Appendix B – Source Code

```java
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import javax.swing.JButton;

import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.awt.event.ActionEvent;
import javax.swing.JComboBox;
import javax.swing.JFileChooser;
import javax.swing.UIManager;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.Reader;
import java.nio.file.Files;
import java.nio.file.Paths;

import org.apache.pdfbox.pdmodel.PDDocument;
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;
import javax.swing.DefaultComboBoxModel;
import java.awt.event.*;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import java.awt.Font;

public class Main extends JFrame {

 public static LinkedHashMap<String, Integer> subjects;
 public static LinkedHashMap<String, String> abbreviations;
 public static Main window;

 private static ArrayList<ArrayList<Test>> allTests;
 private JPanel contentPane;
 private double timer = 0;
 private int testsTaken = 0;
 private double totalScore = 0;
 private JLabel lblTotalTime;
 private JLabel lblAverageScore;
 private JLabel lblAverageTimePerTest;
 private JLabel lblTotalTests;


 public static void main(String[] args) {
  EventQueue.invokeLater(new Runnable() {
   public void run() {
    try {
     Main frame = new Main();
     frame.setVisible(true);

    } catch (Exception e) {
     e.printStackTrace();
    }
   }

  });

 }

 /**
  * Create the frame.
  */
 @SuppressWarnings("rawtypes")
 public Main() {
  WindowListener listener = new WindowAdapter() {

   @Override
   public void windowClosing(WindowEvent we) {
    try {
     writeToFile();
     setVisible(false);
     System.exit(0);
    } catch (IOException e) {
     e.printStackTrace();
    }
   }
  };
  addWindowListener(listener);

  window = this;
  //Sets the UI Style to the system style
  try {
   UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
  } catch (Exception e) {
   e.printStackTrace();
  }

  // initalizes the allTests arraylist
  allTests = new ArrayList<ArrayList<Test>>();
  for (int i = 0; i < 7; i++) {
   allTests.add(new ArrayList<Test>());
```

```java
}

//intializes and populates the dictionaries
subjects = new LinkedHashMap<String, Integer>();
createSubjectDictionary();
abbreviations = new LinkedHashMap<String, String>();
createAbbreviationsDictionary();

setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
setBounds(100, 100, 852, 480);
contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);
contentPane.setLayout(null);

JComboBox testSelector = new JComboBox();
testSelector.setEnabled(false);
testSelector.setBounds(10, 44, 424, 22);
contentPane.add(testSelector);

JComboBox subjectSelector = new JComboBox(getSubjects());
subjectSelector.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  String a = (String) subjectSelector.getSelectedItem();

  if (!a.equals("Choose Subject")) {
   int num = subjects.get(abbreviations.get(a));
   testSelector.setModel(setTestSelectorOptions(num));

   testSelector.setEnabled(true);
  } else {
   testSelector.setEnabled(false);
   testSelector.setModel(new DefaultComboBoxModel());
  }
 }
});

subjectSelector.setBounds(10, 11, 424, 22);
contentPane.add(subjectSelector);

JButton btnQuiz = new JButton("Quiz");
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);
  }
 }
});
btnQuiz.setBounds(10, 131, 189, 29);
contentPane.add(btnQuiz);

JButton btnReview = new JButton("Review Missed Questions");
btnReview.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 {
   Test review = allTests.get(sub).get(test).getReviewTest();
   if (review.getQuestions().size() == 0) {
    JOptionPane.showMessageDialog(null, "Test has nothing to review :D", "Error!",
      JOptionPane.ERROR_MESSAGE);
   } else {
    QuizMenu quiz = new QuizMenu(allTests.get(sub).get(test).getReviewTest());
    quiz.setVisible(true);
