Formal Review Document

Date conducted: 2022-05-16

Reviewers and votes:
David Carpenter (Approved | Conditional approval | Rejected)
Joshua Hwang (Approved | Conditional approval | Rejected)
Saoirse Mooney (Approved | Conditional approval | Rejected)
Joseph Mooney (Approved | Conditional approval | Rejected)

Producer:
Joshua Meharg

Design Artifact Under Inspection:
`VehicleDisplay.java`

David Carpenter's Review:
ALSET Teammember

Preparation Effort: 2 hours
Assessment Effort: 3 hours
Rework Effort: 4 hours
Work Product Size: ~700 lines of code
Error(minor) = 23
Error(major) = 0
Error(total) = 23
Error density = 0.0328 average errors per line

The main issue with this class is its size - such a large class will be difficult to modify later due to its low readability. A good size to aim for is a class is between 50 and 200 lines. Instead, I think that this class should be broken up into more modular classes with their grid bag constraints settings baked in and any other more flexible settings passed as parameters at instantiation time. For example,

```
79 JButton cruise = new JButton("Cruise Control");
80 //c.fill = GridBagConstraints.HORIZONTAL;
81 c.fill = GridBagConstraints.NONE;
82 c.insets = new Insets(0, 0, 0, 15);
83 c.ipady = 0;
84 c.gridx = 0;
85 c.gridy = 2;
86 add(cruise, c);
```

could be moved to a class called `AlsetButton.java` where the parameters to the constructor are the text to be displayed on the button and the `GridBagConstraints c` to edit. All of the edits from lines 81 to 86 would then be taken care of by the `AlsetButton` button class and would not be visible to programmers working on the vehicle display class. Since these details would not be visible, cognitive load of the programmer would be reduced, making the class easier to modify. As it stands, the code is currently WET (Write Everything Twice), so encapsulating elements into their own classes would help DRY (Don't Repeat Yourself) it out.

Nitpicks:

- Remove comments on lines 32, 34-36, 49, 71, 80, 92, 237, 247, 259, 268, 277, 299, 308,

324, 412, 421, 482, 491, 559,

- Shorten lines 314, 362 by moving the string literal onto its own line
- Move all string literals to string constants file
- Name all integer literals with constant variables (remove all “magic numbers”)
- Use lambda expressions rather than anonymous inner classes for all `ActionListener` interfaces

Joshua Hwang's Review:
ALSET Teammember

Preparation Effort: 2 hours
Assessment Effort: 4 hours
Rework Effort: 2 hours
Work Product Size: ~700 lines of code
Error(minor) = 11
Error(major) = 0
Error(total) = 11
Error density = 0.0158 average errors per line

The effort required to correct a minor error (immediately after the review) was found to require 0.1 person-hours. The effort required for a major requirement was found to be 0.5 person-hour. Examining the review data collected, you find that the minor errors occur about 5 times more frequently than major errors. Therefore, you can estimate that the average effort to find and correct a requirements error during review is about 0.5 person-hours.

Joseph Mooney's Review:
Software Engineer at General Dynamics

Preparation Effort: 1 hours
Assessment Effort: 2 hours
Rework Effort: 2 hours
Work Product Size: ~700 lines of code
Work Product Size (WPS) = 2 hours
Error(minor) = 15
Error(major) = 0
Error(total) = 15
Error density = 0.021 average errors per line

•The effort required to correct a minor model error (immediately after the review) was found to require 2 person-hours.
• Examining the review data collected, you find that minor errors occur about 100 times more frequently than major errors.
• Therefore, you can estimate that the average effort to find and correct a requirements error during review is about 2 Person-hours.

The only issue is the readability and redundancy of the code throughout, specifically from lines 287 onwards. One actionPerformed(ActionEvent e) function could handle each action listener with the that ActionEvent parameter selecting which function should be called corresponding with the correct input.

