|  |  |  |  |
| --- | --- | --- | --- |
| **Instructor** |  | **Due Date** |  |

**PROJECT Writing to a Sequential File 50 Points**

**Objective** To write information to a data file.

***PROJECT DESCRIPTION***

The program code given in **Figure 1** which follows, requests payroll information from the user and writes the information into a text file. You’ll modify the code to use a

loop to allow the user to enter additional payroll records into a text file if they desire.

***Information about This Project***

This program illustrates an example of sequential file processing.

***Steps To Complete This Project***

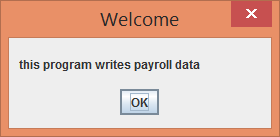
**STEP 1** **Open Eclipse or JCreator and Write the Code**

Open Eclipse or JCreator. From the Eclipse or JCreator main menu, click File , choose New and select Java Project . Name the project and Java file as

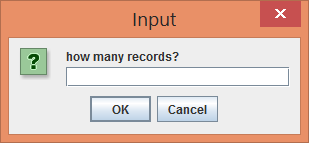
**CreateData** since this is the class name for this project. Click OK to close the **New** dialog box. When the editor opens, type the program code shown in

**Figure 1** ,which follows, exactly as it appears, except substitute your own name in place of Sammy Student.

The program code is written such that when the program is executed, the user is greeted with the message below.



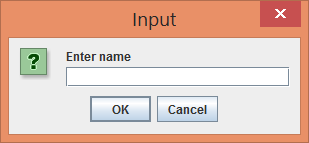
A method named **Write()** is then executed and prompts the user to enter the number of payroll records to be written to the data file.



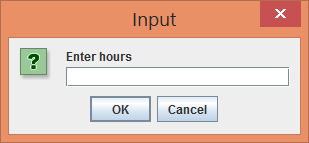
**PROJECT Writing to a Sequential File**

The user responds by entering a whole number, indicating the number of records to be written.

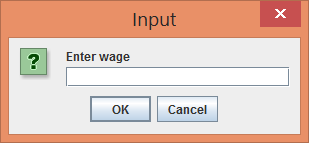
The name of an individual is then entered into the next input dialog box.



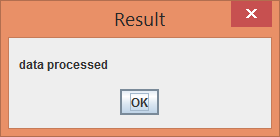
The individual’s number of hours worked is then entered into the next input dialog box.



The individual’s hourly wage is then entered into the next input dialog box.



Finally, the user is displayed this message indicating that the payroll records have been saved to file.



Again, into your Java text editor, copy the programs code from **Figure 1** .

**Figure 1 Program Code for Creating the Payroll File**

|  |
| --- |
|  |

|  |
| --- |
| **//Sammy Student: programmer**  **import** java.io.BufferedWriter;  **import** java.io.File;  **import** java.io.FileWriter;  **import** java.io.IOException;  **import** javax.swing.JOptionPane;  **public class CreateData {**    **public static void main(String[] args)**  **{**  **JOptionPane.showMessageDialog(null,**  **"this program writes payroll data",**  **"Welcome", JOptionPane.PLAIN\_MESSAGE);**    **Write();**  **}**  **static void Write()**  **{**  **try {**  **String firstLine, secondLine, thirdLine, number;**  **File check = new File("payroll.txt");**  **FileWriter file;**  **if(check.exists())**  **//allows appending of data to file**  **file = new FileWriter("payroll.txt", true);**  **else**  **file = new FileWriter("payroll.txt");**  **BufferedWriter buffer = new BufferedWriter(file);**  **int size, count = 1;**  **number = JOptionPane.showInputDialog("how many records?");**  **size = Integer.parseInt(number);**    **do {**  **firstLine = JOptionPane.showInputDialog("Enter name");**  **secondLine = JOptionPane.showInputDialog("Enter hours");**  **thirdLine = JOptionPane.showInputDialog("Enter wage");**  **buffer.write(firstLine);**  **buffer.newLine();**  **buffer.write(secondLine);**  **buffer.newLine();**  **buffer.write(thirdLine);**  **buffer.newLine();**  **count++;**  **}while(count <= size);** |

**Figure 1 Program Code for Creating the Payroll File ( continued )**

|  |
| --- |
|  |

|  |
| --- |
| **buffer.close();**  **JOptionPane.*showMessageDialog*(null, "data processed",**  **"Result", JOptionPane.*PLAIN\_MESSAGE* );**    **System.*exit*(1);**  **}**  **catch (IOException e) { System.*out*.println(e); }**    **}**  **}** |

|  |
| --- |
|  |

**STEP 2**  **Compile, Execute and Test the Program**

After you have typed the given program code, build and compile your program code by selecting Project from the main menu and choosing Build All .

If your program has not compiled correctly and you have errors, read the error messages and make any necessary corrections by comparing your code to that shown on the prior page. Then recompile your program.

