```java
/* Assignment 10: Calculate payroll data using classes

   This program calculates and sorts employee payroll data
   from in input txt file. It take the data and writes it to
   an output file, and echos the information to the console.

   The program uses an array of a class and a seperate class
   to create employee parameters. This program uses methods to
   print headers, calculate gross pay, net pay, wealth (amount
   of money in IRA investment and savings account), Taxes (Fed-
   eral and State). Uses a method to print a data table. The
   program also calculates totals from each of the above areas,
   and calculates the average of the pay rates.

   This program also uses an exception with a method. It will
   throw an IndexOutOfBoundsException if the input file has
   too many employees to process.

   The program sorts the data three seperate ways. Order it was
   entered, pay rate, and gross pay ascending.

   Zachary Stall
   Program #10, CS 1050, Section 2
   JGrasp, Custom PC, Windows 10

   Megillah - A lengthy and tediously complicated situation or
   matter.

   "People won't have time for yo uif you are always angry or
   complaining."
   -Steven Hawking (January 8, 1942)

*/

import java.io.*;
import java.util.Scanner;

public class ZacharyStall_2_10 {

   // Accessing the Toolkit for formatting
   static Toolkit tools = new Toolkit();

   public static void main (String [] args)throws IOException{

      // Instantiating EmpoyeeParemeters() to set up
      // parameters for employees in out input file
      EmployeeParameters emplParams = new EmployeeParameters();

      // Access the input and output files
      final String INPUT_FILE  = "ZacharyStall_2_10_input.txt";
      final String OUTPUT_FILE = "ZacharyStall_2_10_output.txt";


      int maxEmployees;    // Max employees, will be set by params
      int nElement = 0;    // Number a values read from input file
      int sortResult = 0;  // To sort data and catch issues
      double savingsRate;  // Percentage rate to be saved
      double iraRate;      // IRA percentage rate to be saved
      double fedRate;      // FED tax percentage rate
      double stateRate;    // State tax percentage rate
      String warning;      // if Input exceeds maxEmployees

      // Access the input/output files
      File inputDataFile = new File(INPUT_FILE);
      Scanner inputFile  = new Scanner(inputDataFile);

      FileWriter outputDataFile = new FileWriter(OUTPUT_FILE);
      PrintWriter outputFile = new PrintWriter(outputDataFile);

      // Begin program execution
      System.out.println("Reading  file " + INPUT_FILE  + "\r\n" +
                         "Creating file " + OUTPUT_FILE + "\r\n");

      // Get the program parameters from empl
      emplParams.getEmployeeParameters();

      // Store parameters in local vars
      maxEmployees = emplParams.maxEmployees;
      savingsRate  = emplParams.savingsRate;
      iraRate      = emplParams.iraRate;
      fedRate      = emplParams.federalWithholdingRate;
      stateRate    = emplParams.stateWithholdingRate;


      // Create Employee class array and set it's length based
      // on the EmployeeParaters
      Employee[] empl = new Employee[maxEmployees];


      // Dispay the parameters
      emplParams.displayEmployeeParameters();
      System.out.println();

      /* Fill array from input file and store number of emloyees
         processed into nElements
         Throws exception if input file has more employees than
         parameter allows. If too many employees in input file,
         program warns user and terminates
      */

      try {
         // Fill Array
         nElement = fillData(inputFile, empl);

      } catch (IndexOutOfBoundsException excpt) {
         // Print error warning message and terminate if too many
         // employees
         warning = "Warning, number of employees in input file\r\n" +
                "is larger than parameters allow. Too many employees" +
                "\r\n" + "in input file. PROGRAM TERMINATED.";

         outputFile.println(warning);
         System.out.println(warning);

         inputFile.close();
         outputFile.close();

         System.exit(0);

      } // End Try/Catch

      // Calculate the gross pay
      getGrossPay(empl, nElement);

      // Calculating all the savings and taxes
      getAllMoneyAmounts(empl,
                         iraRate,
                         fedRate,
                         stateRate,
                         savingsRate,
                         nElement);

      // Output all the data to the console and output file
      // Sorted by the order it was input
      outputMaster(outputFile, "Input", empl, nElement);

      // Sort the data by employees names
      sortResult = tools.selectionSortArrayOfClass(empl, nElement, "Name");
      outputMaster(outputFile, "Name", empl, nElement);

      // Sort the data by ascending gross pay
      sortResult = tools.selectionSortArrayOfClass(empl, nElement, "Gross Pay");
      outputMaster(outputFile, "Gross Pay", empl, nElement);

      // Close files
      inputFile.close();
      outputFile.close();

      // End program
      System.exit(0);

   } // End Main

   // ***********************************************************
   // Methods Methods Methods Methods Methods Methods Methods
   // ***********************************************************

   // fillData filles the Employee array from the Sanner file
   // and returns the number of data values input.