Saoirse Mooney's Review:
ALSET Teammember

Preparation Effort: 1.5 hours
Assessment Effort: 1 hours
Rework Effort: 3 hours
Work Product Size: ~700 lines of code
Minor Errors: 5 (variable names, test case compatibility)
Major Errors: 2 (missing functions)
Work Product Size (WPS) = 2 hours
Error(minor) = 11
Error(major) = 2
Error(total) = 13
Error density = 0.019 average errors per line

The code was very well commented, which made review a straightforward task. Given my knowledge of how the car should be coded, I could better understand the functions and I saw some missing material. To prepare, I went over the functional architecture to ensure everything was in place and the test cases to see how compatible the code was with them. The rework took a bit of time due to my unfamiliarity with Java, but I was able to notice some errors due to my preparation. Overall, good organization and structure that closely followed the dev.md

CODE EXAMPLE: ~700 lines of code

```
package com.alset.htl;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;

import javax.swing.*;
import javax.swing.border.EtchedBorder;

import java.util.Random;
import java.awt.*;
import javax.swing.event.*;

public class VehicleDisplay extends JFrame {
    private boolean start = false;
    private boolean ccontrol = false;
    private boolean turnL = false;
    private boolean turnR = false;
    private int amount_dirs = 0;
    private boolean headlight = false;
    private String destination;
    private boolean smartweath = false;
    //class constructor
    public VehicleDisplay(){



        setBackground(Color.WHITE);
        setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();
        setTitle("ALSET PROTOTYPE V.1");
        //setExtendedState(JFrame.MAXIMIZED_BOTH);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        //setResizable(false);
        // setVisible(true);
        // c.fill = GridBagConstraints.HORIZONTAL;


        //menu bar
        JMenuBar mb = new JMenuBar();
        JMenu m1 = new JMenu("View");
        mb.add(m1);
        JMenuItem m1a = new JMenuItem("User");
        JMenuItem m1b = new JMenuItem("Technician");
        m1.add(m1a);
        m1.add(m1b);

        c.fill = GridBagConstraints.NONE;
        //c.weightx = 0.5;
        c.anchor = GridBagConstraints.FIRST_LINE_START;
        c.insets = new Insets(0, 0, 10, 0);
        add(mb, c);

        //makes technician login open when pressing technician menu option

        m1b.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
                LoginFrame frame = new LoginFrame();
            }
        });
```

```
67
68          //buttons
69
70          JButton start = new JButton("Button Start");
71          //c.fill = GridBagConstraints.HORIZONTAL;
72          c.fill = GridBagConstraints.NONE;
73          c.insets = new Insets(0, 0, 0, 20);
74          c.ipady = 0;
75          c.gridx = 0;
76          c.gridy = 1;
77          add(start, c);
78
79          JButton cruise = new JButton("Cruise Control");
80          //c.fill = GridBagConstraints.HORIZONTAL;
81          c.fill = GridBagConstraints.NONE;
82          c.insets = new Insets(0, 0, 0, 15);
83          c.ipady = 0;
84          c.gridx = 0;
85          c.gridy = 2;
86          add(cruise, c);
87
88
89          //Cruise Control input box
90          JTextField in = new JTextField(10);
91          c.fill = GridBagConstraints.VERTICAL;
92          //c.fill = GridBagConstraints.HORIZONTAL;
93          c.insets = new Insets(-10, 0, 15, 15);
94          c.ipady = 0;
95          c.gridx = 0;
96          c.gridy = 3;
97          add(in, c);
98
99          JButton curr = new JButton("Speed");
```

```
100         curr.setPreferredSize(new Dimension(100,40));
101         c.fill = GridBagConstraints.NONE;
102         c.insets = new Insets(0, 0, 0, 0);
103         curr.setBorder(BorderFactory.createRaisedBevelBorder());
104         c.ipady = 0;
105         c.gridx = 1;
106         c.gridy = 1;
107         add(curr, c);
108
109         JButton brake = new JButton("Brake");
110         brake.setPreferredSize(new Dimension(100, 40));
111         c.fill = GridBagConstraints.NONE;
112         c.insets = new Insets(0, 0, 10, 0);
113         brake.setBorder(BorderFactory.createRaisedBevelBorder());
114         c.ipady = 0;
115         c.gridx = 1;
116         c.gridy = 2;
117         add(brake, c);
118
119
120         JButton speed = new JButton("SET SPEED");
121         brake.setPreferredSize(new Dimension(100, 40));
122         c.insets = new Insets(0, 0, 0, 0);
123         c.fill = GridBagConstraints.NONE;
124         speed.setBorder(BorderFactory.createRaisedBevelBorder());
125         c.ipady = 0;
126         c.gridx = 2;
127         c.gridy = 0;
128         speed.setBackground(Color.black);
129         speed.setForeground(Color.white);
130         add(speed, c);
131
132         JSlider gears = new JSlider(0, 150);
133         c.insets = new Insets(0, 0, 20, 10);
```

```
134         gears.setPaintTrack(true);
135         gears.setPaintLabels(true);
