Eric Wilson
Project 8

In this particular example, I technically fetch records from the database on a separate thread. It still waits for the data to be loaded, though. My reasoning is this: As we discussed in class when I presented my example, it's a bad idea to store multiple records in the program because the database could easily change while we're holding on to the data and we'd never know until we wrote to the DB and messed everything up. The only way I can think of to retrieve data concurrently would be buffer the data and load images that we don't plan to display yet. This is in conflict with the problems we talked about.

PhotoViewerController.java

```
/**
 * Created by EW043872 on 10/4/2015.
 */

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.*;

import javax.imageio.ImageIO;
import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;

public class PhotoViewerController implements ActionListener {
    int currentImage = 0;

    static PhotoViewerLayout photoView = new PhotoViewerLayout();
    static PhotoCollection collection = new PhotoCollection();
    int maxImages = collection.getCollectionSize();
    private final JFileChooser fc = new JFileChooser();

    public PhotoViewerController(){
        readCollection();
        maxImages = collection.getCollectionSize();
        if(maxImages <= 0){
            photoView.disableNext();
            photoView.disablePrev();
        }
        if(maxImages == 1){
            photoView.disableNext();
        }
        photoView.updatePhotoCount(maxImages);
        photoView.updatePhotoNum(currentImage + 1);
        System.out.print(collection.getCollectionSize());
        maxImages = collection.getCollectionSize();
        photoView.disablePrev();
        if (currentImage > 0) {
            photoView.disableNext();
            photoView.disableDelete();
        }
        else if (collection.getCollectionSize() > 0){
            currentImage = 1;
            collection.setCurrentPhotoNum(currentImage);
            collection.run();
            try {
                collection.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            photoView.updateDisplayPhoto(collection.getPhoto());
            photoView.setDescField(collection.getDescription());
            photoView.setDateField(collection.getDate());
        }


        photoView.deleteAddActionListener(new ActionListener() {        //delete
action
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                collection.deletePhoto(currentImage);
                maxImages--;
                photoView.updatePhotoCount(maxImages);
                if(currentImage == maxImages + 1 && maxImages > 0){ //deleting
last image in list
                    currentImage--;
                    photoView.updatePhotoNum(currentImage);
                    if(currentImage == 1){
                        photoView.disablePrev();
                    }
                    collection.setCurrentPhotoNum(currentImage);
                    collection.run();
                    try {
                        collection.join();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    photoView.updateDisplayPhoto(collection.getPhoto());
                    photoView.setDescField(collection.getDescription());
                    photoView.setDateField(collection.getDate());
                    photoView.disableNext();
                }
                else if (maxImages == 0){ //deleting last image altogether
                    photoView.disablePrev();
                    photoView.disableNext();
                    photoView.disableDelete();
                }
                else {
                    collection.setCurrentPhotoNum(currentImage);
                    collection.run();
                    try {
                        collection.join();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    photoView.updateDisplayPhoto(collection.getPhoto());
                    photoView.setDescField(collection.getDescription());
                    photoView.setDateField(collection.getDate());
                }
                photoView.updatePhotoNum(currentImage);
                /*if (currentImage == maxImages - 1 && maxImages > 1) {

photoView.updateDisplayPhoto(collection.getPhoto(currentImage));
                    maxImages = collection.getCollectionSize();
                    photoView.updatePhotoCount(maxImages);
                    photoView.updatePhotoNum(currentImage + 1);
                } else if (collection.getCollectionSize() == 0) { //if we're
deleting the last photo,
                    photoView.updateDisplayPhoto(new Photograph());
                    photoView.disableDelete();
                    maxImages = collection.getCollectionSize();
                    photoView.updatePhotoCount(maxImages);
                    photoView.disableNext();
                    photoView.disablePrev();
                    photoView.updatePhotoNum(currentImage + 1);
                } else {

photoView.updateDisplayPhoto(collection.getPhoto(currentImage));
                    maxImages = collection.getCollectionSize();
                    photoView.updatePhotoCount(maxImages);
                    photoView.updatePhotoNum(currentImage + 1);
                }
                if (currentImage == 0) {
                    photoView.disablePrev();
                    if (maxImages == 1)
                        photoView.disableNext();
                }*/

            }
        });

        photoView.addAddActionListener(new ActionListener() {           //add
action
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                JFileChooser chooser = new JFileChooser();
                FileNameExtensionFilter filter = new FileNameExtensionFilter(
                        "Images", "jpg", "gif", "bmp", "png");
                fc.setFileFilter(filter);
                int returnVal = fc.showOpenDialog(fc);
                BufferedImage img = null;
                if (returnVal == JFileChooser.APPROVE_OPTION) {
                    File file = fc.getSelectedFile();
                    try {
                        img = ImageIO.read(file);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    ImageIcon image = new ImageIcon(img);
