COS80007 Advanced Java - Assignment One

Formal Written Report

Joshua Skinner *101 601 828*

Bradley Chick *101 626 151*

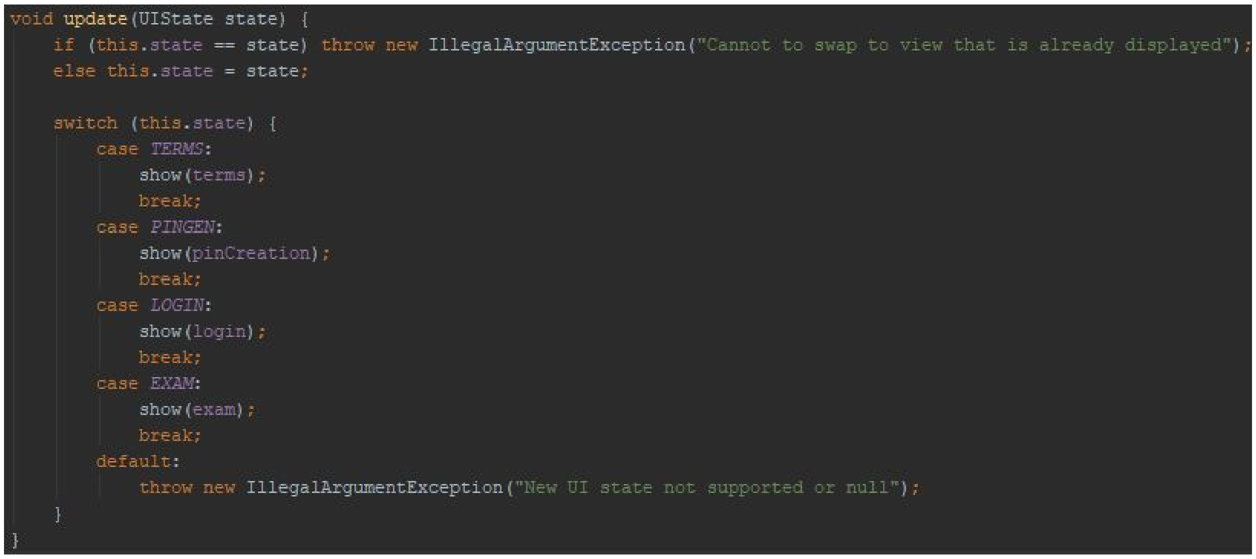
Introduction

This report will go over the running behaviour of our group's assignment implementation. Clearly numbered, sequential screenshots of the program will be included alongside explanations and justification of design choices. That being said, this is one of the many final results produced by students in the unit this semester. Our group, being a couple of undergrads, are pleased with our result, even if it is sub-par for what we might have expected.

Loading Up The Program

When the program is first started, a child class of JFrame called "MainView" takes first precedence and is responsible for the instantiation of everything. Hence, why it is called "MainView". MainView does not create and display anything on its own, but rather, displays instances of other Views which are a child class of JPanel.

