Java Assessment

(Classroom Organiser)

**Student: Benjamin James Moore**

**Student Number: 472819131**

**Date: 16/11/2021**

Unit: ICTPRG418

Teacher: Teacher

Contents

[**IPO Chart 4**](#_heading=h.gjdgxs)

[**TOE Chart 5**](#_heading=h.30j0zll)

[**Stylised Data Flow 5**](#_heading=h.1fob9te)

[**Selected Pseudo Code 6**](#_heading=h.3znysh7)

[*Displaying and Saving Records 6*](#_heading=h.32hioqz)

[*DisplayRecord (CurrentRecord) 6*](#_heading=h.tyjcwt)

[*SaveRecord (CurrentRecord) 6*](#_heading=h.3dy6vkm)

[*File data handling methods – read and write file 7*](#_heading=h.1hmsyys)

[*ReadFile 7*](#_heading=h.4d34og8)

[*WriteFile 7*](#_heading=h.2s8eyo1)

[*Navigation Button Code 8*](#_heading=h.41mghml)

[*First 8*](#_heading=h.3rdcrjn)

[*Last 8*](#_heading=h.26in1rg)

[*Previous 8*](#_heading=h.lnxbz9)

[*Next 9*](#_heading=h.35nkun2)

[*Navigation Button Code 10*](#_heading=h.2grqrue)

[*New 10*](#_heading=h.44sinio)

[*Save 10*](#_heading=h.2jxsxqh)

[*Exit 10*](#_heading=h.z337ya)

[*Delete 10*](#_heading=h.3j2qqm3)

[*Find 11*](#_heading=h.1y810tw)

[**Structured Chart(s) 12**](#_heading=h.4i7ojhp)

[*Managing the GUI elements 12*](#_heading=h.2xcytpi)

[*Managing the user Actions 12*](#_heading=h.1ci93xb)

[**Class diagram 13**](#_heading=h.3whwml4)

[**Testing 14**](#_heading=h.2bn6wsx)

[*Test Plan 14*](#_heading=h.qsh70q)

[*Test Cases: 15*](#_heading=h.vx1227)

[**Evidence of Debugging 16**](#_heading=h.49x2ik5)

[**System generated program documentation 17**](#_heading=h.3fwokq0)

[**User Manual for: Sample IP Database 18**](#_heading=h.147n2zr)

[*Example 1 … (Partial sample only) 18*](#_heading=h.3o7alnk)

[*Example 2 … (Partial sample only) 19*](#_heading=h.23ckvvd)

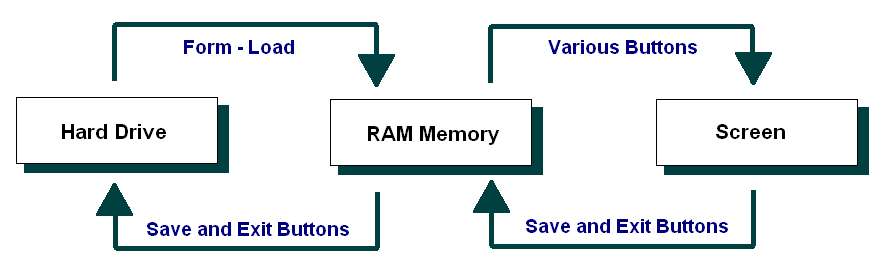
## **IPO Chart**

| **Input** | **Process** | **Output** | **Storage** |
| --- | --- | --- | --- |
| ‘*Records*’ (entries) from the data file:   * Teacher * Class * Room * Date * Student * Desk | * Open the file * Read the first entry * If not all entries have been loaded, load the next entry * Record the total number of entries read in. | * Show first record (entry) on screen | Data File  Temporary storage in RAM memory |
| * **Sort** button, | * sort array * display sorted array into separate window | * Sorted list from array | nill |
| * Search String and **Find** button | * Go through the record (entry) list until the Names matches the entry in the Search text field. | * Record (entry) is shown on screen |  |
| * Record (entry) to be deleted presented on screen and then the **Delete** button. | * For all entries after the current entry, move the other entries up the list, therefore removing the current entry. * Set the number of entries in the list to current number – 1. | Present the next record (entry) after the deleted one. | Updated records temporarily stored in RAM memory |
| * **Save** button | * Save all the entries in RAM Memory into the specified output file. | Export of all records (entries). | All entries saved in a new copy of the data file. |
| * **Exit** button | * Save all data (as per Save button) and… * Close program | Export of all records (entries). | All entries saved in a new copy of the data file. |
| * **Open** button | * Opens windows file dialog * Obtains selected File * Displays File in text fields | Display file data in text fields | all data saved to the array |

