```
/*******************************************************************************************
 *  This class provides the interface to the user to analyze county, state and US population data.
 *
 *  The user is requested to choose a menu option, then enter the data required to fulfill the request.
 *
 *  The program checks for various errors:
 *  the file must exist and data must be stored properly
 *  the user must enter valid responses, or an error message is displayed and the question is reasked:
 *      The option chosen must be between 1 and 8
 *      The FIPS code must have 5 characters
 *      The state code must have 2 characters
 *      The user is not allowed to "x" the corner box to exit the program
 *      The year must be in range
 *      Improper input is detected
 *  The program will allow the user to enter 2 digits for the year
 *  The program will allow the user to use mixed case to search for strings
 *  The program will add " County" to search for a county if that string does not end the search string
 *      for counties
 *
 *  CST 183 Programming Assignment 7
 *  @author Michael Clinesmith
 *******************************************************************************************/
import javax.swing.JOptionPane;
import java.util.Scanner;
import java.io.*;
import java.util.StringTokenizer;

public class PopulationAnalysis
{
    // constants usable by entire class
    final static int NUM_OF_COUNTIES = 5000;
    final static int FIRST_YEAR = 2010, LAST_YEAR = 2017;
    final static int NUM_OF_YEARS = LAST_YEAR - FIRST_YEAR + 1;

    /**
     * This is the main driver method.  It calls methods to load data from a file into an array of
     * CountyData objects, then a method that processes the user requests.
     * @param args String array (not used)
     */
    public static void main(String[] args)
    {
        final String POPULATION_DATA = "countyPopData1017.txt";
        int arrayElements;

        CountyData[] countyData = new CountyData[NUM_OF_COUNTIES];

        displayIntroductionMessage(POPULATION_DATA);

        // open file and create data array
        arrayElements = createDataArray( POPULATION_DATA, countyData );

        // process user requests (main program)
        processUserRequests(countyData, arrayElements);

        displayClosingMessage();
    }

    /**
     * This method prints opening messages, including the filename with the population data.
     * @param filename String: name of the file with the population data
```

30/30 points for Program 7

Great solution. Searches produced accurate results. Additional searches were included that were not required which was cool

Great object oriented approach.

```
 */
public static void displayIntroductionMessage(String filename)
{
    String message;

    message = "Welcome to the Population Analysis Program!\n\n" +
            "This program was designed by Michael Clinesmith";

    JOptionPane.showMessageDialog(null, message);

    message = "This program processes population data located\n" +
            "in the file " + filename + " and gives the\n" +
            "user options to search and display populations\n" +
            "based on county, state or US.";

    JOptionPane.showMessageDialog(null, message);
}
/**
 * This method prints a closing message, thanking the user for using the program
 */
public static void displayClosingMessage()
{
    String message = "Thank you for using the Population Analysis Program!";

    JOptionPane.showMessageDialog(null, message);
}

 /**
 * This method loads the population data from a file into a CountyData object array.
 * The method handles file exception errors and well end the program if an error occurs.
 *
 * @param filename String: name of the file containing population data
 * @param countyArray CountyData array: used to store the population data
 * @return int: the number of objects (counties) stored in the array
 */
public static int createDataArray(String filename, CountyData[] countyArray)
{
    String message;
    File populationData;                // file that holds the population data
    Scanner inputFile;                  // used to get data from file
    int i = 0, j = 0, numElems = 0;
    String inputLine;                   // String used to get a line of file input

    /*code modified from MICountyList by Klinger */
    try
    {
        String fips, name, state;                     // Work variables
        int[] array = new int[NUM_OF_YEARS];

        StringTokenizer lineTokens;     //  used to get tokens from data input

        // Build list of county objects
        populationData = new File(filename);

        if(!populationData.exists())  // file not found
        {
            message = "The file " + filename + " does not exist for processing data.\n" +
                    "The program will now end.";

            JOptionPane.showMessageDialog(null, message);
```

```
            System.exit(0);
        }

        inputFile = new Scanner(populationData);

        // Read input file while more data exist
        // Read one line at a time (assuming each line contains one username)
        i = 0;          // used to work through array elements
        while (inputFile.hasNext())
        {
            inputLine = inputFile.nextLine();
            lineTokens = new StringTokenizer(inputLine,",");

            // Read all data on one line
            fips      = lineTokens.nextToken();
            name      = lineTokens.nextToken();
            state     = lineTokens.nextToken();

            // Read population data
            for (j=0; j<NUM_OF_YEARS; j++)
            {
                array[j] = Integer.parseInt((lineTokens.nextToken()));
            }

            countyArray[i] = new CountyData(fips,name,state, array);