If you have successfully compiled your program, run your program and when prompted, test your program with the following payroll records.

|  |  |  |
| --- | --- | --- |
| **employee name** | **hours** | **wage** |
| David Davies | 30 | 8.75 |
| Eddie Edwards | 40 | 7.75 |
| Fran Francis | 35 | 11.20 |
| Gina Georges | 22 | 6.30 |
| Hyacinth Ho | 40 | 9.21 |

**STEP 3**  **View the Text File**

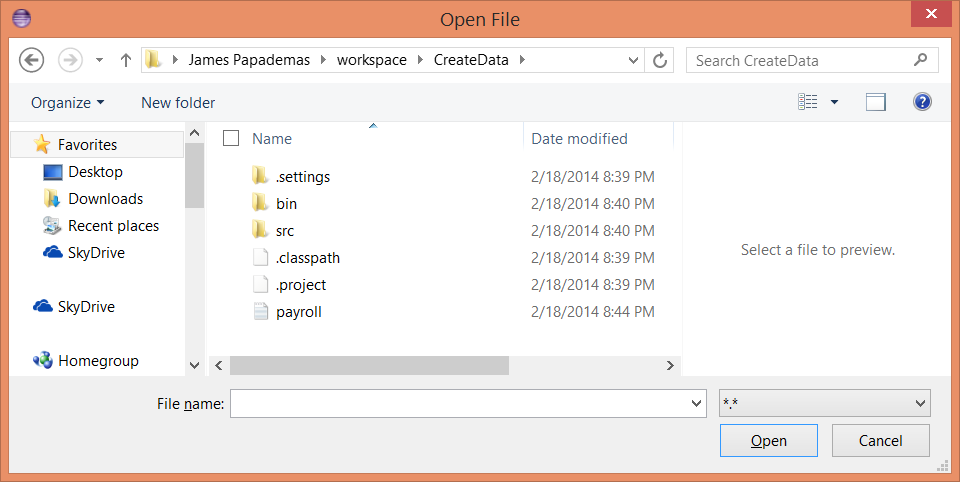
After you enter the above data, close the application and locate the data file that was created. To do this, return to Eclispe or JCreator and click File and select

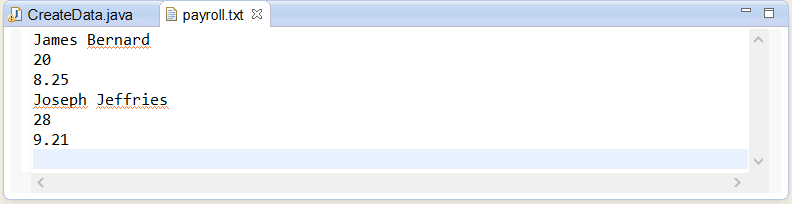
Open File .

If you saved your Java file in the default project location, then the **payroll.txt** data file will be located within your workspace folder. Double - click the file to open it within your Java Developer – snapshots follow…

Note the contents of the file.

**PROJECT Writing to a Sequential File**





**STEP 4 Modify Your Program Code**

The next step is to perform two types of modifications to the code. The first modification will allow the program not to crash if the user clicks OK in an

Input dialog box without entering any data or if the user clicks Cancel .

First add this modification to the first Input dialog box, namely the one that asks the user for the number of records to write to file. Then once you have determined that the modification works add similar code to all OF the Input dialog boxes.

To start this logic, navigate to the Code view of the **CreateData.Java** file.

Locate the **Write()** method and *change* the String declaration statement as follows:

**String firstLine, secondLine, thirdLine, number = "";**

Then insert the line of code

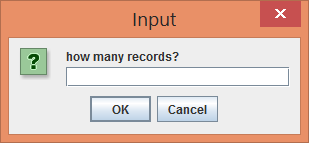
**while(number == null || number.equals(""))**

directly before the following statement:

|  |
| --- |
| **number = JOptionPane.showInputDialog("how many records?");** |

**PROJECT Writing to a Sequential File**

After adding the modifications, run your program and observe the action of your program when the user clicks Cancel or clicks OK without entering data in the Input box.



Close the application and similarly modify the program code for all the Input dialog boxes within your code. This forces the user to respond to the dialog boxes.

**STEP 5 Modify Again Your Program Code**

The next modification allows the user, without the program exiting, to enter additional records to the data file after previous records have been added within the same program run.

To do this, locate, within the **main()** method, the call to method **Write()** .

Add program code that will allow the method to be repeatedly called until the user decides to quit the application. The code for this modification is located within **Figure 2** , which follows. Change therefore your **main()** method code to correspond to **Figure 2** .

Also, to complete this modification, remove the following statement from the

**Write()** method.

**System.exit(1);**

**PROJECT Writing to a Sequential File**

**Figure 2 Program Code for main() Method**

|  |
| --- |
|  |

|  |
| --- |
| **public static void main(String[] args)**  **{**  **JOptionPane.showMessageDialog(null,**  **"this program writes payroll data",**  **"Welcome", JOptionPane.PLAIN\_MESSAGE );**    **int repeat = 1;**  **String answer;**    **do**  **{**  **Write();**  **answer = JOptionPane.showInputDialog("write payroll data?\n"**  **+ "enter 1 to continue or 0 to exit");**  **repeat = Integer.parseInt(answer);**  **}while(repeat == 1);**    **}** |
|  |

|  |
| --- |
|  |

**STEP 6**  **Compile, Execute and Test the Modified Program**

After you have typed the given program code, build and compile your program code by selecting Build from the main menu and choosing Compile File .

For credit **add** in the next two records to your data file. Also include a record with **your** name with hours and a wage as well. Take screen snapshots of these new input items as you enter each record data in.

Finally take a snapshot of your entire payroll.txt file which should consist of eight records total.

|  |  |  |
| --- | --- | --- |
| **name** | **hours** | **wage** |
| Isadora Irving | 27 | 9.60 |
| Jackson Jameson | 35 | 9.20 |

Submit the screen snapshots and your completed program code for credit.

**STEP 7**  **Prepare for the Next Project**

After you run the modified application, review the program code for this application. For a future project we will read the data in the payroll file via a separate Java file, and also create some reports.