136         gears.setOrientation(SwingConstants.HORIZONTAL);
137         gears.setMajorTickSpacing(50);
138         gears.setMinorTickSpacing(5);
139         gears.setBorder(BorderFactory.createLoweredBevelBorder());
140         c.gridx = 1;
141         c.gridy = 0;
142         add(gears, c);
143 
144 
145         //slider action
146         gears.addChangeListener(new javax.swing.event.ChangeListener() {
147             public void stateChanged(ChangeEvent ce){
148                 speed.setText("SET SPEED = " + gears.getValue());
149             }
150 
151         });
152 
153         speed.setText("SET SPEED = " + gears.getValue());
154 
155 
156 
157         JButton forward = new JButton("Forward");
158         brake.setPreferredSize(new Dimension(40, 40));
159         c.fill = GridBagConstraints.HORIZONTAL;
160         c.insets = new Insets(0, 0, 0, 20);
161         c.ipady = 0;
162         c.gridx = 3;
163         c.gridy = 1;
164         add(forward, c);
165 
166         JButton rev = new JButton("Reverse");
```

```
        brake.setPreferredSize(new Dimension(40, 40));
        c.fill = GridBagConstraints.HORIZONTAL;
        c.insets = new Insets(0, 0, 0, 20);
        c.ipady = 0;
        c.gridx = 4;
        c.gridy = 1;
        add(rev, c);

        JButton neutral = new JButton("Neutral");
        brake.setPreferredSize(new Dimension(100, 40));
        c.fill = GridBagConstraints.HORIZONTAL;
        c.insets = new Insets(0, 0, 0, 20);
        c.ipady = 0;
        c.gridx = 3;
        c.gridy = 2;
        add(neutral, c);


        JButton park = new JButton("Park");
        brake.setPreferredSize(new Dimension(100, 40));
        c.fill = GridBagConstraints.HORIZONTAL;
        c.insets = new Insets(0, 0, 0, 20);
        c.ipady = 0;
        c.gridx = 4;
        c.gridy = 2;
        add(park, c);

        JButton left = new JButton("Turn Left");
        left.setPreferredSize(new Dimension(100, 40));
        c.fill = GridBagConstraints.NONE;
        c.insets = new Insets(0, 0, 0, 30);
        left.setBorder(BorderFactory.createRaisedBevelBorder());
        c.ipady = 0;
```

```
200         c.gridx = 2;
201         c.gridy = 1;
202         add(left, c);
203 
204         JButton right = new JButton("Turn Right");
205         right.setPreferredSize(new Dimension(100, 40));
206         c.fill = GridBagConstraints.NONE;
207         c.insets = new Insets(0, 0, 0, 30);
208         right.setBorder(BorderFactory.createRaisedBevelBorder());
209         c.ipady = 0;
210         c.gridx = 2;
211         c.gridy = 2;
212         add(right, c);
213 
214 
215         //display
216         JLabel textlable = new JLabel("      DISPLAY      ");
217         c.fill = GridBagConstraints.HORIZONTAL;
218         c.anchor = GridBagConstraints.CENTER;
219         c.insets = new Insets(0, 0, 0, 0);
220         c.weightx = 0.0;
221         c.gridwidth = 5;
222         c.gridx = 0;
223         c.gridy = 4;
224         c.ipady = 0;
225         textlable.setBorder(BorderFactory.createEtchedBorder(EtchedBorder.RAISED));
226         add(textlable, c);
227 
228         JTextArea disp = new JTextArea("");
229         c.fill = GridBagConstraints.HORIZONTAL;
230         c.anchor = GridBagConstraints.CENTER;
231         c.insets = new Insets(0, 0, 0, 0);
232         c.weightx = 0.0;
233         c.gridwidth = 5;
```

```
234         c.gridx = 0;
235         c.gridy = 5;
236         c.ipady = 40;
237         //c.gridwidth = 3;
238         disp.setOpaque(true);
239         disp.setForeground(Color.green);
240         disp.setBackground(Color.black);
241         disp.setEditable(false);
242         add(disp, c);
243 
244 
245         //GPS
246         JButton gps = new JButton("Enter Destination");
247         //c.fill = GridBagConstraints.HORIZONTAL;
248         c.fill = GridBagConstraints.NONE;
249         c.insets = new Insets(-10, 0, 0, 0);
250         c.ipady = 0;
251         c.gridx = 5;
252         c.gridy = 1;
253         add(gps, c);
254 
255 
256         //GPS input box
257         JTextField in_gps = new JTextField(10);
258         c.fill = GridBagConstraints.NONE;
259         //c.fill = GridBagConstraints.HORIZONTAL;
260         c.insets = new Insets(-10, 0, 15, 0);
261         c.ipady = 0;
262         c.gridx = 5;
263         c.gridy = 2;
264         add(in_gps, c);
265 
266         JButton lights = new JButton("Head Lights: OFF");
```

```
267         c.fill = GridBagConstraints.NONE;
268         //c.fill = GridBagConstraints.HORIZONTAL;
269         c.insets = new Insets(0, 0, 0, 0);
270         c.ipady = 0;
271         c.gridx = 3;
272         c.gridy = 0;
273         add(lights, c);
274 
275         JButton weather = new JButton("Smart Weather: OFF");
276         c.fill = GridBagConstraints.NONE;
277         //c.fill = GridBagConstraints.HORIZONTAL;
278         c.insets = new Insets(0, 0, 0, 0);
279         c.ipady = 0;
280         c.gridx = 5;
281         c.gridy = 0;
282         add(weather, c);
283 
284 
285         //Button actions
286 
287         start.addActionListener(new ActionListener() {