            i++;
        }
        numElems = i;    // Capture number of elements

        inputFile.close();
    }
    catch (IOException e)  // if error loading data, give error message and end program
    {
        message = "There was an error processing the file " + filename + ".\n" +
                "The program will now end.";

        JOptionPane.showMessageDialog(null, message);
        System.exit(0);
    }

    message = "The data from the file " + filename +
            "\nis now uploaded into memory.";
    JOptionPane.showMessageDialog(null, message);

    return numElems;
}

/**
 * This method requests the user enter an option on the type of population data to view
 * then calls the appropriate method to handle the request
 *
 * @param countyData  CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 */
public static void processUserRequests(CountyData[] countyData, int numElems)
{
    String message;
    boolean repeatRequest = true;
    int choice;
```

```
        do      // give user options, and repeat if 8 not selected
        {
            message = "Choose from the following options:\n" +
                    "--------------------------------\n" +
                    "1: Find county FIPS code\n" +
                    "2: County population by year\n" +
                    "3: County population change\n" +
                    "4: State population by year\n" +
                    "5: State population change\n" +
                    "6: US population by year\n" +
                    "7: US population change\n" +
                    "8: Exit program";

            // uses the Dialogue class to handle error checking with the input dialogue boxes
            choice = Dialogue.inputInt(message);

            switch (choice)
            {
                case 1:
                    findFIPScode(countyData, numElems);
                    break;
                case 2:
                    findCountyPopulationByYear(countyData, numElems);
                    break;
                case 3:
                    findCountyPopulationChange(countyData, numElems);
                    break;
                case 4:
                    findStatePopulationByYear(countyData, numElems);
                    break;
                case 5:
                    findStatePopulationChange(countyData, numElems);
                    break;
                case 6:
                    findUSPopulationByYear(countyData, numElems);
                    break;
                case 7:
                    findUSPopulationChange(countyData, numElems);
                    break;
                case 8:
                    repeatRequest = false;
                    break;
                default:
                    message = "Please enter a valid choice from 1 to 8.";
                    JOptionPane.showMessageDialog(null, message);
            }

        }
        while (repeatRequest);
}

/**
 * This method finds searches for the FIPS code of a county, after prompting the user for
 * a county name and state code
 * This method was added for the convenience of the user
 *
 * @param countyArray CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 */
```

Great job with the menu driven interface.

```
public static void findFIPScode(CountyData[] countyArray, int numElems)
{
    String message;
    String stateCode, countyName, FIPScode;
    boolean validCode = false;

    // get user input
    do      // used to validate state code has 2 letters
    {
        message = "You have selected Find county FIPS code.\n\n" +
                "Please enter a 2 letter state code.";

        stateCode = Dialogue.inputString(message);          // get String using Dialogue class

        if (stateCode.length() != 2)
        {
            message = "Please enter a proper 2 letter state code.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            validCode = false;
        } else
        {
            validCode = true;
        }

    } while (!validCode);

    stateCode = stateCode.toUpperCase();                // convert state code to upper case

    message = "You have selected Find county FIPS code.\n" +
            "State code: " + stateCode +
            "\nPlease enter the county name";

    countyName = Dialogue.inputString(message);         // get String using Dialogue class

    // calculate results based on user input
    FIPScode = searchForFIPSCode(countyArray, numElems, stateCode, countyName);

    // display results to user
    message = "You have selected Find county FIPS code.\n" +
            "State Code: " + stateCode +
            "\nCounty Name: " + countyName + "\n\n";

    if (FIPScode.equals("00000"))                       // county and state not found
    {
        message += countyName + " " + stateCode + " was not found.";
        JOptionPane.showMessageDialog(null, message, "ERROR",
                JOptionPane.ERROR_MESSAGE);

    }
    else                                                // found FIPS code
    {
        message += "The FIPS code is " + FIPScode;
        JOptionPane.showMessageDialog(null, message);
    }
}

/**
 * This method prompts the user for a FIPS code and year and displays the population
 * @param countyArray CountyData array: used to store the population data
```

```
 * @param numElems int: number of elements in the array
 */
public static void findCountyPopulationByYear(CountyData[] countyArray, int numElems)
{
    String message;
    String FIPScode;
    int year, population;
    boolean isValid = false;

    // get user input
    do  // used to validate code has 5 digits
    {
        message = "You have selected County population by year.\n\n" +
                "Please enter a 5 digit FIPS code.";

