```
 1 /**
 2  * Class that manages a database of {@code BabyName}s.
 3  * Provides some sorting and querying functionality for the database.
 4  *
 5  * @author Silas Agnew
 6  * @version November 9, 2017
 7  */
 8
 9 import javax.swing.*;
10 import java.io.FileInputStream;
11 import java.io.FileNotFoundException;
12 import java.io.IOException;
13 import java.util.ArrayList;
14 import java.util.Scanner;
15
16 public class BabyNamesDatabase
17 {
18     ArrayList<BabyName> babyNames = null;
19
20     //-Constructor(s)-----------------------------------------------//
21
22     /**
23      * Constructs a empty database of baby names in memory.
24      * Call {@link BabyNamesDatabase#readBabyNameData(String)} to populate
25      * database.
26      */
27     public BabyNamesDatabase()
28     {
29         babyNames = new ArrayList<>();
30     }
31
32     //-Accessors----------------------------------------------------//
33
34     /**
35      * @return Total count of name entries in the database
36      */
37     public int countAllNames() { return babyNames.size(); }
38
39     /**
40      * @return Total count of boy babies in the database
41      */
42     public int countAllBoys()
43     {
44         int count = 0;
45         for (BabyName b : babyNames)
46             if (!b.isFemale()) count += b.getCount();
47         return count;
48     }
49
50     /**
51      * @return Total count of girl babies in the database
52      */
53     public int countAllGirls()
54     {
55         int count = 0;
56         for (BabyName b : babyNames)
```

```
 57             if (b.isFemale()) count += b.getCount();
 58         return count;
 59     }
 60 
 61     /**
 62      * @param year Year to search in
 63      * @return The most popular girl name of {@code year}
 64      */
 65     public BabyName mostPopularGirl(int year)
 66     {
 67         BabyName top = new BabyName("null", true, -1, year);
 68         for (BabyName b : babyNames)
 69         {
 70             if (b.isFemale() && b.getYear() == year && top.compareTo(b) > 0)
 71                 top = b;
 72         }
 73         return top;
 74     }
 75 
 76     /**
 77      * @param year Year to search in
 78      * @return The most popular boy name of {@code year}
 79      */
 80     public BabyName mostPopularBoy(int year)
 81     {
 82         BabyName top = new BabyName("null", true, 0, year);
 83         for (BabyName b : babyNames)
 84         {
 85             if (!b.isFemale() && b.getYear() == year && top.compareTo(b) > 0)
 86                 top = b;
 87         }
 88         return top;
 89     }
 90 
 91     /**
 92      * @param name Name to search for
 93      * @return A list of all entries of {@code name}
 94      */
 95     public ArrayList<BabyName> searchForName(String name)
 96     {
 97         ArrayList<BabyName> names = new ArrayList<>();
 98         for (BabyName b : babyNames)
 99             if (b.getName().equalsIgnoreCase(name))
100                 names.add(b);
101         return names;
102     }
103 
104     /**
105      * @param year The year to search for
106      * @return A list of all entries of {@code year}
107      */
108     public ArrayList<BabyName> searchForYear(int year)
109     {
110         ArrayList<BabyName> names = new ArrayList<>();
111         for (BabyName b : babyNames)
112             if (b.getYear() == year)
```

```
113                 names.add(b);
114         return names;
115     }
116 
117     /**
118      * Sorts the list of baby names in a year for the top ten
119      * @param year Year to search for
120      * @return Top ten popular names. Returns an empty array if no names
121      */
122     public ArrayList<BabyName> topTenNames(int year)
123     {
124         ArrayList<BabyName> topTen = searchForYear(year);
125         topTen.sort(null);
126         return new ArrayList<>(
127                 (topTen.size() > 10) ? topTen.subList(0, 10) : topTen);
128     }
129 
130     //-Mutator(s)-----------------------------------------------//
131 
132     /**
133      * Populates {@code babyNames} with CSV data from the file: {@code filename}
134      * This method will terminate if there isn't a file found with name:
135      *  {@code filename}
136      * @param  filename The name of the file to read data from
137      */
138     public void readBabyNameData(String filename)
139     {
140         FileInputStream file;
141         Scanner scnr;
142         int errorCount = 0;
143         int lineCount = 0;
144 
145         // Included in case specific line numbers were needed for debugging
146         ArrayList<Integer> errorLines = new ArrayList<>();
147 
148         // Open CSV file
149         try {
150             file = new FileInputStream(filename);
151         }
152         catch (FileNotFoundException ex)
153         {
154             JOptionPane.showMessageDialog(null, ex.getMessage());
155             return;
156         }
157         scnr = new Scanner(file);
158 
159         // Scan line per line
160         while (scnr.hasNextLine())
161         {
162             try {
163                 babyNames.add(BabyName.BabyNameBuilder(scnr.nextLine(), lineCount));
164             }
165             catch (IllegalArgumentException ex)
166             {
167                 errorCount++;
168                 errorLines.add(lineCount);
```

```
169             }
170             lineCount++;
171         }
172 
173         // Warn of errors (if any)
174         if (errorCount > 0)
175         {
176             JOptionPane.showMessageDialog(
177                     null, errorCount +
178                             " entries were not included due to formatting errors.");
179         }
180 
181         try
182         {
183             file.close();
184         }
185         catch (IOException ex)
186         {
187             JOptionPane.showMessageDialog(null,
188                     "If you're reading this, you probably deleted the file " +
189                             "before this program was finished with it. Congrats!");
190         }
191     }
192 
193     //-Test Methods----------------------------------------------//
194 
195     /**
196      * A test method that prints out general tests to the console.
197      */
198     public void printNames()
199     {
200         System.out.println("Total Counts\n");
201         System.out.println("Total Girls: " + countAllGirls());
202         System.out.println("Total Boys: " + countAllBoys());
203         System.out.println("Total Names: " + countAllNames());
204 
205         System.out.println("Most Pop. Girl: " + mostPopularGirl(1999));
206         System.out.println("Most Pop. Boy: " + mostPopularBoy(1999));
207         System.out.println();
208 
209         ArrayList<BabyName> l = searchForName("silas");
210         for (BabyName b : l)
211             System.out.println(b);
212         System.out.println();
213         l.clear();
214 
215         l = topTenNames(1999);
216         for (BabyName b : l)
217             System.out.println(b);
218     }
219 }
220 
```