```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

/*********************************************************
 * GUI for a Baby Name Database
 *
 * @author Scott Grissom, Silas Agnew
 * @version November 9, 2017
 *********************************************************/
public class BabyNameGUI extends JFrame implements ActionListener{

    /** Database */
    BabyNamesDatabase namesDB;

    /** Buttons */
    JButton yearBtn;
    JButton popularBtn;
    JButton topTenBtn;
    JButton nameBtn;

    /** Text Fields */
    JTextField yearField;
    JTextField nameField;

    /** Results text area */
    JTextArea resultsArea;

    /** menu items */
    JMenuBar menus;
    JMenu fileMenu;
    JMenuItem quitItem;
    JMenuItem openItem;
    JMenuItem countItem;

    /*****************************************************************
     * Main Method
     ****************************************************************/
    public static void main(String args[]){
        BabyNameGUI gui = new BabyNameGUI();
        gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        gui.setTitle("Baby Names");
        gui.pack();
        gui.setVisible(true);
    }

    /*****************************************************************
     * constructor installs all of the GUI components
     ****************************************************************/
    public BabyNameGUI(){
        // Database
        namesDB = new BabyNamesDatabase();

        // set the layout to GridBag
        setLayout(new GridBagLayout());
        GridBagConstraints loc = new GridBagConstraints();

        // create results area to span one column and 10 rows
        resultsArea = new JTextArea(20,20);
        JScrollPane scrollPane = new JScrollPane(resultsArea);
        loc.gridx = 0;
        loc.gridy = 1;
        loc.gridheight = 10;
        loc.insets.left = 20;
        loc.insets.right = 20;
        loc.insets.bottom = 20;
        add(scrollPane, loc);

        // create Results label
        loc = new GridBagConstraints();
        loc.gridx = 0;
        loc.gridy = 0;
        loc.insets.bottom = 20;
        loc.insets.top = 20;
        add(new JLabel("Results"), loc);

        // create Searches label
        loc = new GridBagConstraints();
        loc.gridx = 1;
        loc.gridy = 0;
        loc.gridwidth = 2;
        add(new JLabel("Searches"), loc);

        // create Year label
        loc = new GridBagConstraints();
        loc.gridx = 1;
        loc.gridy = 1;
        add(new JLabel("Year "), loc);

        // create Name label
        loc = new GridBagConstraints();
        loc.gridx = 1;
        loc.gridy = 7;
        loc.insets.top = 5;
        add(new JLabel("Name "), loc);

        // create Year button
        yearBtn = new JButton("By Year");
        loc = new GridBagConstraints();
        loc.gridx = 2;
        loc.gridy = 2;
        loc.anchor = GridBagConstraints.WEST;
        add(yearBtn, loc);
        yearBtn.setEnabled(false);

        // create Popular sort button
        popularBtn = new JButton("Most Popular");
        loc = new GridBagConstraints();
        loc.gridx = 2;
        loc.gridy = 3;
```

```java
        loc.anchor = GridBagConstraints.WEST;
        add(popularBtn, loc);
        popularBtn.setEnabled(false);

        // create Top Ten button
        topTenBtn = new JButton("Top Ten");
        loc = new GridBagConstraints();
        loc.gridx = 2;
        loc.gridy = 4;
        loc.anchor = GridBagConstraints.WEST;
        add(topTenBtn, loc);
        topTenBtn.setEnabled(false);

        // create Name sort button
        nameBtn = new JButton("By Name");
        loc = new GridBagConstraints();
        loc.gridx = 2;
        loc.gridy = 8;
        loc.anchor = GridBagConstraints.WEST;
        add(nameBtn, loc);
        nameBtn.setEnabled(false);

        // create Year text field
        yearField = new JTextField(5);
        loc = new GridBagConstraints();
        loc.gridx = 2;
        loc.gridy = 1;
        loc.anchor = GridBagConstraints.WEST;
        add(yearField, loc);

        // create Name text field
        nameField = new JTextField(10);
        loc = new GridBagConstraints();
        loc.gridx = 2;
        loc.gridy = 7;
        loc.anchor = GridBagConstraints.WEST;
        loc.insets.top = 5;
        add(nameField, loc);

        yearBtn.addActionListener(this);
        popularBtn.addActionListener(this);
        topTenBtn.addActionListener(this);
        nameBtn.addActionListener(this);

        // Hint at the user to choose a file
        resultsArea.setText("Choose a file under File>Open...");

        // hide details of creating menus
        setupMenus();
    }

    /*****************************************************************
     * This method is called when any button is clicked.  The proper
     * internal method is called as needed.