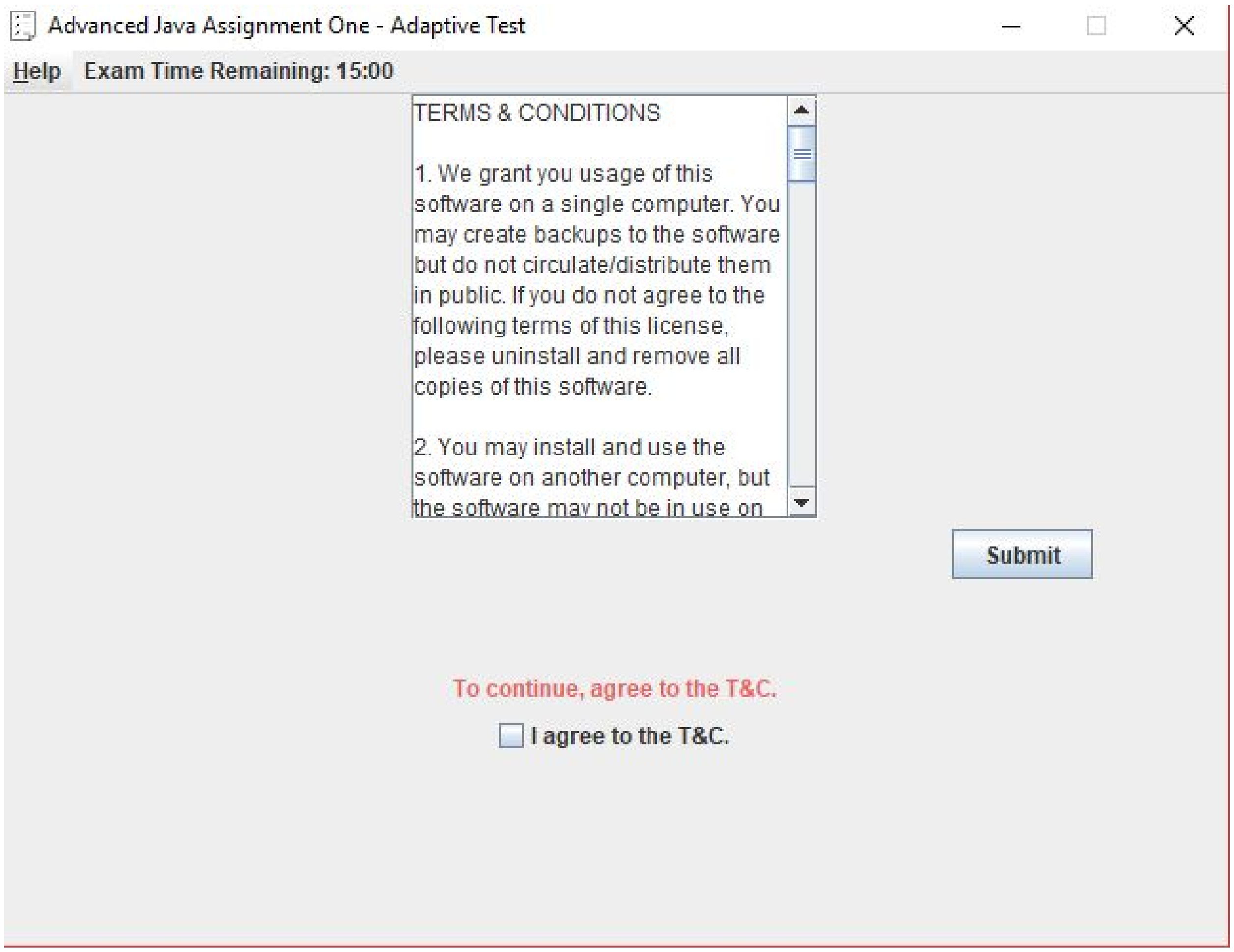
Screenshot 1.1



```
void update(UIState state) {
    if (this.state == state) throw new IllegalArgumentException("Cannot to swap to view that is already displayed");
    else this.state = state;

    switch (this.state) {
        case TERMS:
            show(terms);
            break;
        case PINGEN:
            show(pinCreation);
            break;
        case LOGIN:
            show(login);
            break;
        case EXAM:
            show(exam);
            break;
        default:
            throw new IllegalArgumentException("New UI state not supported or null");
    }
}
```

As shown in Screenshot 1.1, MainView contains a method that takes a "UI State" enumerated type and shows the appropriate View accordingly. Every other View has access to the instance of MainView, but do not have access to instances of other views. This is so Views can pass over their precedence on screen to another view, or access the models to collect information. In the constructor of MainView, all view instances are initialised, a help menu dialog system is created (lots and lots of strings!) and the screen precedence is passed over to the UI state "TERMS". As shown in Screenshot 1.1, this makes the TermsView visible on screen.

Screenshot 1.2


Advanced Java Assignment One - Adaptive Test

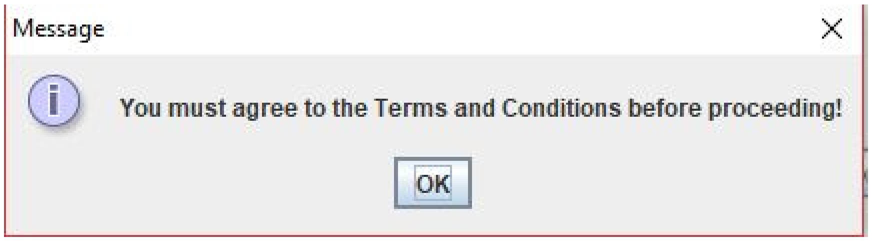
TERMS & CONDITIONS

1. We grant you usage of this software on a single computer. You may create backups to the software but do not circulate/distribute them in public. If you do not agree to the following terms of this license, please uninstall and remove all copies of this software.

2. You may install and use the software on another computer, but the software may not be in use on

To continue, agree to the T&C.