## **TOE Chart**

| **Task** | **Object** | **Event** |
| --- | --- | --- |
| Load Buttons and data grid | Form | Open |
| Clear screen ready for new entry | **Clear** button | Click |
| Save the current entry and then all entries to a clean copy of the data file | **Save** button | Click |
| Sort Entries by name | **Sort** button | Click |
| Delete the current entry | **Delete** button | Click |
| Search all users where name matches search field | **Search** button | Click |
| Exits Application | **Exit** Button | Click |

## **Stylised Data Flow**



## **Selected Pseudo Code**

### **Displaying and Saving Records**

In line with our earlier data flow diagram and structured chart, we might start with a couple of the supporting methods that are called repeatedly.

In particular, let’s plan the method that Displays a Record, and the one that updates a record in RAM memory.

The pseudo code for these might be as follows:

### **DisplayRecord (CurrentRecord)**

Copy the Name for the currentRecord to the indexed text field

Copy the Desk for the currentRecord to the indexed text field

Copy the TeacherName for the currentRecord to the Teacher Name text field

Copy the Class for the currentRecord to the Class text field

Copy the Room for the currentRecord to the Room text field

Copy the Date for the currentRecord to the Date text field

### **SaveRecord (CurrentRecord)**

Copy the Name from the indexed text field to the currentRecord of the Name Array

Copy the Desk from the indexed text field to the currentRecord of the Desk Array

Copy the TeacherName from the TeacherName text field to the currentRecord of the TeacherName Array

Copy the Class from the Class text field to the currentRecord of the Class Array

Copy the Room from the Room text field to the currentRecord of the Room Array

Copy the Date from the Date text field to the currentRecord of the Date Array

### **File data handling methods – read and write file**

Next let’s turn to the file handling methods.

The pseudo code for these might be as follows:

### **ReadFile**

If there is an error, then go to the ErrorHandler

Open the data file ready for reading

Set counter = 0

While there is more data

Read in a line of data

Put the entries in their respective arrays

Set counter = counter + 1

End While

Set the NumberOfEntries = counter

Close the data file

ErrorHandler:

Write the message: “There was a problem in loading the data in the data file”

### **WriteFile**

Open the data file for writing data

For all entries in the arrays (*For count = 1 to*

*NumberOfRecords*)

Write a line of data into the data file

End For

Close the data file

### **Navigation Button Code**

The remaining buttons – Open, Save, Exit, Delete, Search and Sort - require code within the actionPerformed method.

The pseudo code for these might be as follows:

### Open

If the file not empty

Select data

separate data using regex

Display data on allocated field

endif

### **Save**

Save the current record

### **Exit**

Write the array data to the data file

Quit the program

### **Delete**

For all records from the current to the last record

Copy the next Name to the current record

Copy the next Birthday to the current record

Copy the next Likes to the current record

End For

Decrement the NumberOfEntries

If the current entry is now past the new last entry

Set the current record back one

End if

Display the current entry

### Search

Set found = false

Set counter = o

While you have more records to check and the entry has  
not been found

If the current entry is equals to the search string

Set found = true

End If

Increment the counter

End While

If Found

Set the current record to the one that was found

Display the current entry

End if

### Sort

get array from write function

create new instance of class SortingForm

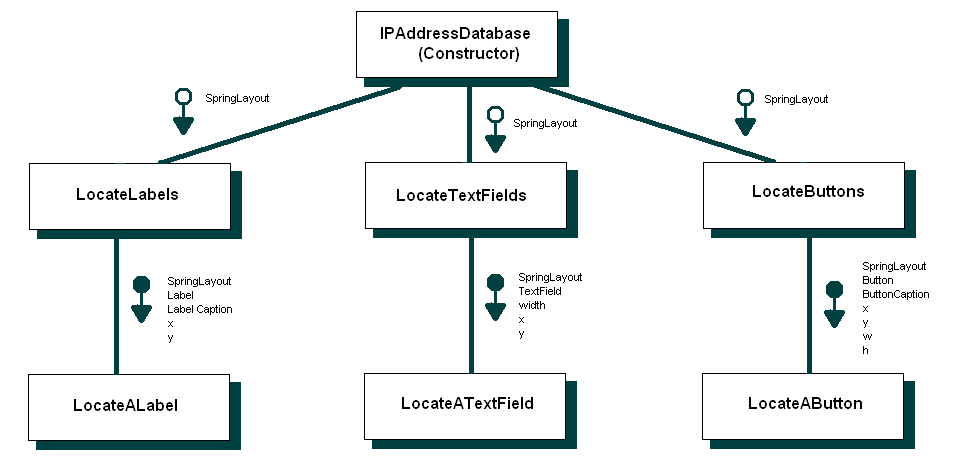
scan through array sorting A-Z (Alphabetically)