    setVisible(false);
   }
  }
 }
});

btnReview.setBounds(10, 171, 189, 29);
contentPane.add(btnReview);

JButton btnExit = new JButton("Exit");
btnExit.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  try {
   writeToFile();
   setVisible(false);
   System.exit(0);
  } catch (IOException e1) {
  }
 }
});
btnExit.setBounds(10, 294, 189, 29);
contentPane.add(btnExit);

JButton btnImport = new JButton("Import");
btnImport.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  ImportMenu menu = new ImportMenu();
  menu.setVisible(true);
 }
});
btnImport.setBounds(10, 212, 189, 29);
contentPane.add(btnImport);

JButton btnExport = new JButton("Export");
btnExport.addActionListener(new ActionListener() {
 btnExport.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
   int sub = subjectSelector.getSelectedIndex() - 1;
   int test = testSelector.getSelectedIndex();
   if (sub == -1)
    JOptionPane.showMessageDialog(null, "No Test Selected", "Error!", JOptionPane.ERROR_MESSAGE);
   else
   {
    int value = JOptionPane.showOptionDialog(null, "Would you like to export the currently selected test or all tests in this subject", "Export Optio
           JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, new String[] {"Only Selected Test", "All Tests"}, null);
    if(value == -1)
     return;
    else
    {
     String fileLocation = selectDir();
     if(fileLocation == null)
      return;
     if(value == JOptionPane.YES_OPTION)
      exportFile(allTests.get(sub).get(test), 20, fileLocation);
     else
     {
      for (int i = 0; i < testSelector.getItemCount(); i++) {
        exportFile(allTests.get(sub).get(i), 20, fileLocation);
        System.out.println("Done with test: " + i);
       }
      }
     }
    }
   }

 });

 btnExport.setBounds(10, 253, 189, 29);
 contentPane.add(btnExport);

 JLabel lblStats = new JLabel("Statistics");
 lblStats.setFont(new Font("Calibri", Font.BOLD, 30));
 lblStats.setVerticalAlignment(SwingConstants.TOP);
 lblStats.setHorizontalAlignment(SwingConstants.CENTER);
 lblStats.setBounds(452, 44, 372, 44);
 contentPane.add(lblStats);

 lblTotalTime = new JLabel("Total Time: ");
 lblTotalTime.setFont(new Font("Calibri", Font.BOLD, 12));
 lblTotalTime.setBounds(462, 137, 362, 22);
 contentPane.add(lblTotalTime);

 lblAverageScore = new JLabel("Average Score:");
 lblAverageScore.setFont(new Font("Calibri", Font.BOLD, 12));
 lblAverageScore.setBounds(462, 171, 362, 22);
 contentPane.add(lblAverageScore);

 lblAverageTimePerTest = new JLabel("Average Time per Test:");
 lblAverageTimePerTest.setFont(new Font("Calibri", Font.BOLD, 12));
 lblAverageTimePerTest.setBounds(462, 205, 362, 22);
 contentPane.add(lblAverageTimePerTest);

 lblTotalTests = new JLabel("Tests Taken:");
 lblTotalTests.setFont(new Font("Calibri", Font.BOLD, 12));
 lblTotalTests.setBounds(462, 240, 362, 22);
 contentPane.add(lblTotalTests);

 JButton btnResetStats = new JButton("Reset Statistics");
 btnResetStats.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
   int result = JOptionPane.showConfirmDialog(null, "Are you sure you want to reset all statistics?", "", 0);
   if (result == JOptionPane.OK_OPTION) {
   timer = 0;
   testsTaken= 0;
   totalScore = 0;
   updateLabels();
   }
  }
 });
 btnResetStats.setBounds(499, 403, 294, 26);
 contentPane.add(btnResetStats);

 // see if there was previously saved data in current directory
 if (new File("data.txt").exists()) {
  //attempts to import the data
  try {
   importExistingTests();
   System.out.println("Tests Imported");
  // If errors, it just continue on
  } catch (IOException e) {
   e.printStackTrace();
   JOptionPane.showMessageDialog(null,
     "A problem occured with loading the data. Continuing without importing data.", "Error!",
     JOptionPane.ERROR_MESSAGE);
  }

 }

 updateLabels();

}

/**
 * Gets the subjects.