        FIPScode = Dialogue.inputString(message);       // get String from user using Dialogue class

        if (FIPScode.length() != 5)
        {
            message = "Please enter a proper 5 digit FIPS code.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }
        else
        {
            isValid = true;
        }

    } while (!isValid);

    do  // used to validate year is valid
    {
        message = "You have selected County population by year.\n" +
                "FIPS code: " + FIPScode +
                "\n\nPlease enter a year from " + FIRST_YEAR + " to " + LAST_YEAR + ".";

        year = Dialogue.inputInt(message);              // get int from user using Dialogue class

        // if year entered had dropped first two digits but in range, still accept as valid
        if ( year >= (FIRST_YEAR-2000) && year <= (LAST_YEAR - 2000))
        {
            year += 2000;
        }

        if( year >= FIRST_YEAR && year <= LAST_YEAR)            // check if year is valid
        {
            isValid = true;
        }
        else
        {
            message = "The year you entered is not in range.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }

    }while(!isValid);

    // compute based on data
```

```
    population = searchForCountyPopulation(countyArray, numElems, FIPScode, year);

    // display results to user
    message = "You have selected County population by year.\n" +
            "FIPS code: " + FIPScode +
            "\nYear: " + year + "\n\n";

    if (population == -1)       // indicates data not found with FIPS code
    {
        message += "No county was found with FIPS code " + FIPScode + ".";
        JOptionPane.showMessageDialog(null, message, "ERROR",
                JOptionPane.ERROR_MESSAGE);
    }
    else                        //  population data found
    {
        message += "The population for " + findCountyName(countyArray, numElems, FIPScode) + " in " + year +
                "\nis " + String.format("%,d",population) + " people.";

        JOptionPane.showMessageDialog(null, message);
    }
}

/**
 * This method prompts the user to enter an FIPS code and starting and ending year and will
 * compute the population change between the two years.
 * @param countyArray CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 */
public static void findCountyPopulationChange(CountyData[] countyArray, int numElems)
{
    String message;
    String FIPScode;
    int startYear, endYear, startPopulation, endPopulation, changeOfPopulation;
    boolean isValid = false;

    // get user input
    do  // used to validate code has 5 digits
    {
        message = "You have selected County population change.\n\n" +
                "Please enter a 5 digit FIPS code.";

        FIPScode = Dialogue.inputString(message);           // get String from user using Dialogue class

        if (FIPScode.length() != 5)
        {
            message = "Please enter a proper 5 digit FIPS code.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }
        else                                                Error checking very thorough.
        {
            isValid = true;
        }

    } while (!isValid);

    do  // used to validate first year is valid
    {
            message = "You have selected County population change.\n" +
```

```
                "FIPS code: " + FIPScode +
                "\n\nPlease enter the first year.\n" +
                "Valid years are " + FIRST_YEAR + " to " + (LAST_YEAR-1) + ".";

        startYear = Dialogue.inputInt(message);          // get int from user using Dialogue class


    // if year entered had dropped first two digits but in range, still accept as valid
        if (startYear >= (FIRST_YEAR - 2000) && startYear <= (LAST_YEAR - 1 - 2000))
        {
            startYear += 2000;
        }

        if ( startYear >= FIRST_YEAR && startYear <= LAST_YEAR - 1)            // check if year is valid
        {
            isValid = true;
        }
        else
        {
            message = "The year you entered is not in range.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }

} while (!isValid);

do  // used to validate last year is valid
{
    message = "You have selected County population change.\n" +
            "FIPS code: " + FIPScode +
            "\nStart year: " + startYear +
            "\n\nPlease enter the last year.\n" +
            "Valid years are " + (startYear + 1) + " to " + LAST_YEAR + ".";

    endYear = Dialogue.inputInt(message);               // get int from user using Dialogue class

    // if year entered had dropped first two digits but in range, still accept as valid
    if (endYear >= (startYear + 1 - 2000) && endYear <= (LAST_YEAR - 2000))
    {
        endYear += 2000;
    }

    if ( endYear >= (startYear + 1) && endYear <= LAST_YEAR)            // check if year is valid
    {
        isValid = true;
    }
    else
    {
        message = "The year you entered is not in range.";
        JOptionPane.showMessageDialog(null, message, "ERROR",
                JOptionPane.ERROR_MESSAGE);
        isValid = false;
    }

} while (!isValid);

// compute based on data
startPopulation = searchForCountyPopulation(countyArray, numElems, FIPScode, startYear);
endPopulation = searchForCountyPopulation(countyArray, numElems, FIPScode, endYear);
```

```
    changeOfPopulation = endPopulation - startPopulation;

    // display results to user
    message = "You have selected County population change.\n" +
            "FIPS code: " + FIPScode +
            "\nStart year: " + startYear +
            "\nEnd year: " + endYear + "\n\n";

    if (startPopulation == -1)      // this happens when the FIPS code is not located
    {
        message += "No county was found with the FIPS code " + FIPScode + ".";
