```java
package view;

import java.awt.Color;
import java.awt.Component;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import java.util.InputMismatchException;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.LineBorder;

import model.Case;
import model.Client;
import model.Employee;
import use_case_controller.SearchCasesController;

/**
 * Class CommonElements to reduce repeating code in package view.
 * Responsible for constructing main navigation menu.
 * Contains other commonly used elements such as displaying an error message and styling a button.
 * @author Daria Vekic (Student ID: 586661)
 *
 */
public class CommonElements extends JFrame {

    private static final long serialVersionUID = 1L;
    private String error3 = "Please enter a valid case reference number.";

    //Controller required for use in Action Performed methods
    private SearchCasesController searchController = new SearchCasesController();

    // Instance fields for navigation menu
    private JPanel menuPanel;
    private JLabel menuLabel;
    private JButton addCaseBtn;
    private JButton viewAllCasesButton;
    private JButton viewCriminalBtn;
    private JButton viewImmBtn;
    private JButton viewPIBtn;
    private JButton ourStaffBtn;
    private JButton genReportBtn;
    private JTextField searchBarTxtField;
    private JButton searchButton;

    /**
```

```
55      * Method to construct the navigation menu.
56      * @param contentPane the JPanel to place menu on
57      * @param employees the List of Employee data required to add new Case
58      * @param clients the List of Client data required to add new Case and Client
59      * @param cases the List of Case data which new Case is added to
60      */
61     public void constructNavMenu(JPanel contentPane, ArrayList<Employee> employees,
62             ArrayList<Client> clients,
63             ArrayList<Case> cases) {
64         menuPanel = new JPanel();
65         menuPanel.setBackground(Color.LIGHT_GRAY);
66         menuPanel.setBounds(10, 11, 181, 646);
67         contentPane.add(menuPanel);
68         menuPanel.setLayout(null);
69
70         menuLabel = new JLabel("MENU");
71         menuLabel.setBounds(58, 11, 70, 23);
72         menuLabel.setFont(new Font("Verdana", Font.BOLD, 18));
73         menuPanel.add(menuLabel);
74
75         addCaseBtn = new JButton("Add New Case");
76         addCaseBtn.setBounds(11, 44, 160, 75);
77         styleButton(addCaseBtn);
78         addCaseBtn.addActionListener(new ActionListener()
79         {
80             public void actionPerformed(ActionEvent event) {
81                 addCaseBtnActionPerformed(event, employees, clients, cases);
82             } //end method actionPerformed
83         } //end method ActionListener
84         ); //end call to addCaseBtnActionPerformed ActionListener
85         menuPanel.add(addCaseBtn);
86
87         viewAllCasesButton = new JButton("View All Cases");
88         viewAllCasesButton.setBounds(11, 130, 160, 75);
89         styleButton(viewAllCasesButton);
90         viewAllCasesButton.addActionListener(new ActionListener()
91         {
92             public void actionPerformed(ActionEvent event) {
93                 viewAllCasesButtonActionPerformed(event, employees, clients, cases);
94             } //end method actionPerformed
95         } //end method ActionListener
96         ); //end call to viewAllCasesButtonActionPerformed ActionListener
97         menuPanel.add(viewAllCasesButton);
98
99         viewCriminalBtn = new JButton("View Criminal Cases");
100        viewCriminalBtn.setBounds(11, 216, 160, 75);
101        styleButton(viewCriminalBtn);
102        menuPanel.add(viewCriminalBtn);
103
104        viewImmBtn = new JButton("View Imm Cases");
105        viewImmBtn.setBounds(11, 302, 160, 75);
106        styleButton(viewImmBtn);
107        menuPanel.add(viewImmBtn);
108
```

```
109         viewPIBtn = new JButton("View PI Cases");
110         viewPIBtn.setBounds(11, 388, 160, 75);
111         styleButton(viewPIBtn);
112         menuPanel.add(viewPIBtn);
113 
114         ourStaffBtn = new JButton("Our Staff");
115         ourStaffBtn.setBounds(11, 474, 160, 75);
116         styleButton(ourStaffBtn);
117         menuPanel.add(ourStaffBtn);
118 
119         genReportBtn = new JButton("Generate Report");
120         genReportBtn.setBounds(11, 560, 160, 75);
121         styleButton(genReportBtn);
122         menuPanel.add(genReportBtn);
123 
124         searchBarTxtField = new JTextField();
125         searchBarTxtField.addMouseListener(new MouseAdapter() {
126             @Override
127             public void mouseClicked(MouseEvent e) {
128                 searchBarTxtField.setText("");
129             }
130         });
131         searchBarTxtField.setFont(new Font("Verdana", Font.PLAIN, 14));
132         searchBarTxtField.setText("Enter a Case Reference Number");
133         searchBarTxtField.setBounds(238, 40, 443, 56);
134         contentPane.add(searchBarTxtField);
135         searchBarTxtField.setColumns(10);
136 
137         searchButton = new JButton("Search");
138         searchButton.setBounds(723, 49, 116, 40);
139         styleButton(searchButton);
140         searchButton.addActionListener(new ActionListener()
141         {
142             public void actionPerformed(ActionEvent event) {
143                 searchButtonActionPerformed(event, cases, employees, clients);
144             } //end method actionPerformed
145         } //end method ActionListener
146         ); //end call to addCaseBtnActionPerformed ActionListener
147         contentPane.add(searchButton);
148     } //end method constructNavMenu
149 
150     /**
151      * Method to outline a JTextField red.