   public static int fillData(Scanner input,
                              Employee[] array)
                              throws IndexOutOfBoundsException {

      int nData = 0; // number of data points read to be returned

      //while (input.hasNext() && (nData < array.length)) {
      while(input.hasNext()) {
         array[nData] = new Employee();
         array[nData].hoursWorked = input.nextDouble();
         array[nData].payRate = input.nextDouble();
         array[nData].name = input.nextLine().trim();
         nData++;
      } // End while loop

      return nData;
   } // End fillData

   // ***********************************************************

   // Calculate the gross pay for employees
   public static void getGrossPay (Employee [] array, int nElements) {

      double hours         = 0.0;   // Hours worked
      double wage          = 0.0;   // Momey per hour
      double timeAndHalf   = 1.5;   // Over time: time and a half
      double doubleTime    = 2.0;   // Over time: double pay
      double moneyPaid     = 0.0;   // Dollar amount for hours worked

      for(int i = 0; i < nElements; i++) {

         hours = array[i].hoursWorked;
         wage = array[i].payRate;

         // Less than 40hrs normal pay
         if (hours <= 40) {
            array[i].grossPay = hours * wage;
         }

         // Between 40 and 50hrs time and a half
         else if (hours <= 50 && hours > 40) {
            array[i].grossPay = wage * (40 + (hours - 40) * timeAndHalf);
         }
         // Over 50 hours double time
         else if (hours > 50) {
```

```java
            array[i].grossPay = wage * (40 + 10 * 1.5 + (hours - 50) * doubleTime);
         }
      } // End for
   } // End getGrossPay
   // ************************************************************

   // Calculate the Savings amount
   public static void getAllMoneyAmounts(Employee[] array,
                                       double ira,
                                       double fedTax,
                                       double stateTax,
                                       double saveRate,
                                       int nElements) {

      double grossPay = 0.0;  // Gross pay amount for each employee
      double tax1 = 0.0;      // To convert taxes into decimals

      tax1 = (stateTax / 100.0) + (fedTax / 100.0);

      // Calculate and store all vars needed
      for(int i = 0; i < nElements; i++) {
         grossPay = array[i].grossPay;
         array[i].iraAmount = grossPay * (ira / 100.0);
         array[i].adjustedGrossPay = grossPay - array[i].iraAmount;
         array[i].taxAmount = array[i].adjustedGrossPay * (tax1);
         array[i].netPay = array[i].adjustedGrossPay - array[i].taxAmount;
         array[i].savingsAmount = array[i].netPay * (saveRate / 100.0);

      } // End For
   } // End getAllMoneyAmounts

   // ************************************************************

   // Print the headers for the table
   public static void printHeader(PrintWriter output, String order) {

      String str; // Store headers str to only type once

      str = // Input order
            "\r\nPrinted in " + order.toLowerCase() +
            " order.\r\n" + "\r\n" +
            // Table title
            tools.padString("Mobile Apps Galore, Inc. - Payroll Report", 65, " ", "") +
            "\r\n" + "\r\n" +
            // table headers
            tools.padString("Name", 21) +
            "   " + tools.padString("Gross Pay", 10) +
            "   " + tools.padString("Net Pay", 8) +
            "   " + tools.padString("Wealth", 10) +
            "   " + tools.padString("Taxes", 8) +
            "   " + tools.padString("Hours", 7) +
            "   " + tools.padString("Pay Rate", 0) +
            "   " + "\r\n" +
                  tools.padString("---------------", 21) +