Screenshot 1.3

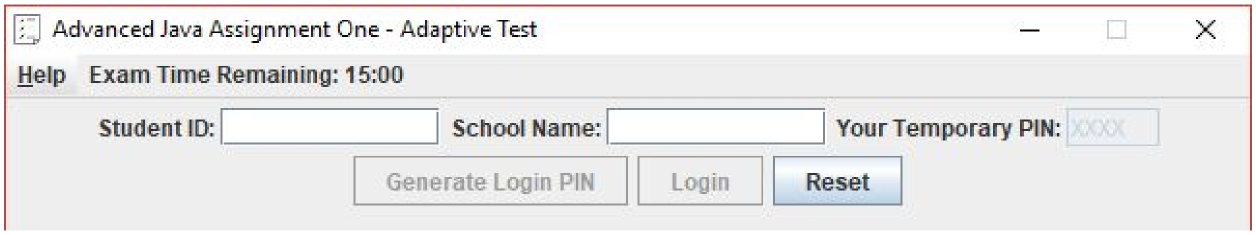

Message

You must agree to the Terms and Conditions before proceeding!

Screenshot 1.2 is what appears when the program is first loaded up. This is the Terms and Conditions View. It should also be noted that there is a “Help” menu item, as well as the Exam Timer, which does not start counting down until later. These however, are part of the MainView (JFrame) and are always visible no matter what View is being displayed. The Terms and Conditions View is relatively simple, as checking “Agree” and submitting it will tell MainView to pass precedence onto the Pin Creation View. Screenshot 1.3 depicts the dialog that is displayed if the user does not agree to the terms and conditions. They cannot use the software if they do not agree to it.

Lastly, the Terms and Conditions View uses a GridLayout in a 2x3 formation to display its components.