288 
289             @Override
290             public void actionPerformed(ActionEvent e) {
291                 if (VehicleDisplay.this.start == false){
292 
293                     File file = new File("start.wav");
294                     String path = file.getAbsolutePath();
295                     //plays ignition sound
296                     try {
297                         SoundPlayer.play(path);
298                     } catch (IOException e1) {
299                         // TODO Auto-generated catch block
```

```
            Logger.inLog("Error: start failed", "SYS");
        }

        //thread sleep so the display starting up happens
        //after ignition sound is played
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

        Logger.inLog("Event: Vehicle started succesfully", "VCS");
        //starts up vehicle display
        disp.setText("=============================================================================================\n");
        disp.append("  Current Gear: " + SensorFusion.getGear() + "\n");

        if(SystemManagement.retrieveUpdate()){
            int n = JOptionPane.showConfirmDialog(VehicleDisplay.this, "Update avaliable, install now?", "System Management", JOptionPa
            System.out.println(n);
            if (n==0){
                try {
                    Thread.sleep(5000);
                } catch (InterruptedException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
                SystemManagement.getUpdate();
                JOptionPane.showMessageDialog(VehicleDisplay.this, "Updated Successfully to version " + SensorFusion.currVer);
            }
        }

        VehicleDisplay.this.start = true;
        return;
```

```
                }

                else if (VehicleDisplay.this.start == true && SensorFusion.getGear() == "Park"){
                    disp.setText("");
                    Logger.inLog("Event: Vehicle turned off succesfully", "VCS");
                    VehicleDisplay.this.start = false;
                    weather.setText("Smart Weather: OFF");
                    lights.setText("Head Lights: OFF");
                    ccontrol = false;
                    headlight = false;
                    return;
                }
            }
        });


        cruise.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                if (VehicleDisplay.this.start == true){
                if (!(in.getText().equals(""))){
                    if (VehicleControl.startCruiseControl(Integer.parseInt(in.getText()))){
                        if (smartweath == true){
                            disp.append("  Smart Weather deactivated\n");
                            weather.setText("Smart Weather: OFF");
                            smartweath = false;
                        }
                        disp.setText("=============================================================================================\n");
                        disp.append("  Cruise Control activated  \n");
                        disp.append("  Speed set to: " + in.getText() + "\n"  );
                        VehicleDisplay.this.ccontrol = true;
                        return;
```

```
                }
            }

        }
        }
    });

    left.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            if (VehicleDisplay.this.start == true && SensorFusion.getSpeed() > 0){
            if (VehicleControl.turnleft()){

                // if gps is activated
                if (amount_dirs > 1 && turnL == true){

                        amount_dirs--;
                        Random rand = new Random();
                        int num = rand.nextInt(2);
                        if (num == 0){
                            turnL = true;
                            disp.append("  GPS: turn left\n");
                        }
                        else{
                            turnL = false;
                            turnR = true;
                            disp.append("  GPS: turn right\n");
                        }

                }
                if (amount_dirs == 1 && turnL == true){
                    amount_dirs--;
```

```
400                     disp.append(   "  GPS: Arrived at destination '" + destination + "'\n");
401                     Logger.inLog("Event: Vehicle arrived at destination '" + destination + "'", "PLN");
402                 }
403             return;
404             }
405
406             File file = new File("sound98.wav");
407             String path = file.getAbsolutePath();
408             //plays alert sound
409             try {
410                 SoundPlayer.play(path);
411             } catch (IOException e1) {
412                 // TODO Auto-generated catch block
413                 Logger.inLog("Error: error with playing chime", "DSP");
414             }
415
416             //creates delay so sound plays before
417             //"can't turn" message is displayed
418             try {
419                 Thread.sleep(1000);
420             } catch (InterruptedException e1) {