                    collection.addPhoto(file, currentImage + 1);
                    maxImages = collection.getCollectionSize();
                    photoView.updatePhotoCount(maxImages);
                }
                photoView.enableDelete();
                photoView.enableNext();

                if(maxImages == 1){
                    currentImage = 1;
                    photoView.disableNext();
                    photoView.disablePrev();
                    collection.setCurrentPhotoNum(currentImage);
                    collection.run();
                    try {
                        collection.join();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    photoView.updateDisplayPhoto(collection.getPhoto());
                    photoView.setDescField(collection.getDescription());
                    photoView.setDateField(collection.getDate());
                }
            }
        });

        photoView.saveAddActionListener(new ActionListener() {          //save
action
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                collection.saveData(photoView.getDescField(),
photoView.getDateField());
            }
        });

        photoView.nextAddActionListener(new ActionListener() {          //next
button
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                currentImage++;
                photoView.enablePrev();
                if (currentImage == maxImages) photoView.disableNext();
                collection.setCurrentPhotoNum(currentImage);
                collection.run();
                try {
                    collection.join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                photoView.updateDisplayPhoto(collection.getPhoto());
                photoView.setDescField(collection.getDescription());
                photoView.setDateField(collection.getDate());
                photoView.updatePhotoCount(maxImages);
                photoView.updatePhotoNum(currentImage);

                System.out.print(photoView.getDescField());
            }


        });

        photoView.prevAddActionListener(new ActionListener() {         //prev
button
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                currentImage--;
                photoView.enableNext();
                if (currentImage == 1) photoView.disablePrev();
                collection.setCurrentPhotoNum(currentImage);
                collection.run();
                try {
                    collection.join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                photoView.updateDisplayPhoto(collection.getPhoto());
                photoView.setDescField(collection.getDescription());
                photoView.setDateField(collection.getDate());
                photoView.updatePhotoCount(maxImages);
                photoView.updatePhotoNum(currentImage);

                System.out.print(photoView.getDescField());
            }
        });

        photoView.browseAddActionListener(new ActionListener() {        //browse
menu
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                photoView.disableMaint();
            }
        });

        photoView.maintAddActionListener(new ActionListener() {          //maint
menu
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                photoView.enableMaint();
            }
        });

        photoView.goToAddActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                int ind = currentImage;

                try {
                    ind = photoView.getPictureNumber();
                } catch (NumberFormatException n){
                    photoView.updatePhotoNum(currentImage);
                }

                if (ind <= maxImages && ind > 0){
                    //change image and data
                    currentImage = ind;
                    collection.setCurrentPhotoNum(currentImage);
                    collection.run();
                    try {
                        collection.join();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    photoView.updateDisplayPhoto(collection.getPhoto());
                    photoView.setDescField(collection.getDescription());
                    photoView.setDateField(collection.getDate());
                    photoView.updatePhotoNum(currentImage);

                    //Manage buttons and stuff.
                    if (ind == 1){
                        photoView.disablePrev();
                        photoView.enableNext();
                    }
                    else if (ind == maxImages){
                        photoView.disableNext();
                        photoView.enablePrev();
                    }
                    else{
```

```java
                        photoView.enablePrev();
                        photoView.enableNext();
                    }
                }
                else{
                    photoView.updatePhotoNum(currentImage);
                }

            }
        });

    }

    public JFrame getFrame(){
        return photoView;
    }

    private void readCollection(){

        /*try {
            FileInputStream fileIn = new FileInputStream("collection.ser");
            ObjectInputStream ObjIn = new ObjectInputStream(fileIn);
            collection = (PhotoCollection)ObjIn.readObject();
        } catch (FileNotFoundException e) {
            System.out.print("FILE IN NOT FOUND!");
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }*/
    }

    /*private void writeCollection(){
        try {
            FileOutputStream fileOut = new FileOutputStream("collection.ser");
            ObjectOutputStream objOut = new ObjectOutputStream(fileOut);
            objOut.writeObject(collection);
            objOut.close();
            System.out.print("Collection serialized to \"collection.ser\"");
        } catch (IOException i) {
            System.out.print("FILE OUT NOT FOUND\n");
        }
    }*/
    @Override
    public void actionPerformed(ActionEvent actionEvent) {

    }
    public static void main(String[] args) {
        JFrame frame = new PhotoViewerController().getFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```

PhotoViewerLayout.java