create new instance of search window

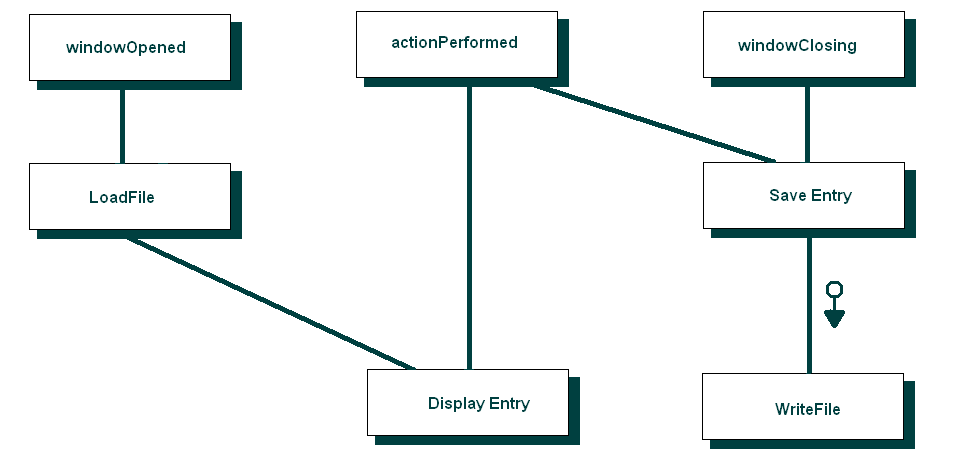
display sorted array in search window

## **Structured Chart(s)**

### **Managing the GUI elements**

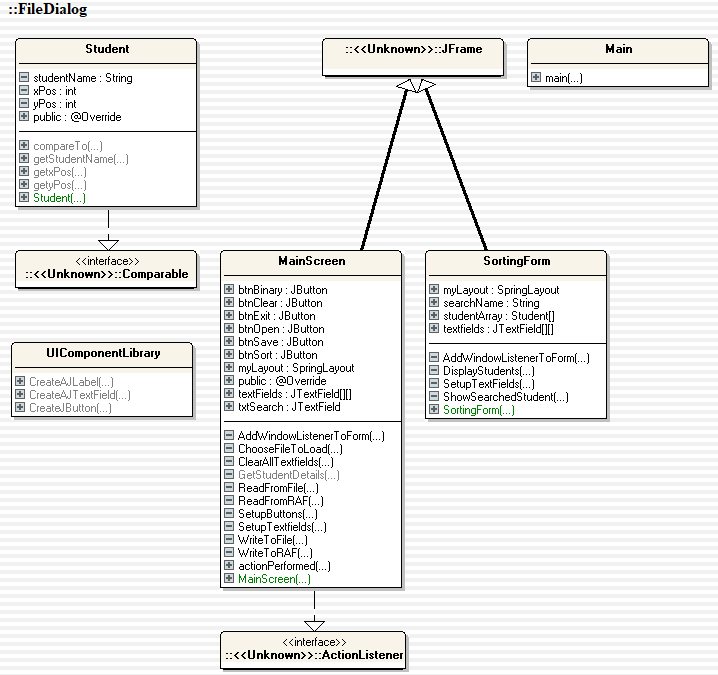


### **Managing the user Actions**



## **Class diagram**

(Using **ESSModel.EXE *(supplied in Third Party Material)***):



## **Testing**

### **Test Plan**

Testing the application will include:

