```
 1 import javax.swing.*;
 2 import java.awt.*;
 3 import java.awt.event.ActionEvent;
 4 import java.awt.event.ActionListener;
 5 import java.util.ArrayList;
 6
 7 /*************************************************************
 8  * GUI for a Baby Name Database
 9  *
10  * @author Scott Grissom, Silas Agnew
11  * @version November 9, 2017
12  ************************************************************/
13 public class BabyNameGUI extends JFrame implements ActionListener{
14
15     /** Database */
16     BabyNamesDatabase namesDB;
17
18     /** Buttons */
19     JButton yearBtn;
20     JButton popularBtn;
21     JButton topTenBtn;
22     JButton nameBtn;
23
24     /** Text Fields */
25     JTextField yearField;
26     JTextField nameField;
27
28     /** Results text area */
29     JTextArea resultsArea;
30
31     /** menu items */
32     JMenuBar menus;
33     JMenu fileMenu;
34     JMenuItem quitItem;
35     JMenuItem openItem;
36     JMenuItem countItem;
37
38     /*****************************************************************
39      * Main Method
40      *****************************************************************/
41     public static void main(String args[]){
42         BabyNameGUI gui = new BabyNameGUI();
43         gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
44         gui.setTitle("Baby Names");
45         gui.pack();
46         gui.setVisible(true);
47     }
48
49     /*****************************************************************
50      * constructor installs all of the GUI components
51      *****************************************************************/
52     public BabyNameGUI(){
53         // Database
54         namesDB = new BabyNamesDatabase();
55
56         // set the layout to GridBag
```

```
 57         setLayout(new GridBagLayout());
 58         GridBagConstraints loc = new GridBagConstraints();
 59
 60         // create results area to span one column and 10 rows
 61         resultsArea = new JTextArea(20,20);
 62         JScrollPane scrollPane = new JScrollPane(resultsArea);
 63         loc.gridx = 0;
 64         loc.gridy = 1;
 65         loc.gridheight = 10;
 66         loc.insets.left = 20;
 67         loc.insets.right = 20;
 68         loc.insets.bottom = 20;
 69         add(scrollPane, loc);
 70
 71         // create Results label
 72         loc = new GridBagConstraints();
 73         loc.gridx = 0;
 74         loc.gridy = 0;
 75         loc.insets.bottom = 20;
 76         loc.insets.top = 20;
 77         add(new JLabel("Results"), loc);
 78
 79         // create Searches label
 80         loc = new GridBagConstraints();
 81         loc.gridx = 1;
 82         loc.gridy = 0;
 83         loc.gridwidth = 2;
 84         add(new JLabel("Searches"), loc);
 85
 86         // create Year label
 87         loc = new GridBagConstraints();
 88         loc.gridx = 1;
 89         loc.gridy = 1;
 90         add(new JLabel("Year "), loc);
 91
 92         // create Name label
 93         loc = new GridBagConstraints();
 94         loc.gridx = 1;
 95         loc.gridy = 7;
 96         loc.insets.top = 5;
 97         add(new JLabel("Name "), loc);
 98
 99         // create Year button
100         yearBtn = new JButton("By Year");
101         loc = new GridBagConstraints();
102         loc.gridx = 2;
103         loc.gridy = 2;
104         loc.anchor = GridBagConstraints.WEST;
105         add(yearBtn, loc);
106         yearBtn.setEnabled(false);
107
108         // create Popular sort button
109         popularBtn = new JButton("Most Popular");
110         loc = new GridBagConstraints();
111         loc.gridx = 2;
112         loc.gridy = 3;
```

```
113         loc.anchor = GridBagConstraints.WEST;
114         add(popularBtn, loc);
115         popularBtn.setEnabled(false);
116 
117         // create Top Ten button
118         topTenBtn = new JButton("Top Ten");
119         loc = new GridBagConstraints();
120         loc.gridx = 2;
121         loc.gridy = 4;
122         loc.anchor = GridBagConstraints.WEST;
123         add(topTenBtn, loc);
124         topTenBtn.setEnabled(false);
125 
126         // create Name sort button
127         nameBtn = new JButton("By Name");
128         loc = new GridBagConstraints();
129         loc.gridx = 2;
130         loc.gridy = 8;
131         loc.anchor = GridBagConstraints.WEST;