152      * Used when invalid data is entered.
153      * @param textField the JTextField to be outlined.
154      */
155     public void outlineRed(JTextField textField) {
156         textField.setBorder(new LineBorder(Color.RED, 2));
157     } //end method outline Red
158 
159     /**
160      * Method to display error message to screen.
161      * @param errorMessage the error message to be displayed.
162      * @param title the title of the JOptionPane.
```

```
163      */
164     public void showError(String errorMessage, String title) {
165         JOptionPane.showMessageDialog(null, errorMessage, title, JOptionPane.ERROR_MESSAGE);
166     } //end method showError
167 
168     /**
169      * Method to style a button.
170      * @param btn the JButton to be styled.
171      */
172     public void styleButton(JButton btn) {
173         btn.setBorder(null);
174         btn.setForeground(Color.WHITE);
175         btn.setBackground(Color.BLACK);
176         btn.setFont(new Font("Verdana", Font.BOLD, 12));
177         btn.setAlignmentX(Component.CENTER_ALIGNMENT);
178     } //end method styleButton
179 
180     /*------------------------ACTION LISTENERS------------------------*/
181 
182     /**
183      * The routine to follow if "Add New Case" button is pressed.
184      * Constructs AddCaseInterface object to display new window to screen.
185      * @param event the "Add New Case" button is pressed.
186      * @param employees the List of Employee data required to be searched.
187      * @param clients the List of Client data to be added to.
188      * @param cases the List of Case data to be added to.
189      */
190     private void addCaseBtnActionPerformed(ActionEvent event, ArrayList<Employee> employees,
191             ArrayList<Client> clients, ArrayList<Case> cases) {
192         this.setVisible(false);
193         AddCaseInterface newCaseForm = new AddCaseInterface(employees, clients, cases);
194         newCaseForm.setVisible(true);
195     } //end method addCaseBtnActionPerformed
196 
197     /**
198      * The routine to follow if the Search Button has been pressed.
199      * @param event the Search Button has been pressed.
200      * @param cases the List of Cases
201      * @param employees the List of Employees
202      * @param clients the List of Clients
203      */
204     private void searchButtonActionPerformed(ActionEvent event, ArrayList<Case> cases, ArrayList<Employee> employees, ArrayList<Client> clients) {
205         String caseNum = searchBarTxtField.getText();
206         try {
207             Client client = searchController.findCase(clients, caseNum);
208             if(client == null) {
209                 showError(error3, "Case Not Found");
210             } else {
211                 this.dispose();
212                 SearchResultInterface searchResults = new SearchResultInterface(employees, clients, cases, caseNum);
213                 searchResults.setVisible(true);
214             } //end if else
215         } //end try
216         catch(InputMismatchException e) {
```

```
217             System.out.println("Case does not exist");
218         } //end catch
219     } //end method searchButtonActionPerformed
220 
221     /**
222      * The routine to follow if "View Caseload" Button is pressed.
223      * Outputs data to console for testing purposes.
224      * @param event the "View Caseload" Button in navigation panel is pressed.
225      * @param clients the List from which data is retrieved.
226      */
227     private void viewAllCasesButtonActionPerformed(ActionEvent event, ArrayList<Employee> employees, ArrayList<Client> clients, ArrayList<Case>
    cases) {
228         //searchController.viewAllCases(tArea, clients);
229         this.dispose();
230         SearchResultInterface searchResults = new SearchResultInterface(employees, clients, cases);
231         searchResults.setVisible(true);
232     } //end method viewAllCasesButtonActionPerformed
233 } //end class CommonElements
```