            "   " + tools.padString("---------", 10) +
            "   " + tools.padString("-------", 8) +
            "   " + tools.padString("-------", 10) +
            "   " + tools.padString("-------", 8) +
            "   " + tools.padString("-------", 6) +
            "   " + tools.padString("--------", 0) +
            "   ";

      output.println(str);
      System.out.println(str);
   } // End printHeaders

   // ************************************************************

   // Calculate the totals
   public static void getTotals(Employee[] array, PrintWriter output, int nElements) {

      final String DOLLAR = "##,##0.00";

      String str;                 // Store message to be output
      double sumGrossPay = 0.0;  // Sum of gross pay
      double sumNetPay = 0.0;    // Sum of net pay
      double sumWealth = 0.0;    // Sum of Wealth
      double sumTaxes = 0.0;     // Sum of taxes
      double sumHours = 0.0;     // Sum of hours worked
      double sumPayRate = 0.0;   // Sum of pay rate to calc the avgPayRate
      double avgPayRate = 0.0;   // Average of the payrates

      // Store each of the array items in the local vars
      for(int i = 0; i < nElements; i++) {
         sumGrossPay += array[i].grossPay;
         sumNetPay   += array[i].netPay;
         sumWealth   += array[i].savingsAmount + array[i].iraAmount;
         sumTaxes    += array[i].taxAmount;
         sumHours    += array[i].hoursWorked;
         sumPayRate  += array[i].payRate;

      } // End for loop

      // Check to make sure there are payrates to calc avg
      if(sumPayRate >= 1) {
         avgPayRate = sumPayRate / nElements;
      } // End if statement

      // Print out all the sums and the average
      str = "Totals: " +
            tools.leftPad(sumGrossPay, 22, DOLLAR) +
            tools.leftPad(sumNetPay, 13, DOLLAR) +
            tools.leftPad(sumWealth, 11, DOLLAR) +
            tools.leftPad(sumTaxes, 12, DOLLAR) +
            tools.leftPad(sumHours, 11, DOLLAR) +
            "\r\n" + tools.padString("Average: ", 83, " ", "") +
            tools.leftPad(avgPayRate, 5, DOLLAR) +
            "\r\n\r\n" +
            "The total number of employees processed: " +
            nElements;

      System.out.println(str);
      output.println(str);

   } // End getTotals

   // ************************************************************

   // Run all the methods to output data
   public static void outputMaster(PrintWriter output,
                                   String order,
                                   Employee[] array,
                                   int nElement) {

      printHeader(output, order);
      outputData(array, output, nElement);
      getTotals(array, output, nElement);


   } // End outputMaster


   // ************************************************************

   // Print out data in a table
   public static void outputData(Employee[] array, PrintWriter output, int nEntries) {
```

```java

      final String DOLLAR = "##,##0.00";

      double wealth = 0.0;

      for(int i = 0; i < nEntries; i++) {

         wealth = array[i].savingsAmount + array[i].iraAmount;

         String str;

         str = tools.padString(array[i].name, 19) +
               "   " + tools.leftPad(array[i].grossPay, 8, DOLLAR) +
               "   " + tools.leftPad(array[i].netPay, 10, DOLLAR) +
               "   " + tools.leftPad(wealth, 8, DOLLAR) +
               "   " + tools.leftPad(array[i].taxAmount, 9, DOLLAR) +
               "   " + tools.leftPad(array[i].hoursWorked, 8, DOLLAR) +
               "   " + tools.leftPad(array[i].payRate, 8, DOLLAR) +
               "   ";

         output.println(str);
         System.out.println(str);

      } // End for loop
   } // End outputData

} // End Class
```