Criterion C: Development

Contents

[Complex Data Types 2](#_Toc124110803)

[2D ArrayList 2](#_Toc124110804)

[LinkedHashMap 2](#_Toc124110805)

[ArrayLists 4](#_Toc124110806)

[File Access 5](#_Toc124110807)

[Writing to File 5](#_Toc124110808)

[Writing to Text File 5](#_Toc124110809)

[Writing to Excel File (.xlsx) 7](#_Toc124110810)

[Reading from File 8](#_Toc124110811)

[Reading from Text File 8](#_Toc124110812)

[Reading from PDF 9](#_Toc124110813)

[Error Handling 10](#_Toc124110814)

[Try Catch 10](#_Toc124110815)

[Managing Null Values 10](#_Toc124110816)

[External Libraries 10](#_Toc124110817)

[Javax 10](#_Toc124110818)

[Apache Poi 11](#_Toc124110819)

[PDFBox 11](#_Toc124110820)

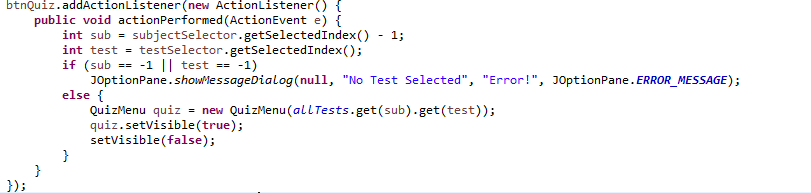
[Encapsulation 11](#_Toc124110821)

[Works Cited 12](#_Toc124110822)

# Complex Data Types

## 2D ArrayList

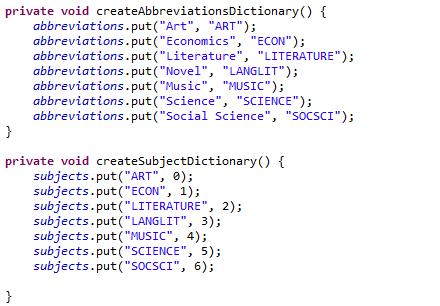


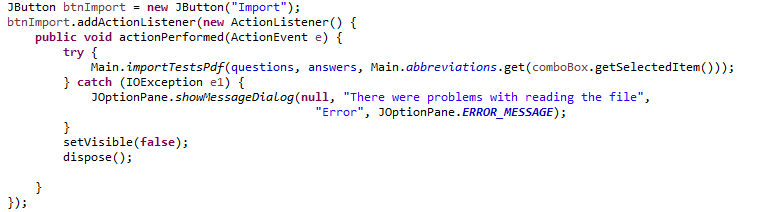


The specified Test object is retrieved using the 2D ArrayList

A two-dimensional ArrayList is utilized in the main class to easily store each test for each subject. This allows the code to easily access the stored tests as it keeps the different subjects split. Any individual test could be found at any time by using the get() method twice with the subject’s index and then the test’s index. Therefore, when the user wants to import data and access the data later, it is much easier to accomplish these goals using a 2D ArrayList. Such examples of the use of this 2D ArrayList use are when the program is accessing a specific test to quiz the user, as shown in the image, or whenever the user imports new pdfs.

## LinkedHashMap



The full subject name is converted to an abbreviated one. For example, Social Science is returned as SOCSCI

The subject names returned as a Set

Text

Description automatically generated

Iterating over those subject names

LinkedHashMaps are used to access and convert from data easily and quickly. LinkedHashMaps are like HashMaps, except that they have the added benefit of retaining order. For example, since each subject directly corresponds to a given index in the 2D ArrayList, whenever the program is given the subject name, the LinkedHashMap *subjects* can quickly convert it to the subject’s index. Another use for the map is to convert from an abbreviated version of the subject name that is found in the PDF’s text. This ability helps with the parsing of the data. Moreover, since the LinkedHashMap has the ability to retain order, the keys or the values can be given as a Collection. This allows the program to be able to iterate over these values, which is helpful when reading from the text file that is saved when the program is closed.

## ArrayLists

Text

Description automatically generated

ArrayLists are used primarily in the Test class to easily add questions to a specific test. Since there is not a set number of questions for each file the user wants to import, the program utilizes the fact that ArrayLists do not have a defined length. Thus, the program is able to handle any valid files that the user specifies. Moreover, ArrayLists also have get() and set() methods to easily be able to access or modify any data.

# File Access

## Writing to File

### Writing to Text File



PrintWriter takes all data and organizes it into a file

The program uses the PrintWriter class in order to write to a text file and save the data. This allows the user to retain their previously imported tests and statistics upon closing the program. Furthermore, it also gives the functionality to allow the user to transfer data to another device without importing every test again.