```java
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import javax.swing.*;

public class PhotoViewerLayout extends JFrame{

    JButton deleteButton = new JButton("Delete");
    JButton saveButton = new JButton("Save Data");
    JButton addButton = new JButton("Add Photo");
    JButton prevButton = new JButton("<prev");
    JButton nextButton = new JButton("next>");
    ImageIcon image = new ImageIcon();
    JLabel imageLabel = new JLabel("", SwingConstants.CENTER);
    JMenuItem browseMode = new JMenuItem("Browse Mode");
    JMenuItem maintMode = new JMenuItem("Maintenance Mode");
    JTextField pictureNumberTextField = new JTextField("3",4);
    JLabel pictureCountLabel = new JLabel(" of 3");
    JTextField dateTextField = new JTextField("1/1/2014");
    JTextArea descriptionTextArea = new JTextArea("jklkjhkl",4,20);
    JButton goToButton = new JButton("Go To");


    public PhotoViewerLayout() {

        Container contentPane = getContentPane();
        descriptionTextArea.setEditable(true);
        JScrollPane scrollPane = new JScrollPane(imageLabel);

        JMenuBar menuBar = new JMenuBar();                                     //menu
        setJMenuBar(menuBar);
        JMenu viewMenu = new JMenu("View");
        menuBar.add(viewMenu);

        viewMenu.add(browseMode);

        viewMenu.add(maintMode);


        imageLabel.setIcon(image);

        contentPane.add(scrollPane, BorderLayout.CENTER);

        JPanel controlPane = new JPanel();
        controlPane.setLayout(new BoxLayout(controlPane, BoxLayout.PAGE_AXIS));

        JPanel descriptionPane = new JPanel();
        descriptionPane.setLayout(new FlowLayout(FlowLayout.LEFT));

        JLabel descriptionLabel = new JLabel("Description:");
        descriptionPane.add(descriptionLabel);
        descriptionPane.add(descriptionTextArea);

        JPanel datePane = new JPanel();

        JLabel dateLabel = new JLabel("Date:");
        dateLabel.setPreferredSize(new Dimension(descriptionLabel.getPreferredSize().width,
dateLabel.getPreferredSize().height));

        datePane.add(dateLabel);
        datePane.add(dateTextField);
        JPanel buttonPane = new JPanel();

        buttonPane.add(deleteButton);
        buttonPane.add(saveButton);
        buttonPane.add(addButton);


        JPanel leftRightPane = new JPanel();
        leftRightPane.setLayout(new BorderLayout());
        leftRightPane.add(datePane,BorderLayout.WEST);
        leftRightPane.add(buttonPane,BorderLayout.EAST);