Screenshot 1.4

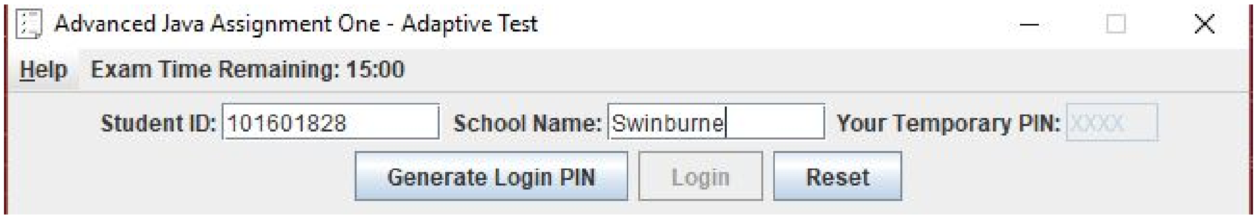


Advanced Java Assignment One - Adaptive Test

Help Exam Time Remaining: 15:00

Student ID: School Name: Your Temporary PIN: XXXX

Screenshot 1.5

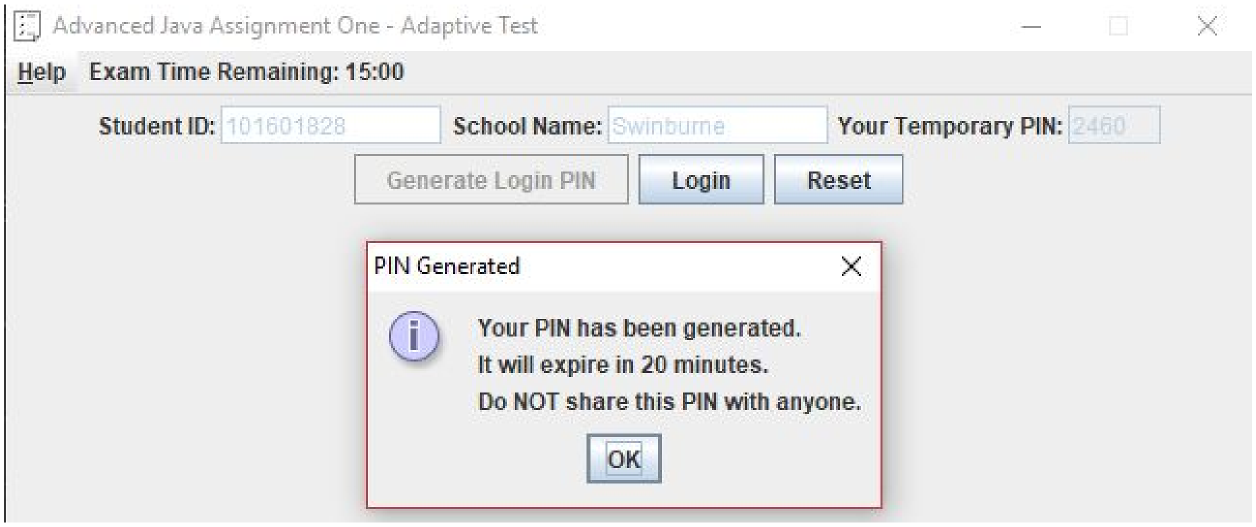


Advanced Java Assignment One - Adaptive Test

Help Exam Time Remaining: 15:00

Student ID: 101601828 School Name: Swinburne Your Temporary PIN: XXXX

Screenshot 1.6



Advanced Java Assignment One - Adaptive Test

Help Exam Time Remaining: 15:00

Student ID: 101601828 School Name: Swinburne Your Temporary PIN: 2460

PIN Generated

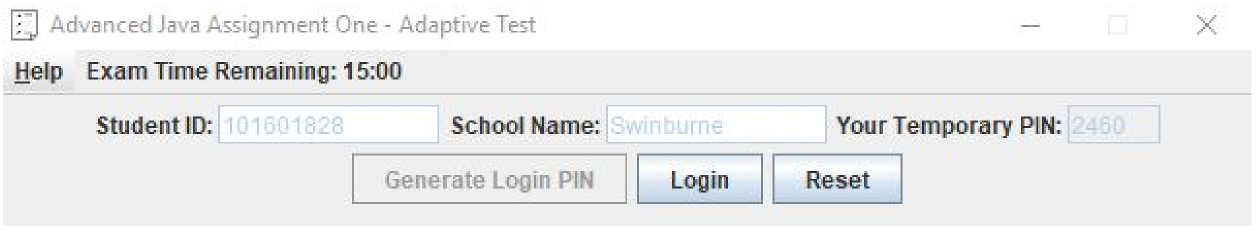
Your PIN has been generated.
It will expire in 20 minutes.
Do NOT share this PIN with anyone.

If the user agrees to the terms and conditions and submits as a symbol of their approval, as told above, Terms and Conditions View will tell MainView to give Pin Creation view the precedence on screen. Screenshot 1.4 is what the Pin Creation View looks like. While lacking any fancy structuring, functionality is completely there. To advance through the software further, as per software specification, the user must enter their Student ID, as well as their School Name. It should be noted that the “Generate Login PIN” button stays disabled until the user enters a Student ID and School Name that are more than 5 characters long. It becomes enabled once these criterions are satisfied, as shown in screenshot 1.5.

Pressing “Generate Login PIN” does a few things (as shown in screenshot 1.6):

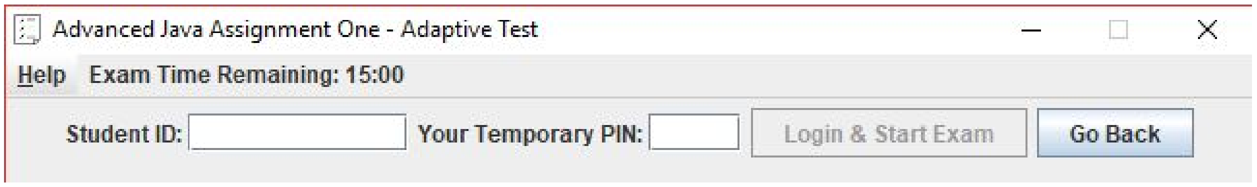
- Creates a new instance of Student, which is an object Model.
- Generates a random, 4-digit pin (done by Student on creation).
- Displays the PIN on screen in the disabled text box.
- Disables the Student ID and School Name text boxes.
- Enables the “Login” button.
- Disables the “Generate Login PIN” button.
- Displays a dialog message instructing the user what to do.

Screenshot 1.7


Advanced Java Assignment One - Adaptive Test
Help
Exam Time Remaining: 15:00
Generate Login PIN
Login
Reset

Student ID: 101601828 School Name: Swinburne Your Temporary PIN: 2460

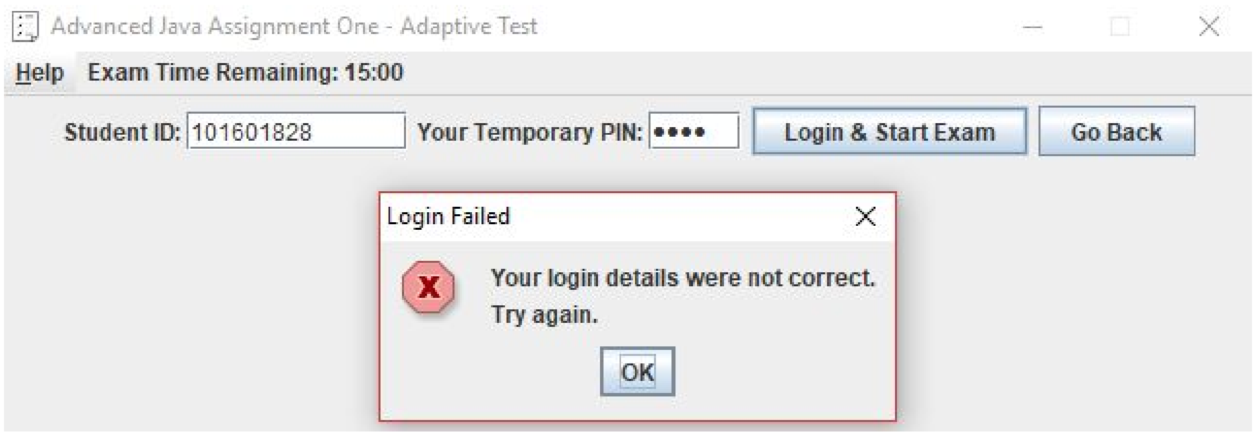
Screenshot 1.8


Advanced Java Assignment One - Adaptive Test
Help
Exam Time Remaining: 15:00
Student ID:
Your Temporary PIN:
Login & Start Exam
Go Back