Text

Description automatically generated A picture containing text

Description automatically generated

Subject Dividers

Statistics Values

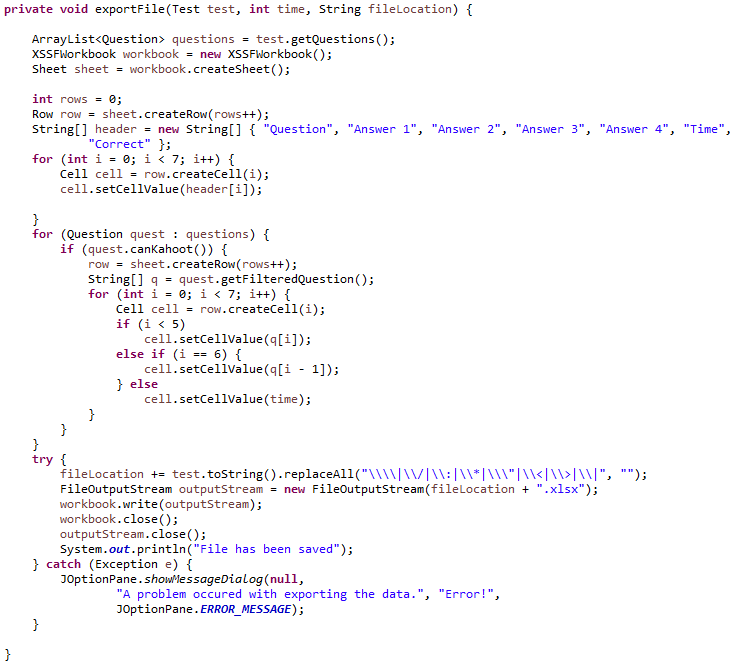
Net Score for each question

Question, Answer Choice, Answer, Explanation

Test Title

The program utilizes a simple text file to store the information as it is able to hold the data in an organized and efficent way. The data is split into different lines representing the specific information pertaining to each subject, test, and question. For example, as shown in the image, ART indicates the subject of the following tests. Then it is followed by a completely capitalized line representing the a test name, which is then followed by lines representing the question.

### Writing to Excel File (.xlsx)

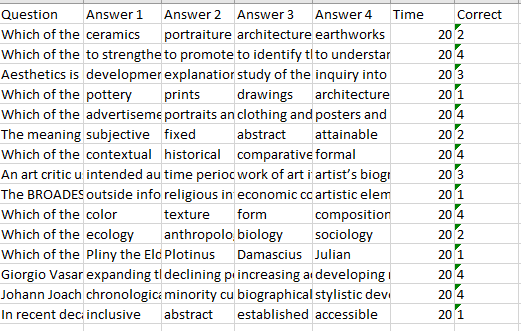


Writes to an excel file

Changes the answer choices from 5 to 4

Ensures each question fits the constraints for Kahoot

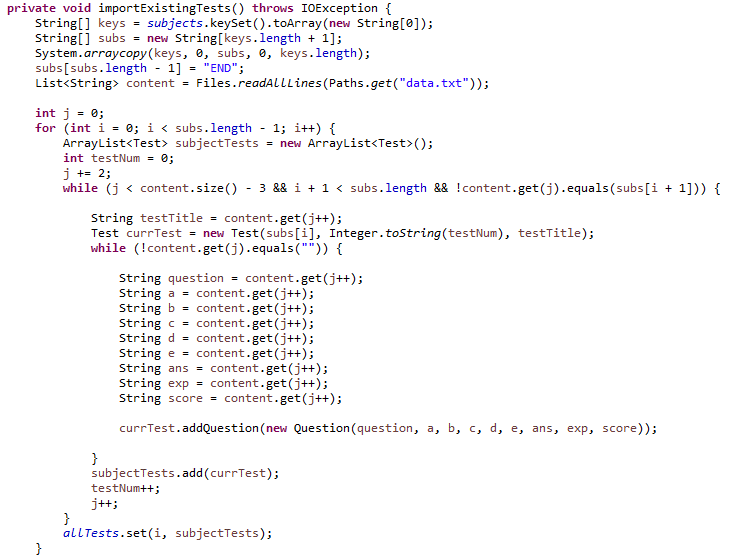
The program also utilized the Apache POI library (which will be further elaborated later) to export to a Microsoft Excel file. It uses XSSFWorkbook, Sheet, Row, and Cell to create a spreadsheet with all of the data from a test and outputs it to an excel file (.xlsx). It allows the easy transfer for data into other external group quizzing programs, such as Kahoot, whenever the client would like to create an activity for an entire class.



The exported file follows a specific format to be compatible with third-party programs. It separates each question into a different row and each part of the question into separate columns.