132         add(nameBtn, loc);
133         nameBtn.setEnabled(false);
134 
135         // create Year text field
136         yearField = new JTextField(5);
137         loc = new GridBagConstraints();
138         loc.gridx = 2;
139         loc.gridy = 1;
140         loc.anchor = GridBagConstraints.WEST;
141         add(yearField, loc);
142 
143         // create Name text field
144         nameField = new JTextField(10);
145         loc = new GridBagConstraints();
146         loc.gridx = 2;
147         loc.gridy = 7;
148         loc.anchor = GridBagConstraints.WEST;
149         loc.insets.top = 5;
150         add(nameField, loc);
151 
152         yearBtn.addActionListener(this);
153         popularBtn.addActionListener(this);
154         topTenBtn.addActionListener(this);
155         nameBtn.addActionListener(this);
156 
157         // Hint at the user to choose a file
158         resultsArea.setText("Choose a file under File>Open...");
159 
160         // hide details of creating menus
161         setupMenus();
162     }
163 
164     /**************************************************************
165      * This method is called when any button is clicked.  The proper
166      * internal method is called as needed.
167      *
168      * @param e the event that was fired
```

```
169      ************************************************************/
170     public void actionPerformed(ActionEvent e){
171
172         // extract the button that was clicked
173         JComponent buttonPressed = (JComponent) e.getSource();
174
175         // Allow user to load baby names from a file
176         if (buttonPressed == openItem){
177             openFile();
178             yearBtn.setEnabled(true);
179             popularBtn.setEnabled(true);
180             topTenBtn.setEnabled(true);
181             nameBtn.setEnabled(true);
182             countItem.setEnabled(true);
183         } else if (buttonPressed == countItem) {
184             displayCounts();
185         } else if (buttonPressed == quitItem) {
186             System.exit(0);
187         } else if (buttonPressed == yearBtn) {
188             displayByYear();
189         } else if (buttonPressed == popularBtn) {
190             displayMostPopular();
191         } else if (buttonPressed == topTenBtn) {
192             displayTopTen();
193         } else if (buttonPressed == nameBtn) {
194             displayByName();
195         }
196     }
197
198     /*****************************************************************
199      * open a data file with the name selected by the user
200      *****************************************************************/
201     private void openFile(){
202
203         // create File Chooser so that it starts at the current directory
204         String userDir = System.getProperty("user.dir");
205         JFileChooser fc = new JFileChooser(userDir);
206
207         // show File Chooser and wait for user selection
208         int returnVal = fc.showOpenDialog(this);
209
210         // did the user select a file?
211         if (returnVal == JFileChooser.APPROVE_OPTION) {
212             String filename = fc.getSelectedFile().getName();
213             namesDB.readBabyNameData(filename);
214             resultsArea.setText("");
215         }
216     }
217
218     /*****************************************************
219     Creates the menu items
220      *****************************************************/
221     private void setupMenus(){
222         fileMenu = new JMenu("File");
223         quitItem = new JMenuItem("Quit");
224         countItem = new JMenuItem("Counts");
```

```
225         countItem.setEnabled(false);
226         openItem = new JMenuItem("Open...");
227         fileMenu.add(countItem);
228         fileMenu.add(openItem);
229         fileMenu.add(quitItem);
230         menus = new JMenuBar();
231         setJMenuBar(menus);
232         menus.add(fileMenu);
233 
234         openItem.addActionListener(this);
235         countItem.addActionListener(this);
236         quitItem.addActionListener(this);
237     }
238 
239     //-Helper Methods-----------------------------------------------//
240 
241     /**
242      * Displays to {@code resultArea} all names in {@code list} as
243      * well as a total of items.