Once a Student’s information has been collected and their PIN has been generated, the user can proceed to login. Pressing “Login” will tell MainView to give Login View precedence on the screen, which further advances the user through the software (Screenshot 1.7). If at any point the user has done something incorrect, they can press the “Reset” button which basically deletes all user input, and changes the View’s state back to if the user had just first seen it. Resetting the form also destroys any created/saved Student information, and they will no longer be able to proceed to the Login View.

Once the user is at the Login View (Screenshot 1.8) and they have their login information given, they can enter the actual exam part of the software. “Login & Start Exam” will not become enabled until the user has entered their username and a 4-digit PIN. Pressing the “Go Back” button will tell MainView to give the PIN Creation View precedence once again, so the user may re-enter their details and get a new PIN. Going back automatically destroys the saved Student login information as if they had pressed the reset button (Screenshot 1.7).

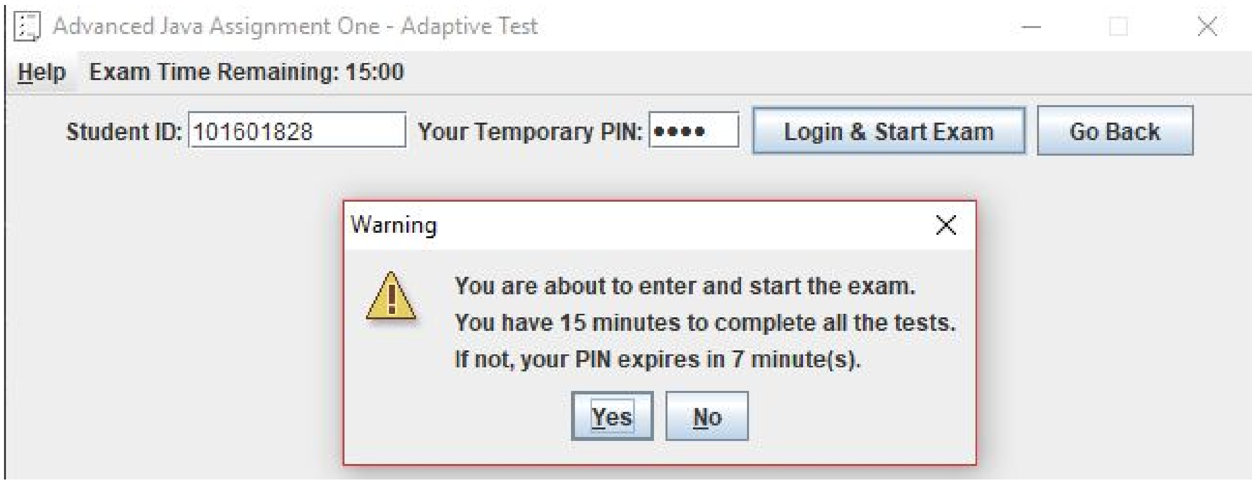
Screenshot 1.9



Student ID: 101601828 Your Temporary PIN: ••••

Your login details were not correct.
Try again.

