Criterion C: Development

Contents

Complex Data Types

2D ArrayList

```
private static ArrayList<ArrayList<Test>> allTests;
```

```
btnQuiz.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int sub = subjectSelector.getSelectedIndex() - 1;
        int test = testSelector.getSelectedIndex();
        if (sub == -1 || test == -1)
            JOptionPane.showMessageDialog(null, "No Test Selected", "Error!", JOptionPane.ERROR_MESSAGE);
        else {
            QuizMenu quiz = new QuizMenu(allTests.get(sub).get(test));
            quiz.setVisible(true);
            setVisible(false);
        }
    }
});
```

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 data 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.

LinkedHashMap

```
public static LinkedHashMap<String, Integer> subjects;
public static LinkedHashMap<String, String> abbreviations;
```

```
private void createAbbreviationsDictionary() {
    abbreviations.put("Art", "ART");
    abbreviations.put("Economics", "ECON");
    abbreviations.put("Literature", "LITERATURE");
    abbreviations.put("Novel", "LANGLIT");
    abbreviations.put("Music", "MUSIC");
    abbreviations.put("Science", "SCIENCE");
    abbreviations.put("Social Science", "SOCSCI");
}

private void createSubjectDictionary() {
    subjects.put("ART", 0);
    subjects.put("ECON", 1);
    subjects.put("LITERATURE", 2);
    subjects.put("LANGLIT", 3);
    subjects.put("MUSIC", 4);
    subjects.put("SCIENCE", 5);
    subjects.put("SOCSCI", 6);

}
```

```
JButton btnImport = new JButton("Import");
btnImport.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            Main.importTestsPdf(questions, answers, Main.abbreviations.get(comboBox.getSelectedItem()));
        } catch (IOException e1) {
            JOptionPane.showMessageDialog(null, "There were problems with reading the file",
                                          "Error", JOptionPane.ERROR_MESSAGE);
        }
        setVisible(false);
        dispose();

    }
});
```

LinkedHashMaps are used to easily and quickly access and convert from associated data. Since each subject directly corresponds to a given index in the 2D ArrayList, LinkedHashMap was specifically chosen for this program because of the need to maintain order of the keys.

ArrayLists

```
private ArrayList<Question> questions;
```

```
    /**
     * Instantiates a new test.
     *
     * @param subject the subject
     * @param testId the test id
     * @param testName the test name
     */
    public Test(String subject, String testId, String testName) {
        questions = new ArrayList<Question>();
        myTestID = testId;
        myTestName = testName.replaceAll("\\|\\/|\\:|\\*|\\?|\\<|\\>|\\||\"", "");
        mySubject = subject;
    }

    /**
     * Instantiates a new test.
     *
     * @param subject the subject
     * @param testId the test id
     * @param testName the test name
     * @param quests the questions
     */
    public Test(String subject, String testId, String testName, ArrayList<Question> quests) {
        questions = new ArrayList<Question>();
        myTestID = testId;
        myTestName = testName.replaceAll("\\|\\/|\\:|\\*|\\?|\\<|\\>|\\||\"", "");
        questions = quests;
        mySubject = subject;
    }
```