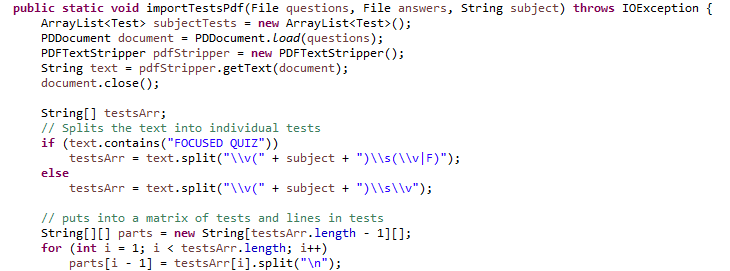
## Reading from File

### Reading from Text File



Using the Files static method to read lines, the program was easily able to separate each line from a text file and store them as elements in a List. This enables the program to read existing files when opened and reload the data into the JComboBoxes and JLabels, improving the usability and functionality of the program.

### Reading from PDF



Loop through all tests in the PDF

Splits the text by each occurrence of a subject using a regular expression.

PDFTextStripper extracts text from a PDF



Parse one test

Importing the text from pdf files is one of the core functionalities of the program. It utilizes PDDocument and PDFTextStripper in conjunction with the String split() method in order to separate the tests and individual lines into an array. This allows for the method to easily parse the data into the format that allows for the quiz portion of the program to function.

# Error Handling

## Try Catch

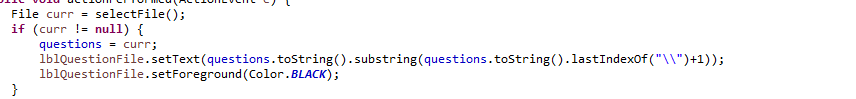
Catches an error with the files that the user has attempted to import.

Graphical user interface, text, application

Description automatically generated with medium confidence

Try/Catch blocks are vital to the functionality of the program and ensure that the user cannot crash the program if they select invalid files or incompatible ones. For example, the user cannot input a questions file from the Art subject and an answer key from the Music subject because the subjects are different.

## Managing Null Values

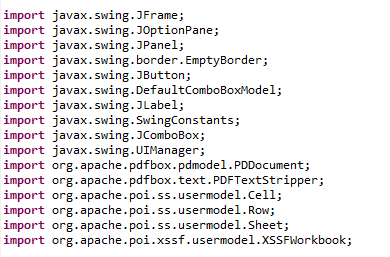


The program also prevents null input situations by ensuring that the user selects both a questions and answers pdf when attempting to import questions. Thus, whenever the user has selected a valid pdf file, the text turns from red to black.

# External Libraries

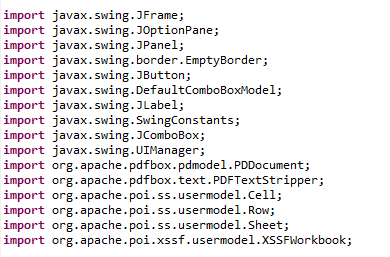
The code implements the following external libraries to employ existing methods and algorithms. These following libraries enhance the program’s functionality, user interface, and overall experience.

## Javax



The Javax library provides many elements to improve the program’s user interface and functionality, such as the inclusion of user interactive buttons and dropdowns while also introducing labels to appropriately communicate information to the user.

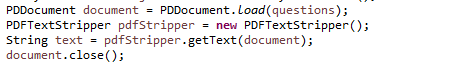
## Apache Poi



Apache POI helps to integrate exporting information to Microsoft documents, which is vital for the export functionality of the program. This is because the program needs to be able to export to a Microsoft Excel format (.xlsx).

## PDFBox





PDFBox allows the program to convert PDF documents to text. Since quizzes are provided in a PDF format by a third-party service, PDFBox allows the program to parse the information.

# Encapsulation

Text

Description automatically generated

Company name

Description automatically generated with medium confidence

A picture containing diagram

Description automatically generated

Text

Description automatically generated with low confidence

The code utilizes encapsulation to keep certain variables private with getter and setter methods to access and modify these variables. It also helped to keep the code more readable, which helps with future development of the program.

# Works Cited

<https://www.codejava.net/coding/how-to-write-excel-files-in-java-using-apache-poi>

<https://www.tutorialspoint.com/how-to-read-data-from-pdf-file-and-display-on-console-in-java>

<https://www.javatpoint.com/pdfbox-environment-setup>

<https://www.toolsqa.com/blogs/download-apache-poi/>

<https://stackoverflow.com/questions/2889777/difference-between-hashmap-linkedhashmap-and-treemap>

<https://stackoverflow.com/questions/13139283/jframe-theme-and-appearance>

<https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html>

<https://stackoverflow.com/questions/20624825/using-jframe-and-jfilechooser-to-select-a-file>

<https://www.geeksforgeeks.org/java-swing-jcombobox-examples/>

<https://www.geeksforgeeks.org/split-string-java-examples/>

<https://regex101.com>

<https://stackoverflow.com/questions/9093448/how-to-capture-a-jframes-close-button-click-event>

Word Count: 1203