 *
 * @return all subjects
 */
private String[] getSubjects() {
 String[] selections = new String[8];
 selections[0] = "Choose Subject";

 String[] objs = abbreviations.keySet().toArray(new String[0]);

```

```java
  for (int i = 0; i < objs.length; i++) {
   selections[i + 1] = objs[i];
  }

  return selections;
 }

 /**
  * Import tests from Pdf files
  *
  * @param questions the location of the pdf that contains the questions
  * @param answers   the location of the pdf that contains the answers
  * @param subject   the subject the tests should be added to
  * @throws IOException Signals that an I/O exception has occurred.
  */
 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");

  // gets data for the questions
  for (int i = 0; i < parts.length; i++) {
   String[] currTestArr = parts[i];
   int j = 0;
   if (currTestArr[j].equals(" ") || currTestArr[j].equals(""))
    j++;
   String id = currTestArr[j].substring(currTestArr[j].length() - 4, currTestArr[j].length() - 2);
   String testTitle = "";
   j++;

   while (currTestArr[j].charAt(0) != ' ')
    testTitle += currTestArr[j++].strip() + " ";
   Test currTest = new Test(subject, id, testTitle);
   j++;

   while (j < currTestArr.length) {
    String question = "";
    String a = "";
    String b = "";
    String c = "";
    String d = "";
    String e = "";
    // Get Question
    while (!currTestArr[j].substring(0, 2).equals("a."))
     question += currTestArr[j++].strip() + " ";

    // Get Choice A
    while (!currTestArr[j].substring(0, 2).equals("b."))
     a += currTestArr[j++].strip() + " ";
    // Get Choice B
    while (!currTestArr[j].substring(0, 2).equals("c."))
     b += currTestArr[j++].strip() + " ";
    // Get Choice C
    while (!currTestArr[j].substring(0, 2).equals("d."))
     c += currTestArr[j++].strip() + " ";
    // Get Choice D
    while (!currTestArr[j].substring(0, 2).equals("e."))
     d += currTestArr[j++].strip() + " ";
    // Get Choice E
    // REGEX: digit 1-9 + period OR 2 digits
    while (j < currTestArr.length && !currTestArr[j].substring(0, 2).matches("[1-9]\\.|\\d\\d")) {
     if (currTestArr[j].contains("DEMIDEC"))
      j += 2;
     else
      e += currTestArr[j++].strip() + " ";
    }
    Question q = new Question(question.strip(), a, b, c, d, e);
    currTest.addQuestion(q);
   }

   subjectTests.add(currTest);

  }
  document = PDDocument.load(answers);
  text = pdfStripper.getText(document);
  document.close();


  // Splits the text into tests
  testsArr = text.split("\\v(" + subject + ")\\s(\\v|F)");
  // puts into a matrix of tests and lines in tests
  parts = new String[testsArr.length - 1][];

  for (int i = 1; i < testsArr.length; i++)
   parts[i - 1] = testsArr[i].split("\n");

  // gets data from answers
  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++;

    }
   }
  }
  System.out.println("Imported Tests");
  int num = subjects.get(subject);
  allTests.set(num, subjectTests);
 }

 /**
  * Import existing tests from data.txt.
  *
  * @throws IOException Signals that an I/O exception has occurred.
  */
 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);
  }
  j++;
  timer = Integer.parseInt(content.get(j++));
  testsTaken = Integer.parseInt(content.get(j++));
  totalScore = Integer.parseInt(content.get(j++));

 }

 /**
  * Write to file.
  *
  * @throws IOException Signals that an I/O exception has occurred.
  */
 private void writeToFile() throws IOException {
  PrintWriter pw = new PrintWriter("data.txt", "UTF-8");
  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();

 }

 /**
  * Sets the test selector options.