        JPanel southButtonPanel = new JPanel();



        southButtonPanel.add(pictureNumberTextField);
        southButtonPanel.add(pictureCountLabel);
        southButtonPanel.add(prevButton);
        southButtonPanel.add(nextButton);
        southButtonPanel.add(goToButton);
        FlowLayout flowLayout = (FlowLayout) southButtonPanel.getLayout();
        flowLayout.setAlignment(FlowLayout.LEFT);


        controlPane.add(descriptionPane);
        controlPane.add(leftRightPane);
        controlPane.add(southButtonPanel);

        contentPane.add(controlPane, BorderLayout.SOUTH); // Or PAGE_END
    }

    public void updateDisplayPhoto (Photograph photo){
        image = photo.getImage();
        imageLabel.setIcon(image);

    }
    public void updateDisplayPhotoTest (ImageIcon photo){
        image = photo;
        imageLabel.setIcon(image);

    }

    public void updateDescription(String desc){
        descriptionTextArea.setText(desc);
    }

    public void disablePrev(){prevButton.setEnabled(false);}

    public void enablePrev(){prevButton.setEnabled(true);}

    public void disableNext(){nextButton.setEnabled(false);}

    public void enableNext(){nextButton.setEnabled(true);}

    public void disableDelete() {deleteButton.setEnabled(false);}

    public void enableDelete() {deleteButton.setEnabled(true);}

    public void enableMaint(){
        deleteButton.setEnabled(true);
        addButton.setEnabled(true);
        saveButton.setEnabled(true);
    }

    public void disableMaint(){
        deleteButton.setEnabled(false);
        addButton.setEnabled(false);
        saveButton.setEnabled(false);
    }

    public void updatePhotoCount(int count) {
        pictureCountLabel.setText(Integer.toString(count));
    }

    public void updatePhotoNum(int count) {
        pictureNumberTextField.setText(Integer.toString(count));
    }

    public void deleteAddActionListener(ActionListener al){
        deleteButton.addActionListener(al);
    }

    public void addAddActionListener(ActionListener al){ addButton.addActionListener(al); }

    public void saveAddActionListener(ActionListener al){ saveButton.addActionListener(al); }

    public void prevAddActionListener(ActionListener al){ prevButton.addActionListener(al); }

    public void nextAddActionListener(ActionListener al){ nextButton.addActionListener(al); }

    public void browseAddActionListener(ActionListener al){ browseMode.addActionListener(al);}

    public void maintAddActionListener(ActionListener al){ maintMode.addActionListener(al);}

    public void goToAddActionListener(ActionListener al){ goToButton.addActionListener(al);}

    public String getDescField(){
        return descriptionTextArea.getText();
    }

    public void setDescField(String string){
        descriptionTextArea.setText(string);
    }

    public void setDateField(String newDate){dateTextField.setText(newDate);System.out.print("NEWDATE: " +
newDate + "\n");}

    public String getDateField(){return dateTextField.getText();}

    public int getPictureNumber() throws NumberFormatException{
        String fieldValue = pictureNumberTextField.getText();

        return Integer.parseInt(fieldValue);
    }
}
```

PhotoCollection.java

```
import javax.swing.*;
import java.io.*;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 * Created by EW043872 on 10/4/2015.
 */
public class PhotoCollection extends Thread implements Serializable {

    public ArrayList<Photograph> photos = new ArrayList<>();
    Photograph currentPhoto = null;
    int currentPhotoNum = 0;
    DataBaseGateWay DBGW;

    @Override
    public void run(){
        try {
            currentPhoto = DBGW.DBgetRecord(currentPhotoNum);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public PhotoCollection(){
        try {
            DBGW = new DataBaseGateWay();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        //DEBUG
        System.out.print("DATABASE COUNT " + Integer.toString(getCollectionSize()) + "\n");

        try {
            if(getCollectionSize() > 0)
            currentPhoto = DBGW.DBgetRecord(1);
        } catch (Exception e) {
            System.out.print("Error getting first image.");
            e.printStackTrace();
        }
    }