        JOptionPane.showMessageDialog(null, message, "ERROR",
                JOptionPane.ERROR_MESSAGE);
    }
    else                            // display population change
    {
        message += "The population for " + findCountyName(countyArray, numElems, FIPScode) + " in " + startYear +
                "\nis " + String.format("%,d",startPopulation) + " people.\n";
        message += "The population for " + findCountyName(countyArray, numElems, FIPScode) + " in " + endYear +
                "\nis " + String.format("%,d",endPopulation) + " people.\n\n";
        message += "The population change from " + startYear + " to " + endYear +
                "\nis " + String.format("%,d",changeOfPopulation) + " people.";

        JOptionPane.showMessageDialog(null, message);
    }
}

/**
 * This method prompts the user for a state code and year and displays the state population
 * for that year
 * @param countyArray CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 */

public static void findStatePopulationByYear(CountyData[] countyArray, int numElems)
{
    String message;
    String stateCode;
    int year, population;
    boolean isValid = false;

    // get user input
    do  // used to validate state code has 2 characters
    {
        message = "You have selected State population by year.\n\n" +
                "Please enter a 2 letter state code.";

        stateCode = Dialogue.inputString(message);          // get String from user using Dialogue class

        if (stateCode.length() != 2)
        {
            message = "Please enter a proper 2 letter state code.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }
        else
        {
            isValid = true;
        }
```

```
    } while (!isValid);

    stateCode = stateCode.toUpperCase();        // make certain code is in uppercase letters

    do  // used to validate year is valid
    {
        message = "You have selected State population by year.\n" +
                "State code: " + stateCode +
                "\n\nPlease enter a year from " + FIRST_YEAR + " to " + LAST_YEAR + ".";

        year = Dialogue.inputInt(message);      // get int from user using Dialogue class

        // if year entered had dropped first two digits but in range, still accept as valid
        if ( year >= (FIRST_YEAR-2000) && year <= (LAST_YEAR - 2000))
        {
            year += 2000;
        }

        if( year >= FIRST_YEAR && year <= LAST_YEAR)            // check if year is valid
        {
            isValid = true;
        }
        else
        {
            message = "The year you entered is not in range.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }


    }while(!isValid);

    // compute based on data
    population = searchForStatePopulation(countyArray, numElems, stateCode, year);

    // display results to user
    message = "You have selected State population by year.\n" +
            "State code: " + stateCode +
            "\nYear: " + year + "\n\n";

    if (population == -1)               // no data with state code
    {

        message += "No state was found with the state code " + stateCode + ".";
        JOptionPane.showMessageDialog(null, message, "ERROR",
                JOptionPane.ERROR_MESSAGE);
    }
    else                                // display population for state
    {
        message += "The population for the state " + stateCode + " in " + year +
                "\nis " + String.format("%,d",population) + " people.";

        JOptionPane.showMessageDialog(null, message);
    }
}

/**
 * This method prompts the user for a state code and starting and ending years and displays the state
```

```
 * population and population change between those years.
 *
 * @param countyArray CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 */
public static void findStatePopulationChange(CountyData[] countyArray, int numElems)
{
    String message;
    String stateCode;
    int startYear, endYear,  startPopulation, endPopulation, changeOfPopulation;
    boolean isValid = false;

    // get user input
    do  // used to validate state code has 2 characters
    {
        message = "You have selected State population change.\n\n" +
                "Please enter a 2 letter state code.";

        stateCode = Dialogue.inputString(message);       // get String from user using Dialogue class

        if (stateCode.length() != 2)
        {
            message = "Please enter a proper 2 letter state code.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }
        else
        {
            isValid = true;
        }

    } while (!isValid);

    stateCode = stateCode.toUpperCase();        // make certain code is in uppercase letters

    do  // used to validate first year is valid
    {
        message = "You have selected State population change.\n" +
                "State code: " + stateCode +
                "\n\nPlease enter the first year.\n" +
                "Valid years are " + FIRST_YEAR + " to " + (LAST_YEAR-1) + ".";

        startYear = Dialogue.inputInt(message);      // get int from user using Dialogue class

        // if year entered had dropped first two digits but in range, still accept as valid
        if (startYear >= (FIRST_YEAR - 2000) && startYear <= (LAST_YEAR - 1 - 2000))
        {
            startYear += 2000;
        }

        if ( startYear >= FIRST_YEAR && startYear <= LAST_YEAR - 1)            // check if year is valid
        {
            isValid = true;
        }
        else
        {
            message = "The year you entered is not in range.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
```

```
        isValid = false;
    }

} while (!isValid);

do  // used to validate last year is valid
{
