```
package org.usfirst.frc.team3243.robot;


import java.util.ArrayList;

import java.util.TimerTask;

import java.util.Timer;

import java.io.*;



public class Recorder implements java.io.Serializable {

	/**

	 * Class Written by Hunter Schmidt for the purpose of recording controller input

	 */

	ArrayList<Double> Data0 = new ArrayList<Double>();//creates 3 object-specific arraylists to
store input data from joystick

	ArrayList<Double> Data1 = new ArrayList<Double>();

	ArrayList<Double> Data2 = new ArrayList<Double>();

	ArrayList<Double> ElevData = new ArrayList<Double>();//creates arraylist to get joystick input
for elevator

	ArrayList<Double> GrabberData0 = new ArrayList<Double>();//creates arraylists to get input for
the grabber

	ArrayList<Double> GrabberData1 = new ArrayList<Double>();

	static transient Timer stopRecord= new Timer(); //creates timer object to stop recording after
15 seconds

	static transient int counter = Reader.getCounter();//sets value of recording counter to the last
recording value

	static transient int planNumber = 2;//number of recording to execute on playback

	static transient boolean isRead= false;//checks to see if the file was correctly read

	static transient boolean startRecord = false;//boolean to control when to record data

	static transient boolean writeToFile =false;//boolean to control when to write data to file

	public static boolean isRecording = false;//secondary boolean to control when to record data
	public static int playIncrement=0;//integer that helps in data retrieval

	public static boolean clearData = false;//boolean to control when to clear excess data

	public static boolean timerOn = false;//boolean to control when the timer starts

	transient InputManager IM = new InputManager();//creates instance of inputmanager class to
receive input


	private class recordingTimer extends TimerTask{//creates task to run after 15 seconds


		@Override

		public void run() {//runs when timer is up


			isRecording = false;//stops recording

			startRecord = false;//sets it to not record again

			writeToFile = true;//allows data to be written to file

			System.out.println("timer ran");//debug check to see if the timer worked

		}


	}


	public void getData(double[] drive, double[] elevator, double[] solenoid){//gets data from
joystick array


		IM.record();

		if (isRecording /*&& startRecord */){//starts recording if it is supposed to

		this.Data0.add(drive[0]);//records data to static arraylists

		this.Data1.add(drive[1]);

		this.Data2.add(drive[2]);

		this.ElevData.add(elevator[0]);

		this.GrabberData0.add(solenoid[0]);
```

```
                    this.GrabberData1.add(solenoid[1]);
                    //this.ElevData.add(array[3]);
                    if (timerOn){//starts timer if told to do so
                              stopRecord.schedule(new recordingTimer(), 15000);//schedules stop in 15
seconds
                              timerOn = false;//stops timer
                              System.out.println("timer started");//debug check to see if timer starts
                              }
                    }


          }




          public double[] playBackDrive(){//plays back recording
                    double[]playArray = new double[3];//creates data array to return and pass to motor
methods
                    if(playIncrement > this.Data0.size()-1){//if it keeps reading larger than the size for any
reason, this stops the robot
                              playArray[0]=0;
                              playArray[1]=0;
                              playArray[2]=0;


                    }else
                    {
                              playArray[0]=this.Data0.get(playIncrement);//sets array elements to saved ones
at the element of the number of loops recorded by playIncrement
                              playArray[1]=this.Data1.get(playIncrement);
                              playArray[2]=this.Data2.get(playIncrement);
                              ++playIncrement;//increments element of arraylist
                    }
                    return playArray;//returns array to pass to motor methods
                    }


            public double[] playBackElevator(){//plays back elevator data
                            double[]playArray = new double[1];//creates array to pass recorded data to
elevator methods
                            if(playIncrement > this.ElevData.size()-1){//if it keeps reading larger than the
size for any reason, this stops the robot
                                    playArray[0]=0;


                            }else
                            {
                                    playArray[0]=this.ElevData.get(playIncrement);//sets return array to
recorded data at playIncrement
                            }
                            return playArray;
                    }


            public double[] playBackGrabber(){//plays back grabber input from recording
                            double[]playArray = new double[2];//creates array to return grabber input to
motor methods
                            if(playIncrement > this.GrabberData0.size()-1){//if it keeps reading larger than
the size for any reason, this stops the robot
                                    playArray[0]=0;
                                    playArray[1]=0;



                            }else
                            {
```

```
                                        playArray[0]=this.GrabberData0.get(playIncrement);//sets array
elements to saved ones at number of loops

                                        playArray[1]=this.GrabberData1.get(playIncrement);

                              }

                              return playArray;//returns array to pass to motor methods

                    }

}
```

```
package org.usfirst.frc.team3243.robot;


import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.ObjectInputStream;


public class Reader {


	/*

	 * Class Written by Hunter Schmidt for the purpose of reading recorded controller input data
from a file

	 */