244      * NOTE: Does not clear the text area
245      * @param list Content to display
246      */
247     private void displayNames(ArrayList<BabyName> list)
248     {
249        for (BabyName b : list)
250            resultsArea.append(b.toString() + "\n");
251        resultsArea.append("\nTotal: " + list.size());
252     }
253 
254     /**
255      * Gets the requested year from GUI and displays the most popular boy
256      * and girl name from that year.
257      * This will return prematurely if the year input is ill-formatted.
258      */
259     private void displayMostPopular()
260     {
261         // Get year
262         int year = -1;
263         try {
264             year = Integer.parseInt(yearField.getText());
265         }
266         catch (NumberFormatException ex)
267         {
268             JOptionPane.showMessageDialog(this, "Enter a valid year.");
269             return;
270         }
271 
272         // Get boy and girl
273         BabyName boy = namesDB.mostPopularBoy(year);
274         BabyName girl = namesDB.mostPopularGirl(year);
275 
276         // Display
277         resultsArea.setText("");
278         resultsArea.append("The most popular names in " + year + "\n\n");
279 
280         if (boy.getCount() > 0)
```

```
281             resultsArea.append(boy.toString());
282         else
283             resultsArea.append("No boy names in " + year);
284 
285         if (girl.getCount() > 0)
286             resultsArea.append("\n" + girl.toString());
287         else
288             resultsArea.append("\nNo girl names in " + year);
289     }
290 
291     /**
292      * Displays all names in a given year.
293      * This will return prematurely if the year input is ill-formatted.
294      */
295     private void displayByYear()
296     {
297         // Get year
298         int year = -1;
299         try {
300             year = Integer.parseInt(yearField.getText());
301         }
302         catch (NumberFormatException ex)
303         {
304             JOptionPane.showMessageDialog(this, "Enter a valid year.");
305             return;
306         }
307 
308         // Get and display
309         resultsArea.setText("");
310         ArrayList<BabyName> names = namesDB.searchForYear(year);
311         displayNames(names);
312         resultsArea.append("\nAll names in " + year);
313     }
314 
315     /**
316      * Displays the top ten baby names in a given year.
317      * This will return prematurely if the year input is ill-formatted
318      */
319     private void displayTopTen()
320     {
321         // Get year
322         int year = -1;
323         try {
324             year = Integer.parseInt(yearField.getText());
325         }
326         catch (NumberFormatException ex)
327         {
328             JOptionPane.showMessageDialog(this, "Enter a valid year.");
329             return;
330         }
331 
332         resultsArea.setText("");
333         resultsArea.append("Top Ten baby names in " + year + "\n\n");
334         ArrayList<BabyName> topTen = namesDB.topTenNames(year);
335         displayNames(topTen);
336     }
```

```
337
338     /**
339      * Displays all entries in the DB that match a given name.
340      */
341     private void displayByName()
342     {
343         String name = nameField.getText();
344         if (name.length() == 0)
345         {
346             JOptionPane.showMessageDialog(this, "Enter a valid name.");
347         }
348         ArrayList<BabyName> names = namesDB.searchForName(name);
349
350         resultsArea.setText("");
351         if (names.size() <= 0)
352         {
353             resultsArea.append("no " + name + " found");
354         }
355         else
356         {
357             displayNames(names);
358             resultsArea.append("\nAll years with " + name);
359         }
360     }
361
362     /**
363      * Displays total entries, boy, and girl counts for the database.
364      */
365     private void displayCounts()
366     {
367         resultsArea.setText("");
368         resultsArea.append("Total Counts\n\n");
369         resultsArea.append("Total Girls: " + namesDB.countAllGirls());
370         resultsArea.append("\nTotal Boys: " + namesDB.countAllBoys());
371         resultsArea.append("\nTotal Names: " + namesDB.countAllNames());
372
373     }
374 }
```