```java
package project3;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.GregorianCalendar;

/**************************************************************************
 *  This program allows users to purchase a truck and makes sure users
 *  enter valid entries.
 *
 * @author Justin Von Kulajta Winn and Nick Layman
 * @version 1.8
 *************************************************************************/

public class boughtOnDialogTruck extends JDialog implements ActionListener {

    /** This is the text field that represents the truck title */
    private JTextField txtName;

    /** This is the text field that represents the truck's purchase date */
    private JTextField txtDate;

    /** This is the text field that represents the truck's trim package */
    private JTextField txtTrimPackage;

    /** This is the boolean that determines if the truck is a four by four */
    private JTextField txtFourbyFour;

    /** This is the text field that represents the truck's cost */
    private JTextField txtCost;

    /** This is the button that represents the 'ok' button */
    private JButton okButton;

    /** This is the button that represents the 'cancel' button */
    private JButton cancelButton;

    /** This is the J combo box that represents the select menu for which vehicle */
    private JComboBox<String> combobox;

    /** This is the integer that represents if the window should close or not */
    private int closeStatus;

    /** This is the auto being built by the user */
    private Auto auto;

    /** This is the integer that represents the action of the OK being pressed */
    static final int OK = 0;

    /** This is the integer that represents the action of the CANCEL being pressed */
    static final int CANCEL = 1;
```

```
 57
 58      /** This is the array that holds the number of days in each month */
 59      private static final int[] DAYS_IN_MONTH = {0, 31, 28, 31, 30, 31, 30, 31,
 60              31, 30, 31, 30, 31};
 61
 62      /**************************************************************************
 63       * Instantiate a Custom Dialog as 'modal' and wait for the
 64       * user to provide data and click on a button.
 65       * @param parent reference to the JFrame application
 66       * @param auto an instantiated object to be filled with data
 67       *************************************************************************/
 68      public boughtOnDialogTruck(JFrame parent, Auto auto) {
 69          // call parent and create a 'modal' dialog
 70          super(parent, true);
 71
 72          this.auto = auto;
 73          setTitle("Buying A Truck");
 74          closeStatus = CANCEL;
 75          setSize(400,200);
 76
 77          // prevent user from closing window
 78          setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);
 79
 80          // instantiate and display two text fields
 81          txtName = new JTextField("F150",30);
 82          txtDate = new JTextField(15);
 83          txtFourbyFour = new JTextField("True",15);
 84          txtTrimPackage = new JTextField("LT",15);
 85          txtCost = new JTextField("10100.00", 15);
 86
 87          String[] autoStrings = { "Truck"};
 88
 89          combobox = new JComboBox<>(autoStrings);
 90          txtDate.setText("10/17/2018");
 91          JPanel textPanel = new JPanel();
 92          textPanel.setLayout(new GridLayout(7,2));
 93
 94          textPanel.add(new JLabel(""));
 95          textPanel.add(combobox);
 96          textPanel.add(new JLabel(""));
 97          textPanel.add(new JLabel(""));
 98
 99          textPanel.add(new JLabel("Name of Car: "));
100          textPanel.add(txtName);
101          textPanel.add(new JLabel("bought on Date: "));
102          textPanel.add(txtDate);
103          textPanel.add(new JLabel("Trim Package"));
104          textPanel.add(txtTrimPackage);
105          textPanel.add(new JLabel("Four by Four"));
106          textPanel.add(txtFourbyFour);
107          textPanel.add(new JLabel("Amount Paid for"));
108          textPanel.add(txtCost);
109
110          getContentPane().add(textPanel, BorderLayout.CENTER);
111
112          // Instantiate and display two buttons
```

```
113         okButton = new JButton("OK");
114         cancelButton = new JButton("Cancel");
115         JPanel buttonPanel = new JPanel();
116         buttonPanel.add(okButton);
117         buttonPanel.add(cancelButton);
118         getContentPane().add(buttonPanel, BorderLayout.SOUTH);
119         okButton.addActionListener(this);
120         cancelButton.addActionListener(this);
121 
122         setVisible (true);
123     }
124 
125     /**************************************************************************
126      * This function activates when a button has been pressed on the dialog
127      * box
128      * @param e is used to check what button has been pressed
129      *************************************************************************/
130     public void actionPerformed(ActionEvent e) {
131 
132         JButton button = (JButton) e.getSource();
133 
134         // if OK clicked the fill the object
135         if (button == okButton) {
136             // save the information in the object