    message = "You have selected County population change.\n" +
            "State code: " + stateCode +
            "\nStart year: " + startYear +
            "\n\nPlease enter the last year.\n" +
            "Valid years are " + (startYear + 1) + " to " + LAST_YEAR + ".";

    endYear = Dialogue.inputInt(message);           // get int from user using Dialogue class

    // if year entered had dropped first two digits but in range, still accept as valid
    if (endYear >= (startYear + 1 - 2000) && endYear <= (LAST_YEAR - 2000))
    {
        endYear += 2000;
    }

    if ( endYear >= (startYear + 1) && endYear <= LAST_YEAR)            // check if year is valid
    {
        isValid = true;
    }
    else
    {
        message = "The year you entered is not in range.";
        JOptionPane.showMessageDialog(null, message, "ERROR",
                JOptionPane.ERROR_MESSAGE);
        isValid = false;
    }

} while (!isValid);
// compute based on data
startPopulation = searchForStatePopulation(countyArray, numElems, stateCode, startYear);
endPopulation = searchForStatePopulation(countyArray, numElems, stateCode, endYear);
changeOfPopulation = endPopulation - startPopulation;

// display results to user
message = "You have selected State population by year.\n" +
        "State code: " + stateCode +
        "\nStart year: " + startYear +
        "\nEnd year: " + endYear + "\n\n";

if (startPopulation == -1)          // indicates state code not found
{

    message += "No state was found with the state code " + stateCode + ".";
    JOptionPane.showMessageDialog(null, message, "ERROR",
            JOptionPane.ERROR_MESSAGE);
}
else
{
    message += "The population for the state " + stateCode + " in " + startYear +
            "\nis " + String.format("%,d",startPopulation) + " people." +
            "\nThe population for the state " + stateCode + " in " + endYear +
            "\nis " + String.format("%,d",endPopulation) + " people." +
            "\n\nThe population change from " + startYear + " to " + endYear +
            "\nis " + String.format("%,d",changeOfPopulation) + " people.";
```

```
        JOptionPane.showMessageDialog(null, message);
    }
}

/**
 * This method prompts the user for a year then displays the US population for that year.
 * @param countyArray CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 */
public static void findUSPopulationByYear(CountyData[] countyArray, int numElems)
{
    String message;
    int year, population;
    boolean isValid = false;

    // get user input

    do  // used to validate year is valid
    {
        message = "You have selected US population by year.\n" +
               "\n\nPlease enter a year from " + FIRST_YEAR + " to " + LAST_YEAR + ".";

        year = Dialogue.inputInt(message);              // get int from user using Dialogue class

        // if year entered had dropped first two digits but in range, still accept as valid
        if ( year >= (FIRST_YEAR-2000) && year <= (LAST_YEAR - 2000))
        {
            year += 2000;
        }

        if( year >= FIRST_YEAR && year <= LAST_YEAR)            // check if year is valid
        {
            isValid = true;
        }
        else
        {
            message = "The year you entered is not in range.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }
    }while(!isValid);

    // compute based on data
    population = searchForUSPopulation(countyArray, numElems, year);

    // display results to user
    message = "You have selected US population by year.\n" +
              "\nYear: " + year + "\n\n";

    message += "The population for the US in " + year +
                "\nis " + String.format("%,d",population) + " people.";

    JOptionPane.showMessageDialog(null, message);
}

/**
 * This method prompts the user for a starting and ending year then displays the US population change
 * for those years and the change between years.
```

```
 * @param countyArray CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 */
public static void findUSPopulationChange(CountyData[] countyArray, int numElems)
{
    String message;
    int startYear, endYear, startPopulation, endPopulation, changeOfPopulation;
    boolean isValid = false;

    // get user input
    do  // used to validate first year is valid
    {
        message = "You have selected US population change.\n" +
                "\nPlease enter the first year.\n" +
                "Valid years are " + FIRST_YEAR + " to " + (LAST_YEAR-1) + ".";

        startYear = Dialogue.inputInt(message);

        // if year entered had dropped first two digits but in range, still accept as valid
        if (startYear >= (FIRST_YEAR - 2000) && startYear <= (LAST_YEAR - 1 - 2000))
        {
            startYear += 2000;
        }

        if ( startYear >= FIRST_YEAR && startYear <= LAST_YEAR - 1)            // check if year is valid
        {
            isValid = true;
        }
        else
        {
            message = "The year you entered is not in range.";
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }

    } while (!isValid);

    do  // used to validate last year is valid
    {
        message = "You have selected US population change.\n" +
                "\nStart year: " + startYear +
                "\n\nPlease enter the last year.\n" +
                "Valid years are " + (startYear + 1) + " to " + LAST_YEAR + ".";

        endYear = Dialogue.inputInt(message);

        // if year entered had dropped first two digits but in range, still accept as valid
        if (endYear >= (startYear + 1 - 2000) && endYear <= (LAST_YEAR - 2000))
        {
            endYear += 2000;
        }

        if ( endYear >= (startYear + 1) && endYear <= LAST_YEAR)            // check if year is valid
        {