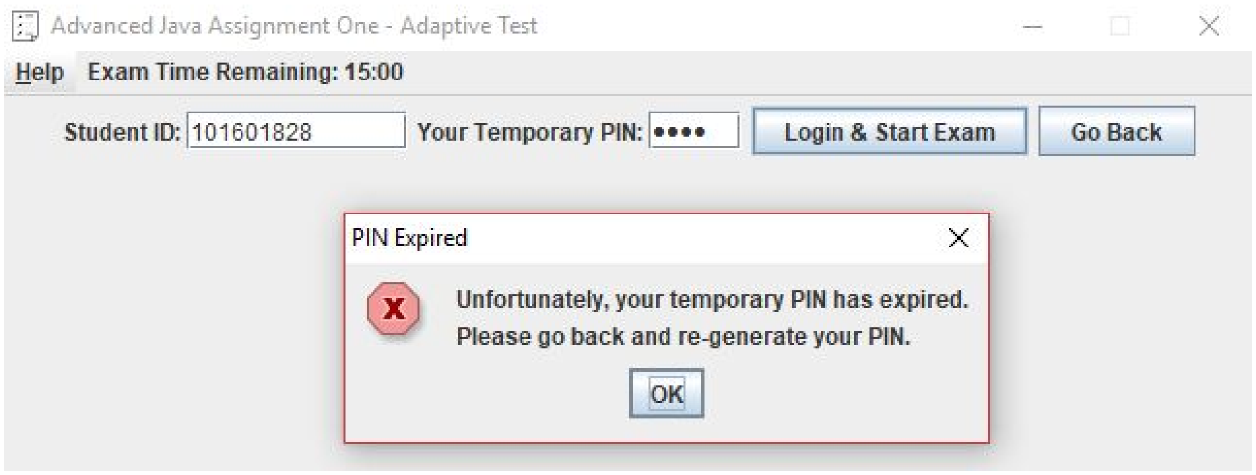
Screenshot 1.10



Student ID: 101601828 Your Temporary PIN: ••••

You are about to enter and start the exam.
You have 15 minutes to complete all the tests.
If not, your PIN expires in 7 minute(s).

Screenshot 1.11

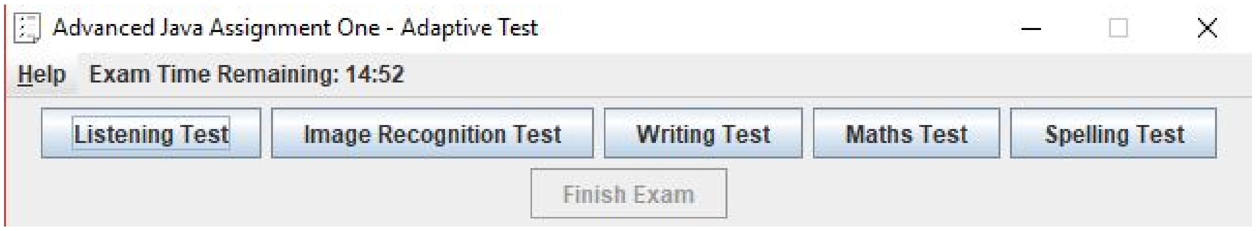


Student ID: 101601828 Your Temporary PIN: ••••

Unfortunately, your temporary PIN has expired.
Please go back and re-generate your PIN.

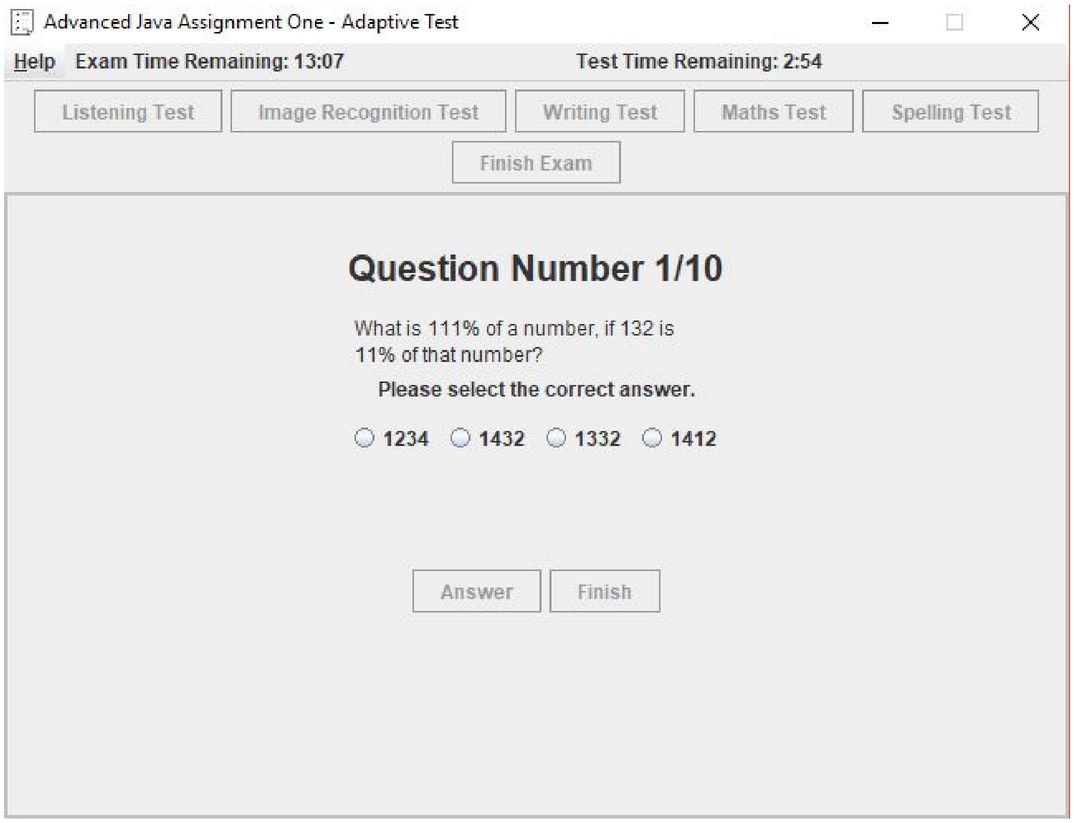
Having a valid (but not necessarily correct) login information allows the user to attempt to “Login & Start Exam”. Screenshot 1.9 depicts what is shown if the user enters incorrect login information. Screenshot 1.10 depicts the warning dialog that is shown if the user has correct login information. If they are 100% certain that they would like to start, clicking the “Yes” button takes them into the exam, clicking the “No” button closes the dialog and nothing happens. Finally, a user only has 20 minutes to login after their PIN is generated. Screenshot 1.11 depicts the dialog shown if the user’s login PIN has expired.