* Assessing requirements (Open, Delete, Name Search, Exit, Sort,Clear) and ensuring that functionality has been addressed
* Create a test design such as a flowchart and relationship diagram to ensure that all features work effectively and to navigate through each step of testing
* Develop the application to specifications including but not limited to the features required
* Once developed, create breakpoints and watches in the application methods to show the output of each data storage point including variables and arrays that are used. this will ensure that all data is correctly being passed to the application.
* after data integrity is confirmed each method/ feature will be tested through a series of dummy data to make sure that data is stored, transferred and displayed correctly
* After it is confirmed that proper data can be used, faulty data will be tested such as data with whitespace or the wrong data types to see how the application reacts and handles the data. appropriate action will be taken such as trimming in the methods.
* the application will then be compiled and deployed on different systems to ensure compatibility with required systems
* reports will be taken based on information received about usage with live data and will be added to a main report to fix potential issues.
* once completed all of the information, testing and logs will be archived and stored for use when necessary

*Regarding the* ***Test Plan*** *below:*

* *Provide detail against each of the Test Plan headings. The goal is to indicate what and how you would be planning to complete your program testing.*

### **Test Plan:**

Test Objectives *(What is your testing aiming to achieve?)*

The objective of the testing is to achieve an application that can Display Classroom layouts with the functionality of Open, Delete, Name Search, Exit, Sort and clear buttons. The information will be read from a RAF file and will highlight and exclude desks from binary search. For the objective to be successful it is required that all functions perform properly without errors. This means that a comprehensive test strategy is required in order to complete the objective of having a stable and operating application.

Test Strategy *(How do you aim to complete your testing?)*

The test strategy breaks the application testing into methods and functions before testing the entire application's ability to manipulate information and display it to the user interface. This is done by creating watches and breakpoints in the code to see how the data is transferred and how the functions operate. then once tested and debugged. The entire application shall be tested with many sets of data, some complete and some incompatible or incomplete for the program. This will test how the program handles the errors and in turn will allow for fixes to handle unexpected information input. This includes trim commands and parsing to integers or strings.

Resource Requirements *(What personnel, computers, software, etc, will you need to complete your testing?)*

For a project of this scale only one developer, preferably the developer who created the application will be required to debug and test the application. they in turn will need a computer, IntelliJ and access to the debugging tools that come with IntelliJ, this may include a licence to the full version of IntelliJ for full debugging tools.

Roles and Responsibilities *(Who will be involved and what will be their respective roles?)*

The Project manager will be involved overseeing the project and talking to the client on behalf of the company. Their role is to meet with the development team and discuss issues and progress throughout the testing phase. The other person who will be required is the developer himself. the developer will be incharge of testing and debugging the application and reporting this to a test log. The responsibility of the developer is also to accurately report the findings and bugs to the project manager, this will be discussed in the next question.

Bug Reporting *(How will you be providing the results of your testing to your manager and/or the development team?)*

Bugs will be reported in the test logs as log dumps from the IDE which in this case is intelliJ along with screenshots of the breakpoints and watches that have been added. The bugs will then have a section for potential solutions where a developer can input potential solutions even if they have not yet been implemented.

*Regarding the* ***Test Cases*** *below:*

* *Note that the headings are a guide only.*
* *In the first section you might copy and paste the content of your current sample data file.*
* *In the second section you could re-copy your data file content, but edit it such that it would cause errors or exceptions.*
* *In the next sections you could copy screen images of your program maybe showing sample data entry.*
* *If you were, for example, to click on the NEW button multiple times, this would create blank entries in your data. People using your program might consider this to be an exception (not preferred).*
* *Note: I am not requiring you to fix such issues, but maybe just comment on them.*
* *You can add a summary of your testing at the end of the testing section.*

### **Test Cases:**

The read file system is created through a buffered reader that saves the data from a RAF file transverted into a csv then to an array, the way that it is tested is by creating dummy data in the RAF file and then through the debugger targeting the output of the array.

the output of the string String[] breaker = line.split(",");. the output of the string is shown above, this shows through the debugger that the array is storing the data on the first line and shows that it correctly stores each value into the correct field.

the complete working code is shown below;

private void ReadFromFile(String filePath)

{

try

{

BufferedReader br = new BufferedReader(new FileReader(filePath));

String line;

while((line = br.readLine()) != null)

{

String[] temp = line.split(",");

int yPos = temp[0];

int xPOs = temp[1];

textFields[yPos][xPOs].setText(temp[2]);

if (temp.length >= 4)

{

if (temp[3].equals("CYAN"))