  *
  * @param sub the subject that is selected
  * @return the DefaultComboBoxModel listing all the tests for JComboBox
  */
 private DefaultComboBoxModel<String> setTestSelectorOptions(int sub) {
  ArrayList<Test> curr = allTests.get(sub);
  DefaultComboBoxModel<String> model = new DefaultComboBoxModel<String>();

  for (Test test : curr) {
   model.addElement(test.toString());
  }
  return model;

 }

 /**
  * Exports the file to an xlsv fommat
  *
  * @param test Test that is wanted to be exported
  * @param time The time that should be allocated to every question
  */
 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 {
   // CLEAN FILES WITH \/:*?<>|
   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) {
   e.printStackTrace();
  }

 }


 /**
  * Selects the directory the file is saved to
  *
  * @return the string containing the directory
  */
 private String selectDir() {

  JFileChooser fc = new JFileChooser();
  fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
  int output = fc.showOpenDialog(this);
  if (output == JFileChooser.APPROVE_OPTION) {
   return fc.getSelectedFile().toString() + "\\";
  }
  else
   JOptionPane.showMessageDialog(null, "Export Cancelled");
  return null;
 }


 private void createAbbreviationsDictionary() {
  abbreviations.put("Art", "ART");
  abbreviations.put("Economics", "ECON");
  abbreviations.put("Literature", "LITERATURE");
  abbreviations.put("Novel", "LANGLIT");
  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);

 }

 /**
  * Gets the abbreviations dictionary.
  *
  * @return the abbreviations dictionary
  */
 public LinkedHashMap<String, String> getAbbreviations() {
  return abbreviations;
 }

 /**
  * Update the statistic private variables
  *
  * @param time    increments the timer by this time (Given in MM.SS)
  * @param percent increments the percent by this parameter
  */
 public void updateStats(double time, double percent) {

  timer += time;
  testsTaken++;
  totalScore += percent;
  updateLabels();

 }

 /**
  * Update labels.
  */
 private void updateLabels() {
  lblTotalTime.setText("Total Time: " + (int) timer / 60 + " minutes " + (int) timer % 60 + " seconds");
  if (testsTaken == 0) {
   lblAverageTimePerTest.setText("Average Time per Test: No Tests Taken");
   lblAverageScore.setText("Average Score: No Tests Taken");
  } else {
   System.out.println(totalScore);
   lblAverageScore.setText("Average Score: " + ((int) (totalScore * 100 / testsTaken)) + "%");
   double avgTime = timer / testsTaken;
   lblAverageTimePerTest.setText(
     "Average Time per Test: " + (int) avgTime / 60 + " minutes " + (int) avgTime % 60 + " seconds");
  }
  lblTotalTests.setText("Tests Taken: " + testsTaken);

 }
}
```

```java
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.awt.event.ActionEvent;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.SwingConstants;

@SuppressWarnings("serial")
public class QuizMenu extends JFrame {

 private JPanel contentPane;
 private Test currTest;
 private int questionNum = 0;
 private Question quest;
 private JLabel[] labels;
 private JLabel lblExp;
 private JButton[] buttons;
 private JButton btnNext;
 private int score;
 private long time = System.nanoTime();

 public QuizMenu(Test test) {

  currTest = test;
  quest = currTest.getSpecificQuestion(questionNum);
  setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
  setBounds(100, 100, 852, 480);
  contentPane = new JPanel();
  contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

  setContentPane(contentPane);
  contentPane.setLayout(null);

  WindowListener listener = new WindowAdapter() {

   @Override
   public void windowClosing(WindowEvent we) {