137             closeStatus = OK;
138             SimpleDateFormat df = new SimpleDateFormat("MM/dd/yyyy");
139             GregorianCalendar temp = new GregorianCalendar();
140             String tempName;
141             double tempCost;
142             String tempDate;
143             int day;
144             int month;
145             int year;
146 
147             try {
148                 tempName = txtName.getText();
149                 tempCost = Double.parseDouble(txtCost.getText());
150                 String[] dates = txtDate.getText().split("/");
151                 String months;
152                 String days;
153                 String years;
154                 if (dates.length == 3){
155                     months = dates[0];
156                     days = dates[1];
157                     years = dates[2];
158                 }
159                 else
160                     throw new IllegalArgumentException();
161 
162                 month = Integer.parseInt(months);//Converts the Strings into integers
163                 day = Integer.parseInt(days);
164                 year = Integer.parseInt(years);
165                 if (month < 1 || day < 1 || year < 1950 || month > 12)
166                     throw new IllegalArgumentException();
167 
168                 if (!isLeapYear(year)) {
```

```
169                     if (day > DAYS_IN_MONTH[month])
170                         throw new IllegalArgumentException();
171                 }
172                 else if (month == 2 && day > 29) {
173                     throw new IllegalArgumentException();
174                 }
175 
176                 if (tempName.equals(""))
177                     throw new Exception();
178                 if (tempCost <= 0)
179                     throw new NumberFormatException();
180             }
181             catch (NumberFormatException e2){
182                 JOptionPane.showMessageDialog(null, "Enter a cost above 0");
183                 return;
184             }
185             catch (IllegalArgumentException e5){
186                 JOptionPane.showMessageDialog(null,
187                         "Enter a correct date with the format month/day/year");
188                 return;
189             }
190             catch (Exception e3){
191                 JOptionPane.showMessageDialog(null, "Enter the name of the Truck");
192                 return;
193             }
194 
195             if (combobox.getSelectedIndex() == 1) {
196                 Date d = null;
197                 try {
198                     d = df.parse(txtDate.getText());
199                     temp.setTime(d);
200 
201                 } catch (ParseException e1) {
202                     JOptionPane.showMessageDialog(null, "Invalid Date");
203                     //unreachable because the date is checked before this
204                 }
205                 auto.setBoughtOn(temp);
206                 auto.setAutoName(txtName.getText());
207                 ((Truck) auto).setTrim(txtTrimPackage.getText());
208                 auto.setBoughtCost(Double.parseDouble(txtCost.getText()));
209 
210             }
211 
212             else {
213                 Date d = null;
214                 try {
215                     d = df.parse(txtDate.getText());
216                     temp.setTime(d);
217 
218                 } catch (ParseException e1) {
219                     JOptionPane.showMessageDialog(null, "Invalid Date");
220                     //unreachable because the date is checked before this
221                 }
222 
223                 auto.setBoughtOn(temp);
224                 auto.setAutoName(txtName.getText());
```

```
225                 auto.setBoughtCost(Double.parseDouble(txtCost.getText()));
226                 ((Truck) auto).setTrim(txtTrimPackage.getText());
227 
228                 if (txtFourbyFour.getText().equalsIgnoreCase("true"))
229                     ((Truck) auto).setFourByFour(true);
230                 else
231                     ((Truck) auto).setFourByFour(false);
232             }
233 
234         }
235 
236         if (button == cancelButton){
237             txtCost.setText("50000");
238         }
239 
240         // make the dialog disappear
241         dispose();
242     }
243 
244     /**************************************************************************
245      * This function returns the current close status
246      * @return the integer representing the current close status
247      *************************************************************************/
248     public int getCloseStatus(){
249         return closeStatus;
250     }
251 
252     /**************************************************************************
253      * This function determines if the year passed is leap year or not.
254      * @param year is the year that is checked if it is a leap year or not
255      * @return true if it is a leap year, false if it is not a leap year
256      *************************************************************************/
257     public static boolean isLeapYear(int year) {
258         return year % 4 == 0 && (year % 100 != 0 || year % 400 == 0);
259     }
260 }
```