            isValid = true;
        }
        else
        {
            message = "The year you entered is not in range.";
```

```
            JOptionPane.showMessageDialog(null, message, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
            isValid = false;
        }

    } while (!isValid);
    // compute based on data
    startPopulation = searchForUSPopulation(countyArray, numElems, startYear);
    endPopulation = searchForUSPopulation(countyArray, numElems, endYear);
    changeOfPopulation = endPopulation - startPopulation;

    // display results to user
    message = "You have selected US population change.\n" +
            "Start year: " + startYear +
            "\nEnd year: " + endYear+ "\n\n";

    message += "The population for the US in " + startYear +
            "\nis " + String.format("%,d",startPopulation) + " people." +
            "\nThe population for the US in " + endYear +
            "\nis " + String.format("%,d",endPopulation) + " people." +
            "\n\nThe population change from " + startYear + " to " + endYear +
            "\nis " + String.format("%,d",changeOfPopulation) + " people.";

    JOptionPane.showMessageDialog(null, message);
}

/**
 * This method searches for a FIPS code given a particular state and county
 * @param countyArray CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 * @param state String: state code to search for
 * @param county String: county to search for
 * @return String: FIPS code or "00000" if not found
 */
public static String searchForFIPSCode(CountyData[] countyArray, int numElems, String state, String county)
{
    String FIPScode="00000";
    boolean found = false;

    if (!county.endsWith("County"))     // if user did not end input with County, add it for search
    {
        county += " County";
    }

    for (int i=0; i<numElems && !found; i++)    // search array and exit if element found
    {
        if (state.equals(countyArray[i].getStateCode()) && county.equalsIgnoreCase(countyArray[i].getCountyName()))
        {
            FIPScode = countyArray[i].getFIPScode();
            found = true;
        }
    }

    return FIPScode;
}

/**
 * This method searches for a given county FIPS code and returns the population for the county given
 * a particular year.
 * @param countyArray CountyData array: used to store the population data
```

```
 * @param numElems int: number of elements in the array
 * @param FIPScode String: FIPS code for county being searched for
 * @param year int: year to find population data for
 * @return int: the population for the county with the given FIPS code or -1 if not found
 */
public static int searchForCountyPopulation(CountyData[] countyArray, int numElems, String FIPScode, int year)
{
    int pop = -1;
    boolean found = false;

    for (int i=0; i<numElems && !found; i++)
    {
        if (FIPScode.equals(countyArray[i].getFIPScode()))
        {
            pop = countyArray[i].getPopulation(year);
            found = true;
        }
    }
    return pop;
}

/**
 * This method searches for a county name given a particular FIPS code
 * @param countyArray CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 * @param FIPScode String: FIPS code for county being searched for
 * @return String: The county name, or "Not Found" if not found
 */
public static String findCountyName(CountyData[] countyArray, int numElems, String FIPScode)
{
    boolean found = false;
    String county = "Not Found";

    for (int i=0; i<numElems && !found; i++)
    {
        if (FIPScode.equals(countyArray[i].getFIPScode()))
        {
            county = countyArray[i].getCountyName();
            found = true;
        }
    }

    return county;
}

/**
 * This method searches for counties with a particular stateCode then totals up the population for
 * all those counties to determine the state population
 * @param countyArray CountyData array: used to store the population data
 * @param numElems int: number of elements in the array
 * @param stateCode String: code representing the state that the population is being searched for
 * @param year int: year that the population is being requested for
 * @return int: the population for the state for a given year, or -1 if the state code is not found
 */
public static int searchForStatePopulation(CountyData[] countyArray, int numElems, String stateCode, int year)
{
    int popSum = 0;            // accumulator
    boolean found = false;

    for (int i=0; i<numElems; i++)
```

```
        {
            if (stateCode.equals(countyArray[i].getStateCode()))
            {
                popSum += countyArray[i].getPopulation(year);
                found = true;
            }
        }

        if ( popSum ==0 )            // if no array elements match state code
        {
            popSum = -1;            // change to -1 to indicate not found
        }

        return popSum;
    }

    /**
     * This method calculates the population for the entire United States for a given year
     * @param countyArray CountyData array: used to store the population data
     * @param numElems int: number of elements in the array
     * @param year int: year that the population is being requested for
     * @return int: the entire population that is uploaded into the array
     */
    public static int searchForUSPopulation(CountyData[] countyArray, int numElems, int year)
    {
        int popSum = 0;            // accumulator

        for (int i=0; i<numElems; i++)
        {
            popSum += countyArray[i].getPopulation(year);
        }

        return popSum;
    }

}
```

```
/*********************************************************************************************
 *  This class stores county FIPS code, state and population data.