{

textFields[yPos][xPOs].setBackground(Color.*CYAN*);

}

}

}

br.close();

}

catch (Exception e)

{

System.*out*.println(e);

}

}

an error that was occurred initaly was;

java: incompatible types: java.lang.String cannot be converted to int

This is as in the array each variable needs to parsed into its corresponding value, this was fixed as seen Below by parsing the string into an integer value using Integer.parseInt();. this allowed for all of the data to be correctly stored into the array. Fixed Code:

private void ReadFromFile(String filePath)

{

try

{

BufferedReader br = new BufferedReader(new FileReader(filePath));

String line;

while((line = br.readLine()) != null)

{

String[] temp = line.split(",");

int yPos = Integer.*parseInt*(temp[0]);

int xPOs = Integer.*parseInt*(temp[1]);

textFields[yPos][xPOs].setText(temp[2]);

if (temp.length >= 4)

{

if (temp[3].equals("CYAN"))

{

textFields[yPos][xPOs].setBackground(Color.*CYAN*);

}

}

}

br.close();

}

catch (Exception e)

{

System.*out*.println(e);

}

}

The write function was created using the code below;

private void WriteToRAF()

{

FileDialog fd = new FileDialog(this,"Save File",FileDialog.*SAVE*);

fd.setDirectory("C://");

fd.setVisible(true);

if (fd.getFile() == null)

{

return;

}

String filePath = fd.getDirectory() + fd.getFile();

if (filePath.endsWith(".RAF") == false)

{

filePath+= ".RAF";

}

try

{

RandomAccessFile raf = new RandomAccessFile(filePath,"rw");

int count = 0;

for (int y = 0; y < textFields.length; y++)

{

for (int x = 0; x < textFields[y].length; x++)

{

if (textFields[y][x].getText().isEmpty() == false)

{

int index = count \* 150; //number indicates length of each set of entries

raf.seek(index);

raf.writeUTF(textFields[y][x].getText());

raf.seek(index + 50);

raf.writeInt(y);

raf.seek(index + 75);

raf.writeInt(x);

if (textFields[y][x].getBackground() == Color.*CYAN*)

{

raf.seek(index + 100);

raf.writeUTF("CYAN");

}

count++;

}

}

}

raf.close();

JOptionPane.*showMessageDialog*(null,"Records Saved Successfully");

}

catch (Exception e)

{

}

}

It can be tested using the same method as the read function though the debugger to see the data that will be written to each value in the text fields according to their index. A possible error that occurred is where the Print writer append the data into the file rather than rewriting, this would cause the file to have duplicate fields. the write method operates in two sections, write to csv or txt and write to RAF file, the write method initialiser is seen here with the new mode added.

Code Write() method

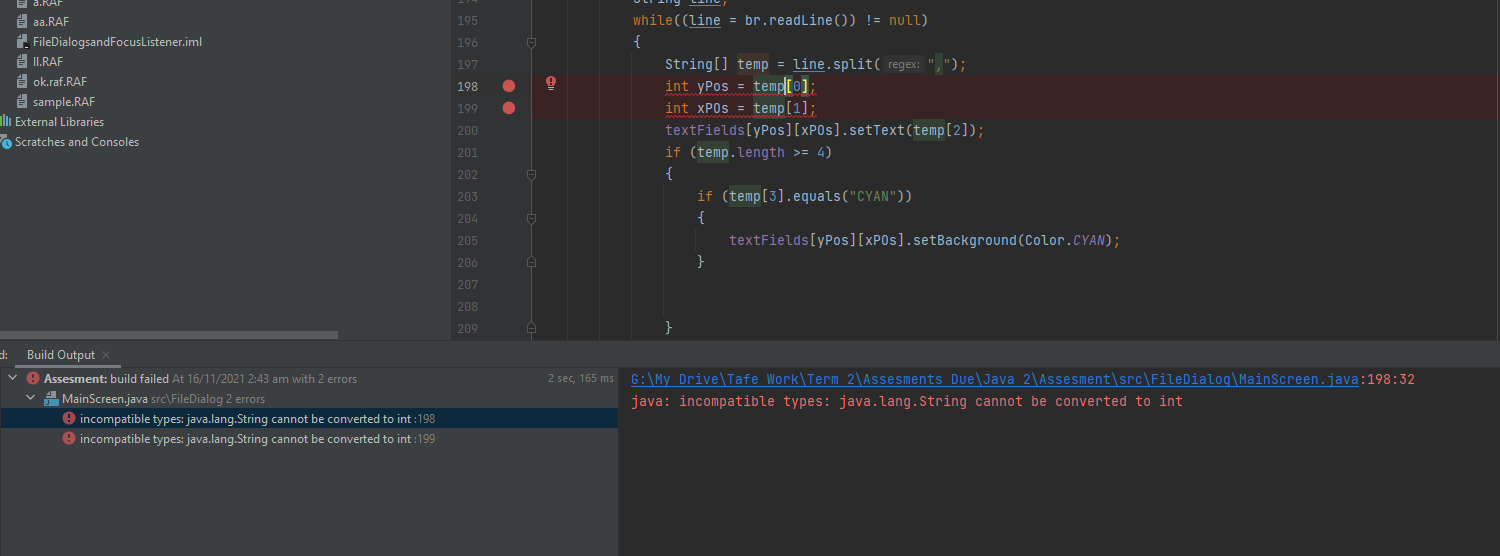
RandomAccessFile raf = new RandomAccessFile(filePath,"rw");