Screenshot 2.1



Once a user has successfully logged in, they will be presented to the Exam View. The 15 minute timer starts counting down straight away, so the user should pick a test to start as quickly as possible. Any test can be started by clicking one of the buttons, as shown in screenshot 2.1.

Screenshot 2.2



Question Number 1/10

What is 111% of a number, if 132 is 11% of that number?

Please select the correct answer.

○ 1234 ○ 1432 ○ 1332 ○ 1412

Clicking on any test button does couple of things:

- Tells the Exam Controller to create a new Test of the chosen category.
- Tells the Test View to start prompting questions.
- Tells the Exam Navbar to disable navigation buttons.

The **Exam Controller** handles all interactions between the Views and the actual Exam model, which holds state information, as well as each Test model. The user will never see these interactions happening, only the result of these interactions. Once the current Test model is active, the Exam Controller asks the **Question Bank** for a random, unique question that is related to the current category of test, as well as the **difficulty** of the question.

The **Question Bank** is a special model that is hidden behind the software runtime that issues the Questions for each Test. That being said, there are two different types of Questions:

- Many-To-One Questions
- One-To-Many Questions

Many-To-One questions arise where there are many different prompts for a type of question, which all have their own unique answer. Take for example, a Math Question. The question “What is 9 + 9?” is one of many math questions that may appear on the exam, but it only has one true, definitive answer, which is 18. This is a Many-To-One Question.

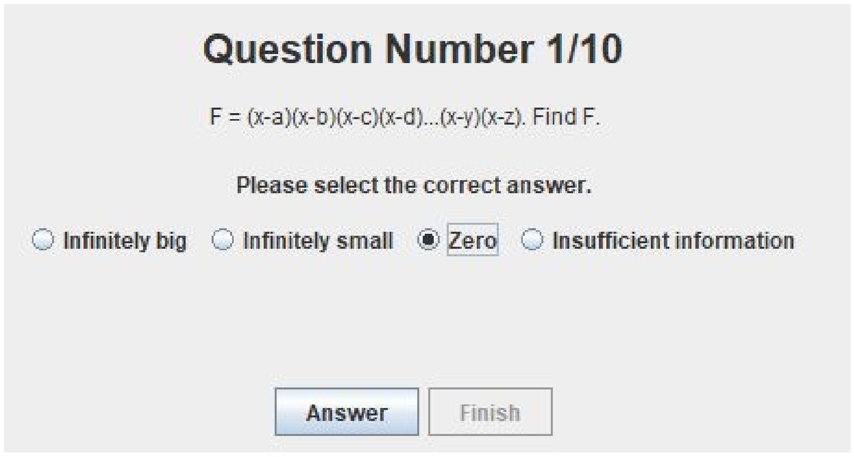
One-To-Many questions arise where there is only one prompt for a type of question, but can have nearly infinitely many answers. Take for example, a Spelling Question. The Question “What is a positive word that starts with ‘G’?” is one of the few spelling questions that may appear on the exam, but it can be answered uniquely, more than once. This is a One-To-Many question.

To distribute this problem, the **Question Bank** employs a **Question Cache** and a **Question Factory**, which both serve the express purpose of providing the Question Bank with Many-To-One and One-To-Many Questions, respectively. Question Cache reads in predefined questions from an internal file (which can be modified to include more questions in the future!) and stores the Question instances until needed. Question Factory sits around waiting for the Question Bank, and creates a new instance of a One-To-Many question on-the-spot when required.

The Question Cache sorts the preloaded Many-To-One questions in lists. This is done by a Map, mapping a category to another Map, mapping a difficulty to a list of questions. This results in each resulting list being sorted by category and difficulty. Each list is shuffled at the start of the program too, so not everyone will receive the same questions in the same order.

