()) {
                        throw new IOException("Could not load the classifier
file.");
                    }

                } catch (Exception e) {
                    if (exception == null) {
                        exception = e;

                    }
                }

            }

            public void setClassifierSmile(String name) {

                try {
```

```
                setClassiferName(name);
                classifierFile = Loader.extractResource(classiferName, null,
"classifier", ".xml");

                if (classifierFile == null || classifierFile.length() <= 0) {
                        throw new IOException("Could not extract \"" +
classiferName + "\" from Java resources.");
                }

                // Preload the opencv_objdetect module to work around a
known bug.
                Loader.load(opencv_objdetect.class);
                classifierSmile = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath()));
                classifierFile.delete();
                if (classifier.isNull()) {
                        throw new IOException("Could not load the classifier
file.");
                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }
```

```
	}

	public void printResult(CvSeq data, int total, Graphics2D g2) {
		for (int j = 0; j < total; j++) {
			CvRect eye = new CvRect(cvGetSeqElem(eyes, j));

			g2.drawOval((eye.x() * 4), (eye.y() * 4), (eye.width() * 4),
(eye.height() * 4));

		}
	}

	public void setClassifierSideFace(String name) {

		try {

			classiferName = name;
			classifierFile = Loader.extractResource(classiferName, null,
"classifier", ".xml");

			if (classifierFile == null || classifierFile.length() <= 0) {
				throw new IOException("Could not extract \"" +
classiferName + "\" from Java resources.");
			}

			// Preload the opencv_objdetect module to work around a
known bug.
```

```
                    Loader.load(opencv_objdetect.class);
                    classifierSideFace = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath()));
                    classifierFile.delete();
                    if (classifier.isNull()) {
                        throw new IOException("Could not load the classifier
file.");
                    }

                } catch (Exception e) {
                    if (exception == null) {
                        exception = e;

                    }
                }

        }

        public void setClassifierFullBody(String name) {

                try {

                    setClassiferName(name);
                    classifierFile = Loader.extractResource(classiferName, null,
"classifier", ".xml");

                    if (classifierFile == null || classifierFile.length() <= 0) {
```

```
                        throw new IOException("Could not extract \"" +
classiferName + "\" from Java resources.");
                    }

                    // Preload the opencv_objdetect module to work around a
known bug.
                    Loader.load(opencv_objdetect.class);
                    classifierFullBody = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath()));
                    classifierFile.delete();
                    if (classifier.isNull()) {
                        throw new IOException("Could not load the classifier
file.");
                    }

                } catch (Exception e) {
                    if (exception == null) {
                        exception = e;

                    }
                }

            }

            public void setClassifierEyeGlass(String name) {

                try {
```

```
                setClassiferName(name);
                classifierFile = Loader.extractResource(classiferName, null,
"classifier", ".xml");

                if (classifierFile == null || classifierFile.length() <= 0) {
                        throw new IOException("Could not extract \"" +
classiferName + "\" from Java resources.");
                }

                // Preload the opencv_objdetect module to work around a
known bug.
                Loader.load(opencv_objdetect.class);
                classifierEyeglass = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath()));
                classifierFile.delete();
                if (classifier.isNull()) {
                        throw new IOException("Could not load the classifier
file.");
                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }
```

```
	}

	public void setClassifierUpperBody(String name) {

		try {

			classiferName = name;
			classifierFile = Loader.extractResource(classiferName, null,
"classifier", ".xml");

			if (classifierFile == null || classifierFile.length() <= 0) {
				throw new IOException("Could not extract \"" +
classiferName + "\" from Java resources.");
			}

			// Preload the opencv_objdetect module to work around a
known bug.
			Loader.load(opencv_objdetect.class);
			classifierUpperBody = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath()));
			classifierFile.delete();
			if (classifier.isNull()) {
				throw new IOException("Could not load the classifier
file.");
			}

		} catch (Exception e) {
			if (exception == null) {
```

```
                    exception = e;

                }
            }

}

public String getClassiferName() {
        return classiferName;
}

public void setClassiferName(String classiferName) {
        this.classiferName = classiferName;
}

public void setFrames2(ImageView frames2) {
        this.frames2 = frames2;
}

public void setSmile(boolean isSmile) {
        this.isSmile = isSmile;
}

public void setUpperBody(boolean isUpperBody) {
        this.isUpperBody = isUpperBody;
}
```

```
public void setFullBody(boolean isFullBody) {
	this.isFullBody = isFullBody;
}

public boolean isEyeDetection() {

	return isEyeDetection;
}

public void setEyeDetection(boolean isEyeDetection) {
	this.isEyeDetection = isEyeDetection;
}

public boolean getOcrMode() {
	return isOcrMode;
}

public void setOcrMode(boolean isOcrMode) {
	this.isOcrMode = isOcrMode;
}

public void destroy() {
}

public boolean isMotion() {
```

```
        return isMotion;
}

public void setMotion(boolean isMotion) {
        this.isMotion = isMotion;
}

public ArrayList<String> getOutput() {
        return output;
}

public void clearOutput() {
        this.output.clear();
}

public void setOutput(ArrayList<String> output) {
        this.output = output;
}

public int getRecogniseCode() {
        return recogniseCode;
}

public void setRecogniseCode(int recogniseCode) {
        this.recogniseCode = recogniseCode;
}
```