    int result = JOptionPane.showConfirmDialog(null, "Are you sure you want to exit this test?", "", 0);
    if (result == JOptionPane.OK_OPTION) {
     setVisible(false);
     Main.window.setVisible(true);
     dispose();
    }
   }
  };
  addWindowListener(listener);

  lblExp = new JLabel("");
  lblExp.setVerticalAlignment(SwingConstants.TOP);
  lblExp.setHorizontalAlignment(SwingConstants.LEFT);
  lblExp.setBounds(35, 357, 570, 73);
  contentPane.add(lblExp);

  JLabel lblQuestion = new JLabel("Question");
  lblQuestion.setHorizontalAlignment(SwingConstants.LEFT);
  lblQuestion.setBounds(37, 11, 688, 53);
  contentPane.add(lblQuestion);

  btnNext = new JButton("Next");
  btnNext.setVisible(false);
  btnNext.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {

    questionNum++;
    if (currTest.hasNext(questionNum)) {
     quest = currTest.getSpecificQuestion(questionNum);
     promptQuestion();
     btnNext.setVisible(false);
    } else {
     double currTime = (System.nanoTime() - time) / Math.pow(10, 9);
     JOptionPane.showMessageDialog(null,
       "Test Completed!\n" + score + " correct out of " + currTest.length() + "\nTime Taken: "
         + (int) currTime / 60 + " minutes " + (int) currTime % 60 + " seconds",
       "WOOOOOOO", JOptionPane.PLAIN_MESSAGE);

     Main.window.updateStats(currTime, (double) score / currTest.length());
     setVisible(false);
     Main.window.setVisible(true);
     dispose();
    }
   }
  });
  btnNext.setBounds(699, 398, 106, 32);
  contentPane.add(btnNext);

  JLabel lblA = new JLabel("A");
  lblA.setBounds(135, 75, 351, 45);
  contentPane.add(lblA);

  JLabel lblB = new JLabel("B");
  lblB.setBounds(135, 130, 351, 45);
  contentPane.add(lblB);

  JLabel lblC = new JLabel("C");
  lblC.setBounds(135, 185, 368, 45);
  contentPane.add(lblC);

  JLabel lblD = new JLabel("D");
  lblD.setBounds(135, 240, 368, 45);
  contentPane.add(lblD);

  JLabel lblE = new JLabel("E");
  lblE.setBounds(135, 295, 368, 45);
  contentPane.add(lblE);

  labels = new JLabel[] { lblQuestion, lblA, lblB, lblC, lblD, lblE, lblExp };


  JButton btnA = new JButton("A");
  btnA.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    processGuess('A');
   }
  });
  btnA.setBounds(35, 75, 90, 45);
  contentPane.add(btnA);

  JButton btnB = new JButton("B");
  btnB.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    processGuess('B');
   }
  });
  btnB.setBounds(35, 130, 90, 45);
  contentPane.add(btnB);

  JButton btnC = new JButton("C");
  btnC.setForeground(new Color(0, 0, 0));
  btnC.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    processGuess('C');
   }
  });
  btnC.setBounds(35, 185, 89, 45);
  contentPane.add(btnC);

  JButton btnD = new JButton("D");
  btnD.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    processGuess('D');
   }
  });
  btnD.setBounds(35, 240, 89, 45);
  contentPane.add(btnD);
```

```java
 JButton btnE = new JButton("E");
 btnE.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
   processGuess('E');
  }
 });
 btnE.setBounds(35, 295, 90, 45);
 contentPane.add(btnE);

 buttons = new JButton[] { btnA, btnB, btnC, btnD, btnE };
 promptQuestion();

}

/**
 * Disable Answer buttons.
 */
private void disableButtons() {
 for (JButton button : buttons)
  button.setEnabled(false);
}

/**
 * Enable Answer buttons.
 */
private void enableButtons() {
 for (JButton button : buttons)
  button.setEnabled(true);
}

/**
 * Prompt the next question and sets all the labels to correct values.
 */
private void promptQuestion() {
 reset();
 String[] data = quest.getQuestion();

 for (int i = 0; i < data.length - 2; i++) {
  labels[i].setText("<html>" + data[i] + "</html>");
 }

}


/**
 * Sets the correct button the green, sets wrong to red.