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

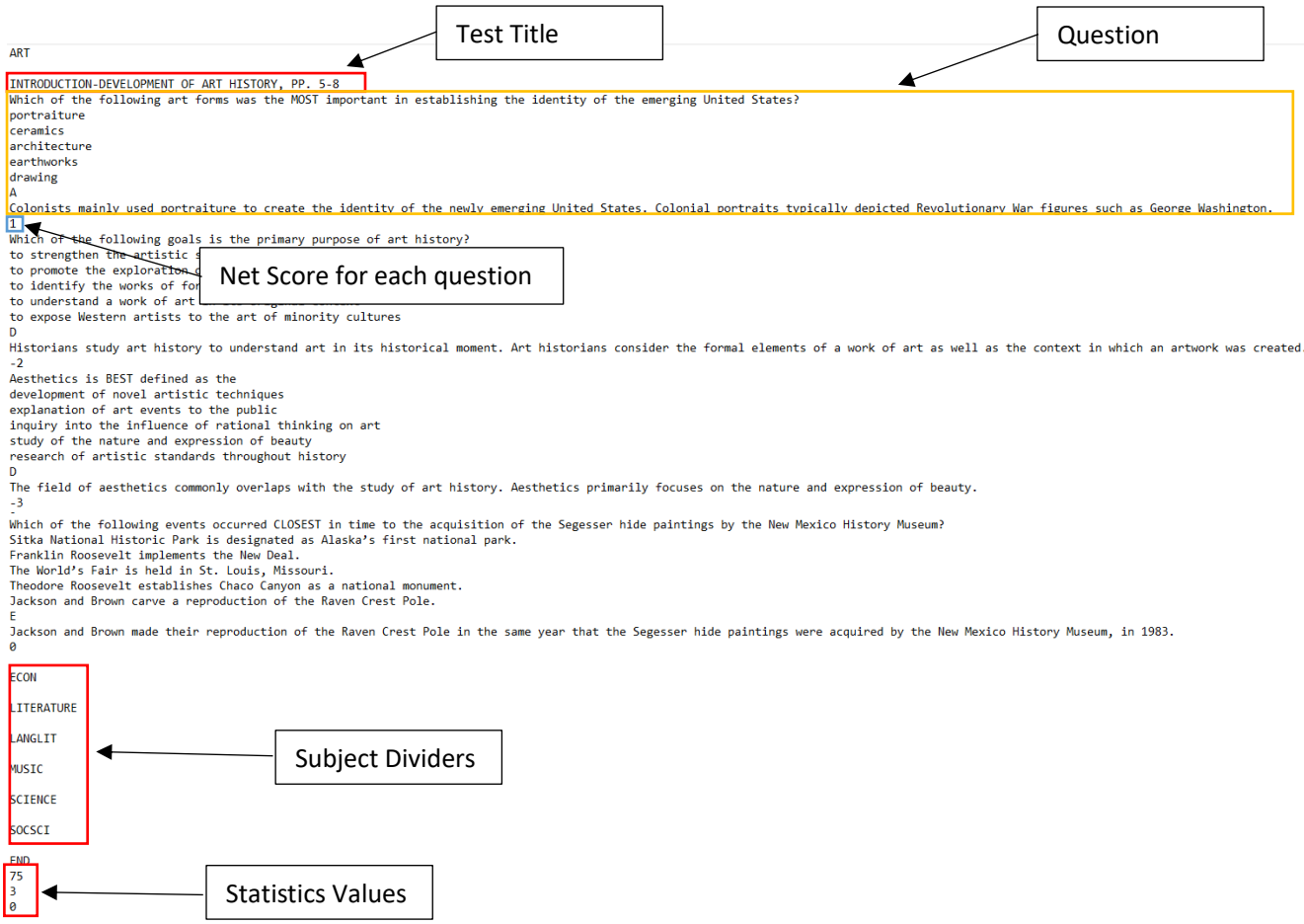
Writing to File

Write to Text File

```
private void writeToFile() throws IOException {
    PrintWriter pw = new PrintWriter("data.txt");
    String[] arr = subjects.keySet().toArray(new String[0]);
    for (int i = 0; i < arr.length; i++) {
        pw.write(arr[i] + "\n");

        ArrayList<Test> currSubject = allTests.get(i);

        for (Test test : currSubject) {
            ArrayList<Question> quests = test.getQuestions();
            pw.write("\n" + test.getTestName() + "\n");

            for (Question q : quests) {
                String[] strs = q.getQuestion();
                for (String str : strs) {
                    pw.write(str + "\n");
                }
                pw.write(q.getScore() + "\n");
            }
        }
        pw.write('\n');
    }
    pw.write("END\n");
    pw.write(Integer.toString((int) timer) + '\n');
    pw.write(Integer.toString(testsTaken) + '\n');
    pw.write(Integer.toString((int) totalScore) + '\n');
    System.out.println("Wrote to File!");
    pw.close();

}
```

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.



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.

Writing to Excel File (.xlsx)

```
private void exportFile(Test test, int time, String fileLocation) {

    ArrayList<Question> questions = test.getQuestions();
    XSSFWorkbook workbook = new XSSFWorkbook();
    Sheet sheet = workbook.createSheet();

    int rows = 0;
    Row row = sheet.createRow(rows++);
    String[] header = new String[] { "Question", "Answer 1", "Answer 2", "Answer 3", "Answer 4", "Time",
            "Correct" };
    for (int i = 0; i < 7; i++) {
        Cell cell = row.createCell(i);
        cell.setCellValue(header[i]);

    }
    for (Question quest : questions) {
        if (quest.canKahoot()) {
            row = sheet.createRow(rows++);
            String[] q = quest.getFilteredQuestion();
            for (int i = 0; i < 7; i++) {
                Cell cell = row.createCell(i);
                if (i < 5)
                    cell.setCellValue(q[i]);
                else if (i == 6) {
                    cell.setCellValue(q[i - 1]);
                } else
                    cell.setCellValue(time);
            }
        }
    }
    try {
        fileLocation += test.toString().replaceAll("\\\\|\\/|\\:|\\*|\\\"|\\<|\\>|\\|", "");
        FileOutputStream outputStream = new FileOutputStream(fileLocation + ".xlsx");
        workbook.write(outputStream);
        workbook.close();
        outputStream.close();
        System.out.println("File has been saved");
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,
                "A problem occured with exporting the data.", "Error!",
                JOptionPane.ERROR_MESSAGE);
    }

}
```