421                 // TODO Auto-generated catch block
422                 e1.printStackTrace();
423             }
424
425             Random rand = new Random();
426             int num = rand.nextInt(2);
427             if (num == 0){
428                 disp.append("  Can't left right object too close!\n");
429             }
430             else {
431                 disp.append("  Warning, drifting out of lane!\n");
432             }
433             return;
```

```
            }
            else{
                Logger.inLog("Error: cannot turn left while not moving", "VCS");
            }

            }
        });

        right.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                if (VehicleDisplay.this.start == true && SensorFusion.getSpeed() > 0){
                if (VehicleControl.turnright()){

                    //if gps is activated
                    if (amount_dirs > 1 && turnR == true){

                        amount_dirs--;
                        Random rand = new Random();
                        int num = rand.nextInt(2);
                        if (num == 0){
                            turnR = false;
                            turnL = true;
                            disp.append("  GPS: turn left\n");
                        }
                        else{
                            turnR = true;
                            disp.append("  GPS: turn right\n");
                        }

                    }
                    if (amount_dirs == 1 && turnR == true){
```

```
467                     amount_dirs--;
468                     disp.append(  "  GPS: Arrived at destination '" + destination + "'\n");
469                     Logger.inLog("Event: Vehicle arrived at destination '" + destination + "'", "PLN");
470                 }
471 
472                 return;
473                 }
474 
475                 File file = new File("sound98.wav");
476                 String path = file.getAbsolutePath();
477 
478                 //plays alert sound
479                 try {
480                     SoundPlayer.play(path);
481                 } catch (IOException e1) {
482                     // TODO Auto-generated catch block
483                     Logger.inLog("Error: error with playing chime", "DSP");
484                 }
485 
486                 //creates delay so sound plays before
487                 //"can't turn" message is displayed
488                 try {
489                     Thread.sleep(1000);
490                 } catch (InterruptedException e1) {
491                     // TODO Auto-generated catch block
492                     e1.printStackTrace();
493                 }
494 
495                 Random rand = new Random();
496                 int num = rand.nextInt(2);
497                 if (num == 0){
498                     disp.append("  Can't turn right object too close!\n");
499                 }
```

```
                else {
                    disp.append("  Warning, drifting out of lane!\n");
                }
                return;
            }
            Logger.inLog("Error: cannot turn right while not moving", "VCS");

            }
        });

        brake.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                //include logic if curise control is on
                if (VehicleDisplay.this.start == true){
                    if (ccontrol){
                        VehicleControl.stopCruiseControl();
                        disp.append("  CruiseControl deactivated\n");
                        ccontrol = false;
                    }
                    VehicleControl.brake();
                }
            }
        });

        curr.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                //include logic if curise control is on
                if (VehicleDisplay.this.start == true){
                disp.append("  Current Speed: " + SensorFusion.getSpeed() + "\n");
                }
```

```
534             }
535         });
536 
537         speed.addActionListener(new ActionListener() {
538 
539             @Override
540             public void actionPerformed(ActionEvent e) {
541                 //include logic if curise control is on
542                 if (VehicleDisplay.this.start == true){
543                     if (VehicleDisplay.this.ccontrol == false){
544                         if (VehicleDisplay.this.smartweath == true){
545                             weather.setText("Smart Weather: OFF");
546                             smartweath = false;
547                         }
548                         Random rand = new Random();
549                         int num = rand.nextInt(101);
550                         if (num > 80){
551                         disp.append("  Warning, moving outside of lane!\n");
552 
553                         File file = new File("sound98.wav");
554                         String path = file.getAbsolutePath();
555                         //plays alert sound
556                         try {