FileInputStream fileIn;//initializes a file input

ObjectInputStream in;//initializes a serialized object input

	public void readData(Recorder r){//method to read in data to a Recorder class instance r

		 Recorder reader= new Recorder();//creates new instance of recorder

		try{//in try-case because files are sometimes weird

		try

	    {

			r.Data0.clear();//clears recorder object data just in case there is garbage

	       r.Data1.clear();

	       r.Data2.clear();

	       r.ElevData.clear();

	       r.GrabberData0.clear();

	       r.GrabberData1.clear();

	       fileIn = new FileInputStream("/home/lvuser/auto/Recording " + Recorder.planNumber +
".JSON");//reads in file with # from Recorder class

	       in = new ObjectInputStream(fileIn);//reads in serialized object
```

```
                reader = (Recorder) in.readObject();//sets reader object to read in object

                in.close();//closes file input streams

                fileIn.close();

                r.Data0 = reader.Data0;//sets data of r to read in data

                            r.Data1 = reader.Data1;

                            r.Data2 = reader.Data2;

                            r.ElevData = reader.ElevData;

                            r.GrabberData0 = reader.GrabberData0;

                            r.GrabberData1 = reader.GrabberData1;

                           Recorder.isRead = true;//lets the robot know a recording has been successfully
loaded

              }catch(IOException i){}

                     catch(ClassNotFoundException c){}


                   }finally{

                           if(fileIn !=null){//makes sure to close streams just in case

                                    try {

                                             fileIn.close();

                                    } catch (IOException e) {

                                             // TODO Auto-generated catch block

                                             e.printStackTrace();

                                    }

                           }if(in !=null){

                                    try {

                                             in.close();

                                    } catch (IOException e) {

                                             // TODO Auto-generated catch block

                                             e.printStackTrace();

                                    }
```

```
                    }
                }


            }
            public static int getCounter(){//method to read in counter to prevent accidently overwriting data
                int reader=0;//creates an int to recieve the data


                try
              {
                   FileInputStream fileIn = new
FileInputStream("/home/lvuser/auto/Counter.JSON");//creates file input stream
                   ObjectInputStream in = new ObjectInputStream(fileIn);//creates stream to input serialized
object
                   reader = (int) in.readObject();//sets reader to read in data
                   in.close();//closes streams
                   fileIn.close();
              }catch(IOException i){ return 1;}//returns one if no other data can be found
                    catch(ClassNotFoundException c){return 1;}


                return reader;//returns read in data
            }
}
```

```
package org.usfirst.frc.team3243.robot;


import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

public class Writer implements java.io.Serializable{

        /**

         * Class written by Hunter Schmidt for purposes of writing recorded input to a file

         */

        private static final long serialVersionUID = 1L;

        int outputCounter = 0;

        FileOutputStream FileOut;//initializes fileoutput

        ObjectOutputStream fileout;//initializes object serialization

        public void writeData(Recorder r){//writes data to file

                try//in try-case in case it fails, as files are sometimes strange

          {



                        try {

                                FileOut = new FileOutputStream("/home/lvuser/auto/Recording " +
Recorder.counter + ".JSON");//outputs recording and # to a json

                     fileout = new ObjectOutputStream(FileOut);//creates a serialized object output
using the file output

                     fileout.writeObject(r);//writes recorder object to file

                     fileout.close();//closes file

                     Recorder.writeToFile = false;//sets it to not write again

                           ++Recorder.counter;//increments # of recording

                        } catch (FileNotFoundException e1) {
```

```
                // TODO Auto-generated catch block
                e1.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

    }finally{//makes sure to close file if it fails to close on its own
                if(FileOut !=null){
                    try {
                        FileOut.close();
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }if(fileout !=null){
                    try {
                        fileout.close();
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }

}


public void setCounter(){//outputs counter so we never accidently overwrite data
```

```
                    FileOutputStream counterOut;//initializes file output

                    try {//in try-case because files are sometimes weird

                              outputCounter = Recorder.counter;//creates a new object to write using data
from Recorder.counter

                              counterOut = new
FileOutputStream("/home/lvuser/auto/Counter.JSON");//outputs to a file called Counter.JSON

                ObjectOutputStream counterFile = new ObjectOutputStream(counterOut);//outputs the
serialized object to the file

                counterFile.writeObject(outputCounter);//writes the object to file

                counterFile.close();//closes streams

                counterOut.close();

                    } catch (FileNotFoundException e) {


                    }catch(IOException i){}


          }


}
```