 *
 *  The user is requested to choose a menu option, then enter the data required to fulfill the request.
 *
 *  The program consists mostly of getters and setters,
 *  however, outside of the constructor, an individual year needs to be to be chosen to get or set
 *  a population number
 *
 *  CST 183 Programming Assignment 7
 *  @author Michael Clinesmith
 *********************************************************************************************/

public class CountyData
{
    // class constants
    private final int FIRST_YEAR = 2010, LAST_YEAR = 2017;
    private final int YEARS_STORED = LAST_YEAR - FIRST_YEAR + 1;

    // class fields
    private String FIPScode;
    private String countyName, stateCode;

    private int[] population = new int[YEARS_STORED];   √

    /**
     * No parameter constructor
     */
    public CountyData()
    {
        FIPScode = "00000";
        countyName = "None";
        stateCode = "NA";

        for (int i = 0; i<YEARS_STORED; i++)
        {
            population[i] = 0;
        }
    }

    /**
     * Constructor with parameters
     * @param code String: County FIPS code
     * @param county String: County name
     * @param state String: State code
     * @param popData int array: contains population data
     */
    public CountyData(String code, String county, String state, int[] popData)
    {
        FIPScode = code;
        countyName = county;
        stateCode = state;

        if (popData.length >= YEARS_STORED)         // if complete data, fill entire population array with data
        {
            for (int i = 0; i < YEARS_STORED; i++)
            {
                population[i] = popData[i];
            }
```

```
    }
    else                                        // if not enough data, fill extra array elements with 0s
    {
        for (int i = 0; i < popData.length; i++)
        {
            population[i] = popData[i];
        }
        for (int i = popData.length; i < YEARS_STORED; i++)
        {
            population[i] = 0;
        }
    }

}

/**
 * Mutator method to set the FIPS code
 * @param FIPScode String: FIPS code of a county
 */
public void setFIPScode(String FIPScode)
{
    this.FIPScode = FIPScode;
}

/**
 * Mutator method to set the county name
 * @param countyName String: A county name
 */
public void setCountyName(String countyName)
{
    this.countyName = countyName;
}

/**
 * Mutator method to set the state code
 * @param stateCode String: A state code representing the state a county is in
 */
public void setStateCode(String stateCode)
{
    this.stateCode = stateCode;
}

/**
 * Mutator method to set the county population for a particular year
 * @param year int: The year to set the population for
 * @param pop int: The population in a particular year
 */
public void setPopulation( int year, int pop)
{
    if (year >= FIRST_YEAR && year <= LAST_YEAR)        // make certain year is in proper range
    {
        population[year-FIRST_YEAR] = pop;
    }
}

/**
 * Accessor method to get a county's FIPS code
 * @return String: The FIPS code of a county
 */
public String getFIPScode()
```

```
    {
        return FIPScode;
    }

    /**
     * Accessor method to get a county's name
     * @return String: The name of a county
     */
    public String getCountyName()
    {
        return countyName;
    }

    /**
     * Accessor method to get a county's state code
     * @return String: The state code of a county
     */
    public String getStateCode()
    {
        return stateCode;
    }

    /**
     * Accessor method to get a county's population in a particular year
     * @param year int: a year
     * @return int: the county's population in that year
     */
    public int getPopulation( int year)
    {
        int pop = -1;
        if (year >= FIRST_YEAR && year <= LAST_YEAR)        // make certain year is in proper range
        {
            pop = population[year-FIRST_YEAR];
        }

        return pop;
    }

    /**
     * Method to return a string representing the data stored in a CountyData obuect
     * @return String: Contains the FIPS code, county name, state code, and population data of a CountyData object
     */
    @Override
    public String toString()
    {
        String data;
        data = "FIPS Code: " + FIPScode +
                "\nCounty Name: " + countyName +
                "\nState: " + stateCode + "\n";

        for (int i = 0; i<YEARS_STORED; i++)
        {
            data += "" + (FIRST_YEAR + i) + " population: " + population[i] + "\n";
        }

        return data;
    }

}
```

```
/*******************************************************************************
 * This class contains methods that do error checking to get valid input from dialogue boxes
 *
 * It handles requests for char, int, double and String
 * The program will reask a question for input if the following things happen:
 *      the user enters the wrong type of data,
 *      the user enters no data
 *      the user cancels the dialogue box
 *
 * @author  Michael Clinesmith
 *******************************************************************************/

import javax.swing.JOptionPane;

public class Dialogue
{
    /**
     *  inputChar() gets a char of input from a user using dialog box
     *  This method validates the input to prevent runtime errors
     *
     * @param outputString  The string to be displayed to the user to get the required char
     * @return  a char from the first character entered by the user
     */
    public static char inputChar (String outputString)
    {
        String input;
        String messageString;
        char inputChar = ' ';
        boolean isValid = false;

        while (!isValid)
        {

            input = JOptionPane.showInputDialog(outputString);