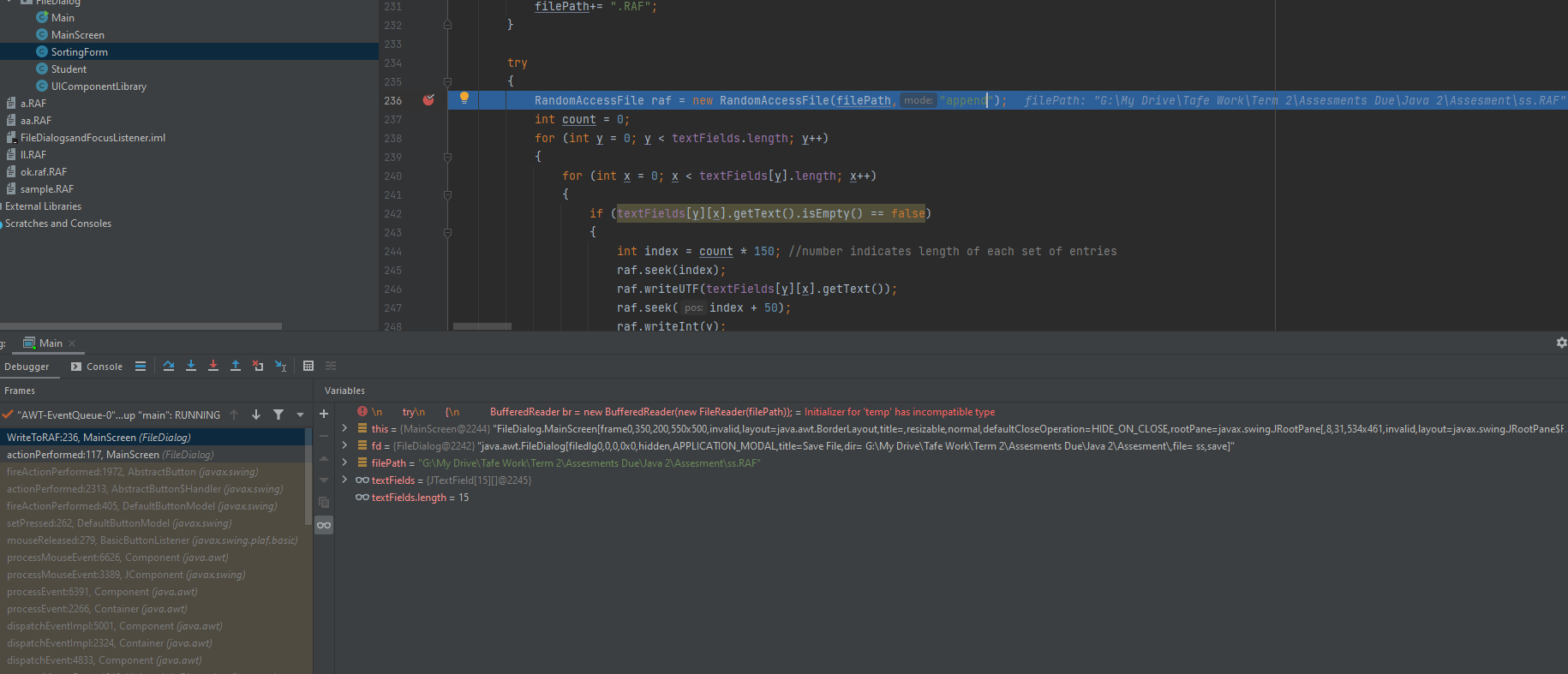
This is a simple fix and involves removing the append value from the RandomAccessFile Writer.