 *
 * @param btn the button to set to green
 */
private void setColors(JButton btn) {

 for (JButton l : buttons) {
  if (l != btn)
   l.setBackground(Color.RED);
  else
   l.setBackground(Color.GREEN);
 }
}

/**
 * Processes the guess.
 *
 * @param letter the letter corresponding the the button the user has pressed
 */
private void processGuess(char letter) {
 if (quest.correct("" + letter)) {
  score++;
  quest.changeScore(true);
  setColors(buttons[letter - 'A']);
 } else {
  setColors(buttons[quest.getQuestion()[6].charAt(0) - 'A']);
  quest.changeScore(false);
 }
 disableButtons();
 lblExp.setText("<html>" + quest.getQuestion()[7] + "</html>");
 btnNext.setVisible(true);
 }

 /**
  * Resets the button colors to grey.
  */
 private void reset() {
  for (JButton b : buttons)
   b.setBackground(Color.GRAY);
  labels[6].setText("");

  enableButtons();
 }
}
```

```java
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.JComboBox;
import javax.swing.JFileChooser;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.util.LinkedHashMap;
import java.awt.event.ActionEvent;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Color;

public class ImportMenu extends JFrame{

 private JPanel contentPane;
 private File questions;
 private File answers;

 @SuppressWarnings("unchecked")
 public ImportMenu() {
  setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  setBounds(100, 100, 450, 300);
  contentPane = new JPanel();
  contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

  setContentPane(contentPane);
  contentPane.setLayout(null);

  JLabel lblAnswerFile = new JLabel("No File Chosen");
  lblAnswerFile.setForeground(Color.RED);
  lblAnswerFile.setBounds(101, 168, 224, 14);
  contentPane.add(lblAnswerFile);

  JLabel lblQuestionFile = new JLabel("No File Chosen");
  lblQuestionFile.setForeground(Color.RED);
  lblQuestionFile.setBounds(101, 111, 224, 14);
  contentPane.add(lblQuestionFile);

  JComboBox comboBox = new JComboBox();
  comboBox.setModel(new DefaultComboBoxModel(
    new String[] { "Art", "Economics", "Literature", "Novel", "Music", "Science", "Social Science" }));
  comboBox.setBounds(271, 38, 153, 27);
  contentPane.add(comboBox);

  JButton btnQuestions = new JButton("Choose File");
  btnQuestions.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    File curr = selectFile();
    if (curr != null) {
     questions = curr;
     lblQuestionFile.setText(questions.toString().substring(questions.toString().lastIndexOf("\\")+1));
     lblQuestionFile.setForeground(Color.BLACK);
    }

   }
  });
  btnQuestions.setBounds(325, 105, 99, 27);
  contentPane.add(btnQuestions);

  JButton btnAnswers = new JButton("Choose File");
  btnAnswers.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    File curr = selectFile();
    if (curr != null) {
     answers = curr;
     lblAnswerFile.setText(answers.toString().substring(answers.toString().lastIndexOf("\\")+1));
     lblAnswerFile.setForeground(Color.BLACK);
    }
   }
  });
  btnAnswers.setBounds(325, 162, 99, 27);
  contentPane.add(btnAnswers);

  JLabel lblQuestions = new JLabel("Questions PDF:");
  lblQuestions.setBounds(10, 105, 89, 27);
  contentPane.add(lblQuestions);

  JLabel lblAnswersPdf = new JLabel("Answers PDF:");
  lblAnswersPdf.setBounds(10, 162, 81, 27);
  contentPane.add(lblAnswersPdf);

  JLabel lblSubjects = new JLabel("Subjects");
  lblSubjects.setBounds(10, 38, 196, 27);
  contentPane.add(lblSubjects);

  JButton btnImport = new JButton("Import");
  btnImport.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    if(answers == null || answers == null)