            if (input == null)
            {
                messageString = "Proper input not entered.\n" +
                        "Please enter input in the correct format.";

                JOptionPane.showMessageDialog(null, messageString, "ERROR",
                        JOptionPane.ERROR_MESSAGE);
            }
            else if (input.length()==0)
            {
                messageString = "Proper input not entered.\n" +
                        "Please enter input in the correct format.";

                JOptionPane.showMessageDialog(null, messageString, "ERROR",
                        JOptionPane.ERROR_MESSAGE);
            }
            else
            {
                inputChar = input.charAt(0);
                isValid = true;
            }

        }
        return inputChar;
    }
```

```
/**
 *  inputInt() gets a int of input from a user using dialog box
 *  This method validates the input to prevent runtime errors
 *
 * @param outputString  The string to be displayed to the user to get the required int
 * @return  an int entered by the user
 */
public static int inputInt (String outputString)
{
    String input;
    String messageString;
    int inputInt = 0;
    boolean isValid = false;

    while (!isValid)
    {

        try                     // used to catch bad input data exception
        {
            input = JOptionPane.showInputDialog(outputString);
            if (input == null)
            {
                messageString = "Proper input not entered.\n" +
                        "Please enter input in the correct format.";

                JOptionPane.showMessageDialog(null, messageString, "ERROR",
                        JOptionPane.ERROR_MESSAGE);
            } else if (input.length() == 0)
            {
                messageString = "Proper input not entered.\n" +
                        "Please enter input in the correct format.";

                JOptionPane.showMessageDialog(null, messageString, "ERROR",
                        JOptionPane.ERROR_MESSAGE);
            } else
            {
                inputInt = Integer.parseInt(input);
                isValid = true;
            }
        }
        catch (NumberFormatException e)     // catches exception where user inputs improperly formatted data
        {
            messageString = "Proper input not entered.\n" +
                    "Please enter input in the correct format.";

            JOptionPane.showMessageDialog(null, messageString, "ERROR",
                    JOptionPane.ERROR_MESSAGE);

        }
    }
    return inputInt;
}

/**
 *  inputDouble() gets a double of input from a user using dialog box
 *  This method validates the input to prevent runtime errors
 *
 * @param outputString  The string to be displayed to the user to get the required double
 * @return  a double entered by the user
 */
```

```
public static double inputDouble (String outputString)
{
    String input;
    String messageString;
    double inputDouble = 0.0;
    boolean isValid = false;

    while (!isValid)
    {
        try                 // used to catch bad input data exception
        {
            input = JOptionPane.showInputDialog(outputString);
            if (input == null)
            {
                messageString = "Proper input not entered.\n" +
                        "Please enter input in the correct format.";

                JOptionPane.showMessageDialog(null, messageString, "ERROR",
                        JOptionPane.ERROR_MESSAGE);
            } else if (input.length() == 0)
            {
                messageString = "Proper input not entered.\n" +
                        "Please enter input in the correct format.";

                JOptionPane.showMessageDialog(null, messageString, "ERROR",
                        JOptionPane.ERROR_MESSAGE);
            } else
            {
                inputDouble = Double.parseDouble(input);
                isValid = true;
            }
        }
        catch (NumberFormatException e)     // catches exception where user inputs improperly formatted data
        {
            messageString = "Proper input not entered.\n" +
                    "Please enter input in the correct format.";

            JOptionPane.showMessageDialog(null, messageString, "ERROR",
                    JOptionPane.ERROR_MESSAGE);
        }
    }
    return inputDouble;
}
/**
 *  inputString() gets a String of input from a user using dialog box
 *  This method validates the input to prevent runtime errors
 *
 * @param outputString  The string to be displayed to the user to get the required String
 * @return  a String entered by the user
 */
public static String inputString (String outputString)
{
    String input;
    String messageString;
    String inputStr = "";
    boolean isValid = false;

    while (!isValid)
    {
```

```
            input = JOptionPane.showInputDialog(outputString);
            if (input == null)
            {
                messageString = "Proper input not entered.\n" +
                        "Please enter input in the correct format.";

                JOptionPane.showMessageDialog(null, messageString, "ERROR",
                        JOptionPane.ERROR_MESSAGE);
            }
            else if (input.length()==0)
            {
                messageString = "Proper input not entered.\n" +
                        "Please enter input in the correct format.";

                JOptionPane.showMessageDialog(null, messageString, "ERROR",
                        JOptionPane.ERROR_MESSAGE);
            }
            else
            {
                inputStr = input;        // gives inputStr the same address as input, which is okay
                isValid = true;
            }

        }
        return inputStr;
    }

}
```