Through the testing errors were found and debugged accordingly as seen above. these will have all been reported to the test log as to allow for transparency for future updates.

## **Evidence of Debugging**

Two to three screen images demonstrating debugging:





## **System generated program documentation (Javadoc)**

Prepare automated program documentation using a facility provided within your IDE,

such as **Javadoc,** or using a third-party product such as ESS Model.

Ensure this automated documentation is provided within your submission.

***Included in submission***

# **User Manual for:** Classroom Organiser

|  | Features by Component/Button  **Screen opens:** Load Birthday-Info.csv and the Birthday tracker window  **New** button: Clear screen ready for new entry  **Save** button: Save the current entry and then all entries to a clean copy of the data file  **Find** button: Use in conjunction with the Search text box to find an existing entry  **Delete** button: Deletes currently selected person from array and CSV file  **Navigation** buttons: navigates through the array to display either the first, last, next or previous entry in the csv file  **Search** button: Takes the input from the search box to find all birthdays in a given month.  **Binary Search** button: When clicked it will display all users names alphabetically  **Exit** button: Exits the application |
| --- | --- |
| How Do I?  **Add and Save a New Entry**   1. Click on the **New Entry** button 2. Type in your new entry 3. Click on the **Save** **Entry** button to save the entry | How Do I?  **Find and Update an Existing Entry**   1. Type the Name 2. Click on the **Find Entry** button 3. Check the entry presented is correct 4. Edit the entry as required 5. Click on the **Save** **Entry** button to save the entry |
| How Do I?  **Delete an entry**   1. Type the Name 2. Click on the **Find Entry** button 3. Check the entry presented is correct 4. Select **Delete** | How Do I?  **Search By month**   1. Enter month number in text field 2. click **Search** 3. view results in box on the left hand side of the search box |
| How Do I?  **Delete an Navigate**   1. Select navigation option, **forward, backward, First, Last** 2. Click button 3. See results on the text fields above | How Do I?  **Exit**   1. Locate **Exit** button 2. Click button |

# **Evidence of correspondence with your Manager**

*Maintain appropriate ongoing communication with your manager.*

*This can be done with a series of applicable emails.*

1. *Email your manager before the commencement of this project to secure the* ***project specification*** *and the* ***organisation’s programming and documentation standards****.*
2. *Prepare a closing email to your manager regarding the* ***success of your project development*** *endeavours. You should suggest any areas of concern or risk to your manager for inclusion in the development team’s risk matrix.*

To: Project Manager / Manager

From:

CC:

BCC:

Date: 19/07/2023

Subject: Request for Program Specifications and Standards

-------------------------------------------------------------

Hi <Project Manager>,

Regarding… the Program Specifications and the Organisation’s Programming and Documentation Standards

The program has been determined that it needs to have the ability to Open, Delete, Name Search, Exit, Sort and clear a display of classroom layouts and class information in order for the application to function properly. this will be implemented through a saved array and a read/write method. The specifications of the project will be followed. Please specify your organisation's programming and documentation standards before development proceeds to ensure that they are in line with other softwares that your company has developed.

If you have any queries please reach out.

Regards,

<Lead Developer>

----------------------------------------------

< Lead Developer >

Lead Developer@BusinessName.com.au

-----------------------------------------------

To: Project Manager / Manager

From:

CC:

BCC:

Date: 10/09/2023

Subject: Project Success and Information

-------------------------------------------------------------

Hi <Project Manager>,

Regarding… the project’s success

The program has been developed successfully and has been tested appropriately, errors and fixes have been logged in the documentation. Please be aware that the software uses a FileWriter, RAFWriter and a PrintWriter that connects to a statically referenced CSV file and changing the file path will cause errors in the program. All functions perform their tasks appropriately.

I am pleased with the outcome and again please reach out with any queries or concerns.

Regards,

<Lead Developer>

----------------------------------------------

< Lead Developer >

Lead Developer@BusinessName.com.au

-----------------------------------------------