     JOptionPane.showMessageDialog(null, "No File Selected!", "Error", JOptionPane.ERROR_MESSAGE);
    else if(!answers.toString().contains((String) comboBox.getSelectedItem()) || !questions.toString().contains((String)comboBox.getSelectedItem()))
     JOptionPane.showMessageDialog(null, "Please check the selected files to ensure you are importing the correct subject!", "Error", JOptionPane.ERR
    else
    {
     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();
    }
   }
  });
  btnImport.setBounds(104, 227, 89, 23);
  contentPane.add(btnImport);

  JButton btnCancel = new JButton("Cancel");
  btnCancel.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    setVisible(false);
    dispose();
   }
  });
  btnCancel.setBounds(226, 227, 89, 23);
  contentPane.add(btnCancel);

 }

 /**
  * Select file through JFileChooser
  *
  * @return the file
  */
 private File selectFile() {
  JFileChooser jc = new JFileChooser();
  FileNameExtensionFilter filter = new FileNameExtensionFilter(
          "Pdfs", "pdf");
  jc.setFileFilter(filter);
  jc.showOpenDialog(this);
  File file = jc.getSelectedFile();
  return file;

 }



}
```

```java
/**
 * The Question Class.
 */
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;


 /**
  * Instantiates a new question.
  *
  * @param Question the question
  * @param A Choice A
  * @param B Choice B
  * @param C Choice C
  * @param D Choice D
  * @param E Choice E
  * @param ans Answer
  * @param ansExp Answer Explanation
  * @param score the score
  */
 public Question(String Question, String A, String B, String C, String D, String E, String ans, String ansExp,
   String score) {
  myQuestion = Question;
  myA = A;
  myB = B;
  myC = C;
  myD = D;
  myE = E;
  myAnswer = ans;
  myAnswerExplanation = ansExp;
  myScore = Integer.parseInt(score);
 }

 /**
  * Instantiates a new question.
  *
  * @param Question the question
  * @param A Choice A
  * @param B Choice B
  * @param C Choice C
  * @param D Choice D
  * @param E Choice E
  */
 public Question(String Question, String A, String B, String C, String D, String E) {
  myQuestion = Question;
  myA = A;
  myB = B;
  myC = C;
  myD = D;
  myE = E;
  cleanStrings();
 }

 /**
  * Gets the question.
  *
  * @return the question
  */
 public String[] getQuestion() {
  return new String[] { myQuestion, myA, myB, myC, myD, myE, myAnswer, myAnswerExplanation };
 }

 /**
  * Gets the question that only has 4 responses in a random order.
  * Used to export to other programs such as Kahoot that only accept 4 answer questions.
  *
  * @return the filtered question
  */
 public String[] getFilteredQuestion() {
  String[] arr = new String[6];
  String[] quest = getQuestion();

  arr[0] = quest[0];

  int rand = (int) (Math.random() * 4) + 1;

  arr[rand] = answer();
  arr[5] = "" + rand;
  int i = 1;
  int j = 1;
  while (i <= 4) {
   if (i != rand && !quest[j].equals(answer())) {
    arr[i] = quest[j];
    i++;
    j++;
   } else if (i == rand)
    i++;
   else if (quest[j].equals(answer()))
    j++;
  }
  return arr;
 }
```

```java
/**
 * Gets the Answer.
 *
 * @return the answer
 */
private String answer() {
 if (myAnswer.equals("A"))
  return myA;
 else if (myAnswer.equals("B"))
  return myB;
 else if (myAnswer.equals("C"))
  return myC;
 else if (myAnswer.equals("D"))
  return myD;
 else
  return myE;
}

/**
 * Checks if the given String matches Correct answer.
 *
 * @param guesss The guess
 * @return true, if correct
 */
public boolean correct(String guess) {
 if (guess.equals(myAnswer)) {
  return true;
 }
 return false;
}

/**
 * Sets the answer.