Ensures each question fits the constraints for Kahoot

Changes the answer choices from 5 to 4

Writes to an excel file

The program also utilized the Apache POI library to export to a Microsoft Excel file. It uses XSSFWorkbook, Sheet, Row, and Cell in order 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 quizzing programs.

Question	Answer 1	Answer 2	Answer 3	Answer 4	Time	Correct
Which of the	ceramics	portraiture	architecture	earthworks	20	2
Which of the	to strengthe	to promote	to identify tl	to understar	20	4
Aesthetics is	developmer	explanatior	study of the	inquiry into	20	3
Which of the	pottery	prints	drawings	architecture	20	1
Which of the	advertiseme	portraits an	clothing and	posters and	20	4
The meaning	subjective	fixed	abstract	attainable	20	2
Which of the	contextual	historical	comparative	formal	20	4
An art critic u	intended au	time perioc	work of art i	artist's biogr	20	3
The BROADES	outside info	religious in	economic cc	artistic elem	20	1
Which of the	color	texture	form	composition	20	4
Which of the	ecology	anthropolo	biology	sociology	20	2
Which of the	Pliny the Elc	Plotinus	Damascius	Julian	20	1
Giorgio Vasar	expanding tl	declining p	increasing a	developing	20	4
Johann Joach	chronologica	minority cu	biographical	stylistic dev	20	4
In recent dec	inclusive	abstract	established	accessible	20	1

The file that is exported 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 a separate columns.

Reading from File

Text File

```
private void importExistingTests() throws IOException {
    String[] keys = subjects.keySet().toArray(new String[0]);
    String[] subs = new String[keys.length + 1];
    System.arraycopy(keys, 0, subs, 0, keys.length);
    subs[subs.length - 1] = "END";
    List<String> content = Files.readAllLines(Paths.get("data.txt"));

    int j = 0;
    for (int i = 0; i < subs.length - 1; i++) {
        ArrayList<Test> subjectTests = new ArrayList<Test>();
        int testNum = 0;
        j += 2;
        while (j < content.size() - 3 && i + 1 < subs.length && !content.get(j).equals(subs[i + 1])) {

            String testTitle = content.get(j++);
            Test currTest = new Test(subs[i], Integer.toString(testNum), testTitle);
            while (!content.get(j).equals("")) {

                String question = content.get(j++);
                String a = content.get(j++);
                String b = content.get(j++);
                String c = content.get(j++);
                String d = content.get(j++);
                String e = content.get(j++);
                String ans = content.get(j++);
                String exp = content.get(j++);
                String score = content.get(j++);

                currTest.addQuestion(new Question(question, a, b, c, d, e, ans, exp, score));

            }
            subjectTests.add(currTest);
            testNum++;
            j++;
        }
        allTests.set(i, subjectTests);
    }
```

Using the Files static method to read all line, the code was easily able to separate each line of a text file into elements in a List. This allows the program to be able to read existing file when it is opened and reloads the data into the JComboBoxes and JLabels, which improves the quality of life of the program.