    public void addPhoto(File file, int index){
        String date = "";
```

```java
        String description = "";
        try {
            DBGW.DBcreateRecord(file, date,description, index);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void deletePhoto(int index){
        try {
            DBGW.DBdeleteRecord(index);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public Photograph getPhoto(){

        return currentPhoto;
    }

    public int getCollectionSize(){
        int count = 0;
        try {
            count = DBGW.countRows();
        } catch (SQLException e) {
            System.out.print("Error getting collection size.\n");
            e.printStackTrace();
        }
        return count;
    }

    public void setDescription(int index, String desc){
        photos.get(index).description = desc;
    }

    public String getDescription(){
        return currentPhoto.description;
    }

    public String getDate(){ return currentPhoto.date;}

    public void saveData (String description, String date){
        try {
            DBGW.updateImageData(description, date, currentPhoto.id);
        } catch (SQLException e) {
            System.out.print("Error updating image data.");
            e.printStackTrace();
        }

    }

    public void setCurrentPhotoNum(int num){
        currentPhotoNum = num;
    }
}
```

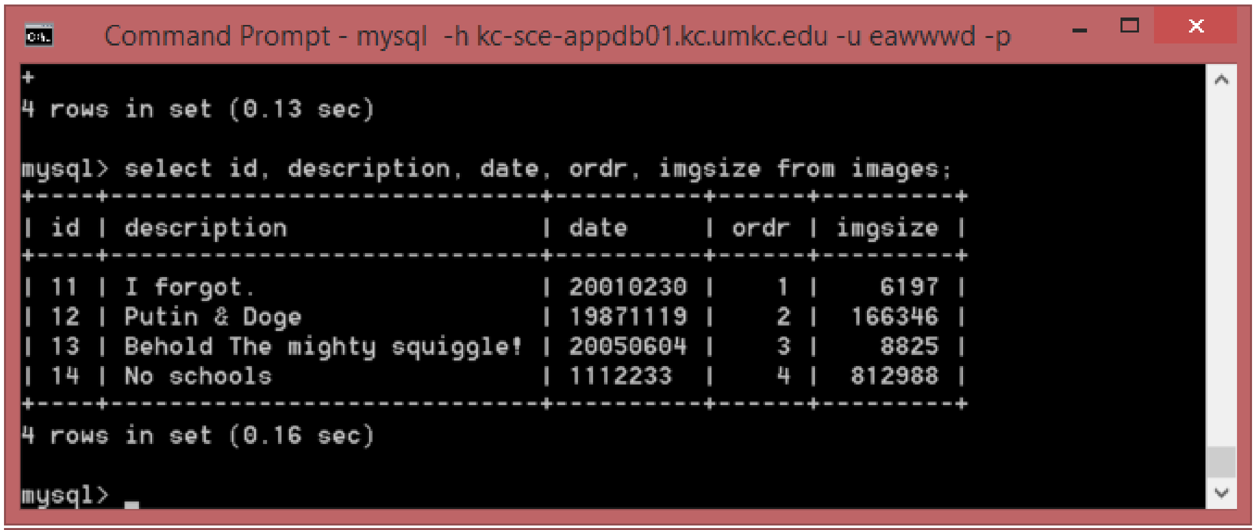
DatabaseGateway.java