     *
     * @param e the event that was fired
     *************************************************************/
    public void actionPerformed(ActionEvent e){

        // extract the button that was clicked
        JComponent buttonPressed = (JComponent) e.getSource();

        // Allow user to load baby names from a file
        if (buttonPressed == openItem){
            openFile();
            yearBtn.setEnabled(true);
            popularBtn.setEnabled(true);
            topTenBtn.setEnabled(true);
            nameBtn.setEnabled(true);
            countItem.setEnabled(true);
        } else if (buttonPressed == countItem) {
            displayCounts();
        } else if (buttonPressed == quitItem) {
            System.exit(0);
        } else if (buttonPressed == yearBtn) {
            displayByYear();
        } else if (buttonPressed == popularBtn) {
            displayMostPopular();
        } else if (buttonPressed == topTenBtn) {
            displayTopTen();
        } else if (buttonPressed == nameBtn) {
            displayByName();
        }
    }

    /**************************************************************
     * open a data file with the name selected by the user
     *************************************************************/
    private void openFile(){

        // create File Chooser so that it starts at the current directory
        String userDir = System.getProperty("user.dir");
        JFileChooser fc = new JFileChooser(userDir);

        // show File Chooser and wait for user selection
        int returnVal = fc.showOpenDialog(this);

        // did the user select a file?
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            String filename = fc.getSelectedFile().getName();
            namesDB.readBabyNameData(filename);
            resultsArea.setText("");
        }
    }

    /*****************************************************
    Creates the menu items
     *****************************************************/
    private void setupMenus(){
        fileMenu = new JMenu("File");
        quitItem = new JMenuItem("Quit");
        countItem = new JMenuItem("Counts");
        countItem.setEnabled(false);
        openItem = new JMenuItem("Open...");
        fileMenu.add(countItem);
        fileMenu.add(openItem);
        fileMenu.add(quitItem);
        menus = new JMenuBar();
        setJMenuBar(menus);
        menus.add(fileMenu);

        openItem.addActionListener(this);
        countItem.addActionListener(this);
        quitItem.addActionListener(this);
    }

    //-Helper Methods---------------------------------------------//

    /**
     * Displays to {@code resultArea} all names in {@code list} as
     * well as a total of items.
     * NOTE: Does not clear the text area
     * @param list Content to display
     */
    private void displayNames(ArrayList<BabyName> list)
    {
       for (BabyName b : list)
           resultsArea.append(b.toString() + "\n");
       resultsArea.append("\nTotal: " + list.size());
    }

    /**
     * Gets the requested year from GUI and displays the most popular boy
     * and girl name from that year.
     * This will return prematurely if the year input is ill-formatted.
     */
    private void displayMostPopular()
    {
        // Get year
        int year = -1;
        try {
            year = Integer.parseInt(yearField.getText());
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(this, "Enter a valid year.");
            return;
        }

        // Get boy and girl
        BabyName boy = namesDB.mostPopularBoy(year);
        BabyName girl = namesDB.mostPopularGirl(year);

        // Display
        resultsArea.setText("");
        resultsArea.append("The most popular names in " + year + "\n\n");

        if (boy.getCount() > 0)
```

```java
            resultsArea.append(boy.toString());
        else
            resultsArea.append("No boy names in " + year);

        if (girl.getCount() > 0)
            resultsArea.append("\n" + girl.toString());
        else
            resultsArea.append("\nNo girl names in " + year);
    }

    /**
     * Displays all names in a given year.
     * This will return prematurely if the year input is ill-formatted.
     */
    private void displayByYear()
    {
        // Get year
        int year = -1;
        try {
            year = Integer.parseInt(yearField.getText());
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(this, "Enter a valid year.");
            return;
        }

        // Get and display
        resultsArea.setText("");
        ArrayList<BabyName> names = namesDB.searchForYear(year);
        displayNames(names);
        resultsArea.append("\nAll names in " + year);
    }

    /**
     * Displays the top ten baby names in a given year.
     * This will return prematurely if the year input is ill-formatted
     */
    private void displayTopTen()
    {
        // Get year
        int year = -1;
        try {
            year = Integer.parseInt(yearField.getText());
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(this, "Enter a valid year.");
            return;
        }

        resultsArea.setText("");
        resultsArea.append("Top Ten baby names in " + year + "\n\n");
        ArrayList<BabyName> topTen = namesDB.topTenNames(year);
        displayNames(topTen);
    }

    /**
     * Displays all entries in the DB that match a given name.
     */
    private void displayByName()
    {
        String name = nameField.getText();
        if (name.length() == 0)
        {
            JOptionPane.showMessageDialog(this, "Enter a valid name.");
        }
        ArrayList<BabyName> names = namesDB.searchForName(name);

        resultsArea.setText("");
        if (names.size() <= 0)
        {
            resultsArea.append("no " + name + " found");
        }
        else
        {
            displayNames(names);
            resultsArea.append("\nAll years with " + name);
        }
    }

    /**
     * Displays total entries, boy, and girl counts for the database.
     */
    private void displayCounts()
    {
        resultsArea.setText("");
        resultsArea.append("Total Counts\n\n");
        resultsArea.append("Total Girls: " + namesDB.countAllGirls());
        resultsArea.append("\nTotal Boys: " + namesDB.countAllBoys());
        resultsArea.append("\nTotal Names: " + namesDB.countAllNames());

    }
}
```