PDF

```
public static void importTestsPdf(File questions, File answers, String subject) throws IOException {
    ArrayList<Test> subjectTests = new ArrayList<Test>();
    PDDocument document = PDDocument.load(questions);
    PDFTextStripper pdfStripper = new PDFTextStripper();
    String text = pdfStripper.getText(document);
    document.close();

    String[] testsArr;
    // Splits the text into individual tests
    if (text.contains("FOCUSED QUIZ"))
        testsArr = text.split("\\v(" + subject + ")\\s(\\v|F)");
    else
        testsArr = text.split("\\v(" + subject + ")\\s\\v");

    // puts into a matrix of tests and lines in tests
    String[][] parts = new String[testsArr.length - 1][];
    for (int i = 1; i < testsArr.length; i++)
        parts[i - 1] = testsArr[i].split("\n");

for (int i = 0; i < parts.length; i++) {
    int questionNum = 0;
    String[] currTestArr = parts[i];

    int j = 0;

    if (currTestArr[j].equals(" "))
        j++;

    while (currTestArr[j].length() != 2)
        j++;
    j++;

    while (j < currTestArr.length) {
        String currAnswer = "";

        do {
            if (currTestArr[j].matches("\s\s"))
                j++;
            else if (currTestArr[j].contains("DEMIDEC"))
                j += 2;
            else
                currAnswer += currTestArr[j++].strip() + " ";
        } while (j < currTestArr.length && !currTestArr[j].matches("\s\s") && (currTestArr[j].length() <= 5
                || !currTestArr[j].substring(0, 5).matches("[1-9]\\.\s[A-Z]\s|\\d\\d\\.\\s[A-Z]")));

        if (currAnswer.length() != 0) {
            int index = currAnswer.indexOf('.');
            String ans = currAnswer.substring(index + 2, index + 3);
            int end = currAnswer.indexOf('[');
            String ansExp = currAnswer.substring(index + 4);
            if (end != -1)
                ansExp = currAnswer.substring(index + 4, end);
            subjectTests.get(i).getSpecificQuestion(questionNum).setAnswer(ans, ansExp);
            questionNum++;

        }
    }
}
```

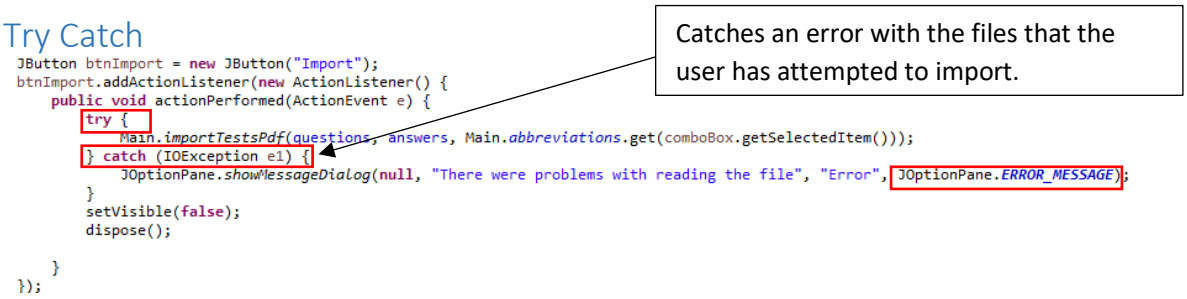
Loop through all tests in the PDF

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 rest of the program to function.

Error Handling

Try Catch



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 in the Art subject and an answer key from the Music subject, because they do not match.

Managing Null Values

```
File curr = selectFile();
if (curr != null) {
    questions = curr;
    lblQuestionFile.setText(questions.toString().substring(questions.toString().lastIndexOf("\\")+1));
    lblQuestionFile.setForeground(Color.BLACK);
}
```

The program also ensures that the user has selected 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 employ existing methods and algorithms. These following libraries enhance the program's functionality, user interface, and experience.

Javax

```
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JComboBox;
import javax.swing.UIManager;
```

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 allow the program to convey information to the user.

Apache Poi

```
import org.apache.pdfbox.text.PDFTextStripper;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
```

Apache POI helps to more easily integrate exporting information to Microsoft Excel (.xlsx) formats.

PDFBox

```
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;
```

```
PDDocument document = PDDocument.load(questions);
PDFTextStripper pdfStripper = new PDFTextStripper();
String text = pdfStripper.getText(document);
document.close();
```

PDFBox allows the program to convert PDF documents to text, which then allows the program to parse the information.

Encapsulation

```
public class Question {

    private String myQuestion;
    private String myA;
    private String myB;
    private String myC;
    private String myD;
    private String myE;

    /** The answer to the question. */
    private String myAnswer;
    /** The explanation why the given answer is correct. */
    private String myAnswerExplanation;
    /** The net amount that the user has gotten this question right/wrong. */
    private int myScore;

public String[] getQuestion() {
    return new String[] { myQuestion, myA, myB, myC, myD, myE, myAnswer, myAnswerExplanation };
}

public int getScore()
{
    return myScore;
}

public void setAnswer(String ans, String ansExp) {
    myAnswer = ans;
    myAnswerExplanation = ansExp;
}
```

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 as well as making the classes easier to use only being able to access or modify certain variables.