```java
import javax.swing.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.sql.*;
```

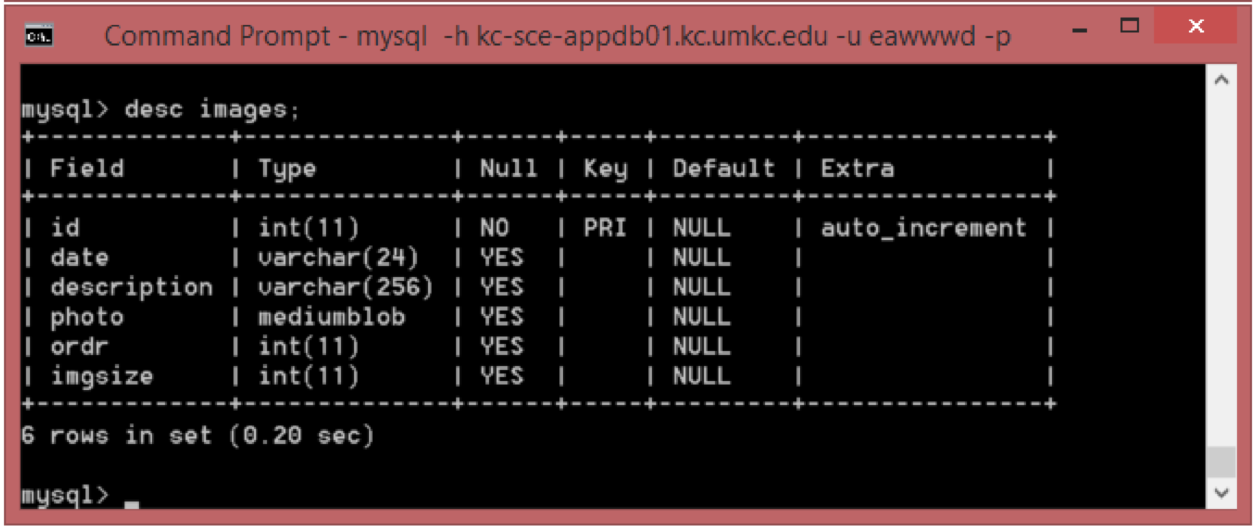
```java
/**
 * Created by EW043872 on 10/18/2015.
 */
public class DataBaseGateWay {
    private Connection con;
    private Statement stmt;
    String connectionString = "jdbc:mysql://kc-sce-appdb01.kc.umkc.edu/eawwwd";
    String userID = "eawwwd";
    String password = "ZDkWx5Dydgwe";

    DataBaseGateWay()throws SQLException{
        //initialize connection
        con = DriverManager.getConnection(connectionString, userID, password);
        //initialize statement
        stmt = con.createStatement();
        //create any needed tables
        createTables();

        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch(java.lang.ClassNotFoundException e) {
            System.out.println(e);
            System.exit(0);
        }
    }

    //creates necessary tables if they are not found in the database
    public void createTables() throws SQLException {
        //String sqlcmd = "USE eawwwd;";

        String createTable = "CREATE TABLE IF NOT EXISTS Images" + "(id INT NOT
NULL AUTO_INCREMENT,"
                + " date VARCHAR(24), description VARCHAR(256),"
                + " photo LONG VARBINARY NULL, ordr INT, PRIMARY KEY (id));";
        //stmt.execute(sqlcmd);
        stmt.execute(createTable);
    }

    public void DBcreateRecord(File file, String date, String description, int
index) throws Exception{
        String incrementOrder = "UPDATE images SET ordr = ordr + 1 WHERE ordr >= ?
";
        String sqlUpdate = " insert into images(date, description, photo, ordr,
imgsize) values (?, ?, ?, ?, ?)";
        PreparedStatement pstmt = null;
        //Increment all values of ordr in table greater than the index we're
trying to insert
        pstmt = con.prepareStatement(incrementOrder);
        pstmt.setInt(1, index);
        pstmt.executeUpdate();
        //File file = null;
        FileInputStream fileIn = null;

        pstmt = con.prepareStatement(sqlUpdate);

        //file = new File(filePath);
        fileIn = new FileInputStream(file);

        pstmt.setNString(1, date);
        pstmt.setNString(2, description);
        pstmt.setBlob(3, fileIn, (int) file.length());
        pstmt.setInt(4, index);
        pstmt.setInt(5, (int) file.length());
        pstmt.executeUpdate();
        fileIn.close();
        pstmt.close();
    }

    public Photograph DBgetRecord(int index) throws Exception{
        PreparedStatement ps = null;
        Photograph returnPhoto = new Photograph();
        ps = con.prepareStatement("select * from images where ordr = ? ;");
        ps.setInt(1, index);
        ResultSet rs = ps.executeQuery();
        rs.first();
        returnPhoto.setID(rs.getInt("id"));
        returnPhoto.setDate(rs.getString("date"));
        returnPhoto.setDescription(rs.getString("description"));