```
public int getCode() {
	return code;
}

public void setCode(int code) {
	this.code = code;
}

public String getFname() {
	return fname;
}

public void setFname(String fname) {
	this.fname = fname;
}

public String getLname() {
	return Lname;
}

public void setLname(String lname) {
	Lname = lname;
}

public int getReg() {
```

```
        return reg;
}

public void setReg(int reg) {
        this.reg = reg;
}

public int getAge() {
        return age;
}

public void setAge(int age) {
        this.age = age;
}

public String getSec() {
        return sec;
}

public void setSec(String sec) {
        this.sec = sec;
}

public void setFrame(ImageView frame) {
        this.frames = frame;
}
```

```
    public void setSaveFace(Boolean f) {
        this.saveFace = f;
    }

    public Boolean getIsRecFace() {
        return isRecFace;
    }

    public void setIsRecFace(Boolean isRecFace) {
        this.isRecFace = isRecFace;
    }

}
```

3) DataBase.java

```
package application;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

class Database {
	public int code;

	public String fname;
	public String Lname;
	public int reg;
	public int age;
	public String sec;

	public final String Database_name = "ghosteye";
	public final String Database_user = "root";
	public final String Database_pass = "";

	public Connection con;

	public boolean init() throws SQLException {
```

```
            try {
                Class.forName("com.mysql.jdbc.Driver");

                try {
                    this.con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/" +
Database_name, Database_user,
                                                Database_pass);
                } catch (SQLException e) {

                    System.out.println("Error: Database Connection
Failed ! Please check the connection Setting");

                    return false;

                }

            } catch (ClassNotFoundException e) {

                e.printStackTrace();

                return false;
            }

            return true;
        }
```

```
	public void insert() {
		String sql = "INSERT INTO face_bio (code, first_name, last_name,
reg, age , section) VALUES (?, ?, ?, ?,?,?)";

		PreparedStatement statement = null;
		try {
			statement = con.prepareStatement(sql);
		} catch (SQLException e1) {
			// TODO Auto-generated catch block
			e1.printStackTrace();
		}

		try {

			statement.setInt(1, this.code);
			statement.setString(2, this.fname);

			statement.setString(3, this.Lname);
			statement.setInt(4, this.reg);
			statement.setInt(5, this.age);
			statement.setString(6, this.sec);

			int rowsInserted = statement.executeUpdate();
			if (rowsInserted > 0) {
				System.out.println("A new face data was inserted
successfully!");
			}
```

```
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}

	public ArrayList<String> getUser(int inCode) throws SQLException {

		ArrayList<String> user = new ArrayList<String>();

		try {

			Database app = new Database();

			String sql = "select * from face_bio where code=" + inCode
+ " limit 1";

			Statement s = con.createStatement();

			ResultSet rs = s.executeQuery(sql);

			while (rs.next()) {

				/*
				 * app.setCode(rs.getInt(2));
app.setFname(rs.getString(3));
```

```
                                * app.setLname(rs.getString(4));
app.setReg(rs.getInt(5));
                                * app.setAge(rs.getInt(6));
app.setSec(rs.getString(7));
                                */

                                user.add(0, Integer.toString(rs.getInt(2)));
                                user.add(1, rs.getString(3));
                                user.add(2, rs.getString(4));
                                user.add(3, Integer.toString(rs.getInt(5)));
                                user.add(4, Integer.toString(rs.getInt(6)));
                                user.add(5, rs.getString(7));

                                /*
                                 * System.out.println(app.getCode());
                                 * System.out.println(app.getFname());
                                 * System.out.println(app.getLname());
                                 * System.out.println(app.getReg());
                                 * System.out.println(app.getAge());
                                 * System.out.println(app.getSec());
                                 */

                                // nString="Name:" + rs.getString(3)+"
"+rs.getString(4) +
                                // "\nReg:" + app.getReg() +"\nAge:"+app.getAge()
+"\nSection:"
                                // +app.getSec() ;
```

```
                    // System.out.println(nString);
                }

                con.close(); // closing connection
        } catch (Exception e) {
                e.getStackTrace();
        }
        return user;
}

public void db_close() throws SQLException
{
        try {
                con.close();
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}


public int getCode() {
        return code;
}
```

```
public void setCode(int code) {
        this.code = code;
}

public String getFname() {
        return fname;
}

public void setFname(String fname) {
        this.fname = fname;
}

public String getLname() {
        return Lname;
}

public void setLname(String lname) {
        Lname = lname;
}

public int getReg() {
        return reg;
}

public void setReg(int reg) {
        this.reg = reg;
```

```
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	public String getSec() {
		return sec;
	}

	public void setSec(String sec) {
		this.sec = sec;
	}
}
```