557                             SoundPlayer.play(path);
558                         } catch (IOException e1) {
559                             // TODO Auto-generated catch block
560                             Logger.inLog("Error: error with playing chime", "DSP");
561                         }
562 
563                         }
564 
565 
566                             System.out.println(smartweath);
```

```
                        VehicleControl.setSpeed(gears.getValue());
                        return;
                    }
                    else if (VehicleDisplay.this.ccontrol == true){
                        VehicleControl.stopCruiseControl();
                        disp.append("  CruiseControl deactivated\n");
                        VehicleControl.setSpeed(gears.getValue());
                        VehicleDisplay.this.ccontrol = false;
                        return;
                    }


                }
            }
        });

        forward.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                //include logic if curise control is on
                if (VehicleDisplay.this.start == true){
                VehicleControl.setGear("Forward");
                disp.append("  Current Gear: " + SensorFusion.getGear() + "\n");
                }
            }
        });

        rev.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                //include logic if curise control is on
                if (VehicleDisplay.this.start == true){
```

```
601                 VehicleControl.setGear("Reverse");
602                 disp.append("  Current Gear: " + SensorFusion.getGear() + "\n");
603                 }
604             }
605         });
606 
607         neutral.addActionListener(new ActionListener() {
608 
609             @Override
610             public void actionPerformed(ActionEvent e) {
611                 //include logic if curise control is on
612                 if (VehicleDisplay.this.start == true){
613                 VehicleControl.setGear("Neutral");
614                 disp.append("  Current Gear: " + SensorFusion.getGear() + "\n");
615                 }
616             }
617         });
618 
619         park.addActionListener(new ActionListener() {
620 
621             @Override
622             public void actionPerformed(ActionEvent e) {
623                 //include logic if curise control is on
624                 if (VehicleDisplay.this.start == true){
625                 if (SensorFusion.getSpeed() == 0){
626                 VehicleControl.setGear("Park");
627                 disp.append("  Current Gear: " + SensorFusion.getGear() + "\n");
628                 return;
629                 }
630                 disp.append("  Can't Park while moving!\n");
631                 Logger.inLog("Error: Cannot Current Gear to 'Park' when current speed is not 0", "VCS");
632              }
633             }
```

```
667             public void actionPerformed(ActionEvent e) {
668                 //include logic if curise control is on
669                 if (VehicleDisplay.this.start == true){
670                     if (!headlight){
671                         lights.setText("Head Lights: ON");
672 
673                         //if adaptiveLights() returns true its night and
674                         //sets lights to Night Mode
675                         //else its day time and sets lights to default
676                         if (VehicleControl.adaptiveLights()){
677                             disp.append("  Headlights activated to default brightness\n");
678                         }
679                         disp.append("  Headlights activated to Night brightness\n");
680                         headlight = true;
681                     }
682                     else{
683                         lights.setText("Head Lights: OFF");
684                         disp.append("  Headlights deactivated\n");
685                         headlight = false;
686                     }
687                 }
688             }
689         });
690 
691 
692         weather.addActionListener(new ActionListener() {
693 
694             @Override
695             public void actionPerformed(ActionEvent e) {
696                 if (VehicleDisplay.this.start == true){
697                     if (!smartweath){
698                         if (VehicleControl.smartWeather()){
699                             disp.append("  Rainy Weather detected\n");
```

```
                            weather.setText("Smart Weather: ON");
                            smartweath = true;
                            return;
                        }
                            disp.append("  Rainy Weather not detected\n");
                    }
                    else {
                        weather.setText("Smart Weather: OFF");
                        smartweath = false;
                    }
                }

            }

        });




        //options


        //displaying

        setSize(1000, 1000);
        setVisible(true);
    }




}

}
```