        InputStream in = rs.getBinaryStream("photo");
        int size = rs.getInt("imgsize");
        byte[] buffer = new byte[size];
        int currentLength = 0;
        int bytesRead = 0;
        while ((bytesRead = in.read(buffer, currentLength, size - currentLength))
!= -1) {
            currentLength = currentLength + bytesRead;
        }

        ImageIcon newIcon = new ImageIcon(buffer);
        returnPhoto.setImage(newIcon);
        ps.close();
        return returnPhoto;
    }

    public int countRows() throws SQLException{
        //count number of rows, expect only an int in rs
        ResultSet rs = stmt.executeQuery("select count(*) from images;");
        rs.first();
        int count = rs.getInt(1);
        return count;
    }

    public void updateImageData(String description, String date, int index)throws
SQLException{
        String updateSQL = "UPDATE images SET date = ?, description = ? WHERE id =
? ;";
        PreparedStatement pstmt = null;

        pstmt = con.prepareStatement(updateSQL);

        pstmt.setNString(1, date);
        pstmt.setNString(2, description);
        pstmt.setInt(3, index);

        pstmt.executeUpdate();
        pstmt.close();
    }
    public void DBdeleteRecord(int ind) throws SQLException{
        String deleteRecord = new String("DELETE FROM images WHERE ordr = ? ;");
        String updateOrder = new String("UPDATE images SET ordr = ordr - 1 WHERE 
ordr > ? ;");

        PreparedStatement pstmt = null;

        pstmt = con.prepareStatement(deleteRecord);
        pstmt.setInt(1, ind);
        pstmt.executeUpdate();

        pstmt = con.prepareStatement(updateOrder);
        pstmt.setInt(1, ind);
        pstmt.executeUpdate();

        pstmt.close();
    }
}
```

Photograph.java

```
import javax.swing.*;
import java.io.Serializable;

/**
 * Created by EW043872 on 10/6/2015.
 */
public class Photograph implements Serializable {
    ImageIcon image = new ImageIcon();
    String date = new String("");
    public String description = new String("");
    int id;

    Photograph(ImageIcon img){
        image = img;
    }

    Photograph(){}

    public void setImage(ImageIcon img){
        image = img;
    }

    public ImageIcon getImage(){
        return image;
    }

    public void setDate(String dat ){
        date = dat;
    }

    public String getDate(){
        return date;
    }

    public void setDescription(String desc){
        description = desc;
    }

    public String getDescription(){
        return description;
    }

    public void setID(int newID){id = newID;}

    public int getID(){return id;}

}
```


Command Prompt - mysql -h kc-sce-appdb01.kc.umkc.edu -u eawwwd -p

```
+
4 rows in set (0.13 sec)

mysql> select id, description, date, ordr, imgsize from images;
+----+----------------------------+----------+------+---------+
| id | description                | date     | ordr | imgsize |
+----+----------------------------+----------+------+---------+
| 11 | I forgot.                  | 20010230 |    1 |    6197 |
| 12 | Putin & Doge               | 19871119 |    2 |  166346 |
| 13 | Behold The mighty squiggle! | 20050604 |    3 |    8825 |
| 14 | No schools                 | 1112233  |    4 |  812988 |
+----+----------------------------+----------+------+---------+
4 rows in set (0.16 sec)

mysql> _
```


Command Prompt - mysql -h kc-sce-appdb01.kc.umkc.edu -u eawwwd -p

```
mysql> desc images;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| date        | varchar(24)  | YES  |     | NULL    |                |
| description | varchar(256) | YES  |     | NULL    |                |
| photo       | mediumblob   | YES  |     | NULL    |                |
| ordr        | int(11)      | YES  |     | NULL    |                |
| imgsize     | int(11)      | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
6 rows in set (0.20 sec)

mysql> _
```