 *
 * @param ans the ans
 * @param ansExp the ans exp
 */
public void setAnswer(String ans, String ansExp) {
 myAnswer = ans;
 myAnswerExplanation = ansExp;
}

public String toString() {
 return myQuestion + "\nA: " + myA + "\nB: " + myB + "\nC: " + myC + "\nD: " + myD + "\nE: " + myE + "\nAnswer: "
   + myAnswer + " " + myAnswerExplanation;
}

/**
 * Runs the cleanString method on every text field
 */
private void cleanStrings()
{
 myQuestion = cleanString(myQuestion);
 myA = cleanString(myA);
 myB = cleanString(myB);
 myC = cleanString(myC);
 myD = cleanString(myD);
 myE = cleanString(myE);
}

/**
 * Cleans the string.
 *
 * @param str String to be cleaned
 * @return cleaned string
 */
private String cleanString(String str)
{
  int i = str.indexOf(".");
  str= str.substring(i+2);

 return str;

}

/**
 * Can export to kahoot.
 *
 * @return true, if successful
 */
public boolean canKahoot()
{
 return myQuestion.length() <= 120 && myA.length() <= 75 && myB.length() <= 75 && myC.length() <= 75 && myD.length() <= 75 && myE.length() <= 75;
}

/**
 * Change score.
 *
 * @param correct If the user has gotten the question right. True if Right, False if wrong
 */
public void changeScore(boolean correct)
{
 if(correct)
  myScore++;
 else
  myScore--;
}

/**
 * Gets the score.
 *
 * @return the score
 */
public int getScore()
{
 return myScore;
```

```
    return myScore;
  }


}

```

```java
import java.util.ArrayList;

public class Test {

 /** The questions. */
 private ArrayList<Question> questions;

 /** The test ID. */
 private String myTest;

 /** The test name. */
 private String myTestName;

 /** The subject. */
 private String mySubject;

 /**
  * 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>();
  myTest = 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>();
  myTest = testId;
  myTestName = testName.replaceAll("\\|\\/|\\:|\\*|\\?|\\<|\\>|\\||\"", "");
  questions = quests;
  mySubject = subject;
 }

 /**
  * Gets the test name.
  *
  * @return the test name
  */
 public String getTestName() {
  return myTestName;
 }

 /**
  * Gets the test ID.
  *
  * @return the test ID
  */
 public String getTestID() {
  return myTest;
 }

 /**
  * Gets the subject.
  *
  * @return the subject
  */
public String getSubject() {
 return mySubject;
}

/**
  * Sets the subject.
  *
  * @param subject the new subject
  */
public void setSubject(String subject) {
 mySubject = subject;
}



/**
  * Gets the questions.
  *
  * @return the questions
  */
public ArrayList<Question> getQuestions(){
 return questions;
}

/**
  * Gets the specific question.
  *
  * @param index the index
  * @return the specific question
  */
public Question getSpecificQuestion(int index){
 return questions.get(index);
}

/**
  * Adds the question.
  *
  * @param ques the ques
  */
public void addQuestion(Question ques) {
 questions.add(ques);
}


/**
  * To string.
  *
  * @return the string
  */
public String toString()
{
 return myTestName;
}

/**
  * Checks if there is a next question.
  *
  * @param index the index
  * @return true, if successful
  */
public boolean hasNext(int index)
{
 return index < questions.size();
}
```

```java

  /**
   * Length.
   *
   * @return the int of the questions ArrayList
   */
  public int length()
  {
   return questions.size();
  }


  /**
   * Gets the review test.
   *
   * @return a test that only contains questions with a score < 0
   */
  public Test getReviewTest()
  {
   ArrayList<Question> reviewQuests = new ArrayList<Question>();
   for(Question q: questions)
   {
    if(q.getScore() < 0)
     reviewQuests.add(q);
   }

   return new Test(mySubject, myTest, myTestName, reviewQuests);
  }

}
```