```java
// Johnny Zielinski Create Project - Independent

import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Label;
import java.awt.TextArea;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Collections;
import java.util.Scanner;

import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;

public class SHCheckin extends Frame implements ActionListener, SerialPortEventListener
{
    private static final long serialVersionUID = 1L;
    private static Label lblInput;
    private static TextArea display;
    private static TextField tfInput;
    private static ArrayList<Student> athleteReq;
    private static ArrayList<Student> athletePres;
    private static String timeStamp = new
    SimpleDateFormat("MM-dd-yyyy").format(Calendar.getInstance().getTime()) +
    "AbsentList.txt";
    private static BufferedWriter writer = null;
    public static File attendance = new File("AbsentLists", timeStamp);
    private static InputStream in;
    private static SerialPort serialPort;


    private SHCheckin() throws IOException // main program that is called from main
    {
        setLayout(new FlowLayout());
        writer = new BufferedWriter(new FileWriter(attendance));

        lblInput = new Label("Enter your ID number: ");

        display = new TextArea(17, 40);
        tfInput = new TextField(20);

        display.setEditable(false);

        Font font = new Font(Font.SANS_SERIF, Font.PLAIN, 20); // bigger
        Font font2 = new Font(Font.SANS_SERIF, Font.PLAIN, 18); // smaller
        display.setFont(font);
        lblInput.setFont(font2);
        tfInput.setFont(font2);

        add(lblInput);
        add(tfInput);
        add(display);

        tfInput.addActionListener(this);

        setTitle("Study Hall Check-In");
        setSize(550, 550);
        setVisible(true);

        try
        {
        connect("COM3");
        }
        catch(Exception e)
        {
            System.out.println("ERROR: Couldn't connect to Scanner");
        }

        // Close on exit, calls write out to text file and then closes file.
        addWindowListener(new WindowAdapter(){
              public void windowClosing(WindowEvent we){
                try
                {
                writeOut();
                writer.close();
                in.close();
                serialPort.removeEventListener();
                }
                catch(Exception e)
                {
                    System.out.println("ERROR DURING CLOSING");
                }
                System.exit(0);
              }
            });

    }

    protected void connect( String portName ) throws Exception  // connects the scanner to the program through serial port emulation
    {
        CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier( portName );

        if( portIdentifier.isCurrentlyOwned() )
        {
            System.out.println( "Error: Port is currently in use" );
        }
        else
        {
            int timeout = 10000;
            CommPort commPort = portIdentifier.open( this.getClass().getName(), timeout );

            if(commPort instanceof SerialPort)
            {
                serialPort = ( SerialPort )commPort;
                serialPort.setSerialPortParams( 9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

                in = serialPort.getInputStream();
                serialPort.addEventListener(new SerialReader(in));
                serialPort.notifyOnDataAvailable(true);
            }
            else
            {
                System.out.println( "Error: Not a serial port." );
            }
        }
    }

    private void writeOut() // Writes the absent list out to time stamped file
    {
        ArrayList<Student> studentsAbsent = new ArrayList<Student>(athleteReq);
        studentsAbsent.removeAll(athletePres); // remove all present people.
        Collections.sort(studentsAbsent);

        for(int a = 0; a < studentsAbsent.size(); a++)
        {
                try
                {
                    writer.write(studentsAbsent.get(a).getName());
                    writer.newLine();
                }
                catch(Exception e)
                {
                    System.out.println("ERROR WRITING TO TEXT FILE");
                }
        }
    }

    public void actionPerformed(ActionEvent evt) // Checks for enter key pressed
    {
        String id = tfInput.getText();
        boolean found = false;

        for(int a = 0; a < athleteReq.size(); a++)
        {
            if(athleteReq.get(a).getId().equals(id))
            {
                display.append(athleteReq.get(a).toString()+"\n");
                athletePres.add(athleteReq.get(a));
                found = true;
            }
        }

        if(!found)
        {
            display.append("ID NOT FOUND \n");
        }

        tfInput.setText("");

    }

    public static void main(String[] args) throws IOException // reads in the required students and starts program
    {
        Scanner input = new Scanner(new File("resources", "athletes.txt"));

        athleteReq = new ArrayList<Student>(); // list of athletes required
        athletePres = new ArrayList<Student>();

        while(input.hasNextLine())
        {
            String[] line = input.nextLine().split(" ");

```

```java
                String id = line[line.length-1];

                String name = "";

                for(int i = 0; i < line.length-1; i++) // two part names
                {
                    name += line[i] + " ";
                }

                name = name.substring(0, name.length()-1); // cuts off final space

                Student kid = new Student(name, id);

                athleteReq.add(kid);
            }

            new SHCheckin();


            input.close();
        }



        protected static void scanned(String id) // handles the scanned id and finds it in athlete list
        {
            String subID = id.substring(7, id.length()-2);

            boolean found = false;

            for(int a = 0; a < athleteReq.size(); a++)
            {
                if(athleteReq.get(a).getId().equals(subID))
                {
                    display.append(athleteReq.get(a).toString()+"\n");
                    athletePres.add(athleteReq.get(a));
                    found = true;
                }
            }

            if(!found)
            {
                display.append("ID NOT FOUND \n");
            }
        }

        @Override
        public void serialEvent(SerialPortEvent evt) // simply here to make complier happy doesn't serve a purpose
        {


        }

    }
```