For example, if I wanted a Hard Spelling question, firstly, I would retrieve the map of spelling questions, then secondly, I would retrieve the list of hard questions. This is how the Question Cache responds to the requests given by Question Bank.
Now that the process of loading questions is better understood, the process of answering, correcting, and adaptively prompting additional questions is next.

Screenshot 2.3


Question Number 1/10

F = (x-a)(x-b)(x-c)(x-d)...(x-y)(x-z). Find F.

Please select the correct answer.

Infinitely big [✓ Zero] Infinitely small Insufficient information

At the start of any Test, the Exam Controller will also issue a Question of medium (tougher) difficulty to the Test; a medium Maths question is shown in screenshots 2.2 and 2.3.

The Test View is responsible for prompting questions, as shown, as well as passing user answers back to the Exam Controller. Each individual question category has a different way of being displayed and being answered, so the specifics will be discussed in their respective categories. After a user has entered any answer, the “Answer” button will become enabled, allowing the user to submit their answer for the current question (screenshot 2.3).

Pressing “Answer” will pass the user’s answer back to the Exam Controller, where it will tell the Question to mark itself based on the answer given. A few things happen:

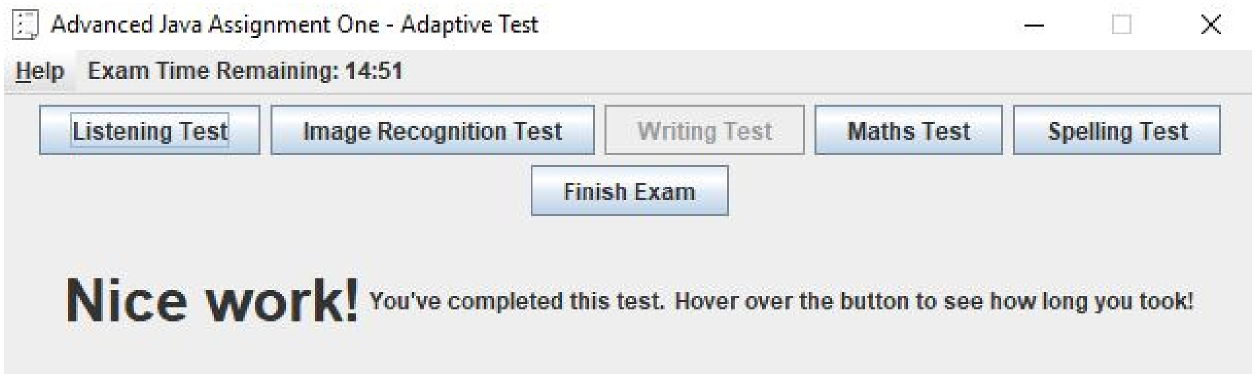
- The Question is marked as “answered”.
- If the answer is correct, the Question is marked as “correct”.
- Marks are recorded for the Test itself and the Exam overall.
- The difficulty is adjusted. (up if correct, down if incorrect)

The same process is followed for each question, regardless of category. The details are abstracted to the individual child classes of Question, so we need not worry about the specifics in this context. The Exam View, after having a Question answered, will do one of two things:

- Complete the test, if all questions have been answered.
- Prompt the next question in the test.

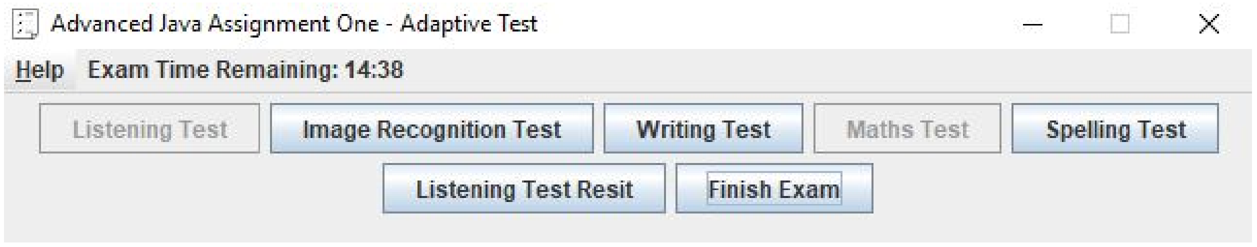
Once at least one question has been answered, the “Finish” button will become enabled, allowing the user to finish the test early if desired. This marks the test as incomplete. Running out of time on a test also marks the test as incomplete. If a test is repeatable, it may be resumed at a later time at the cost of 5 marks.
Once a test has been finished (complete or incomplete), the Exam View notifies the Exam Navbar to re-enable buttons for tests that are still awaiting to be taken, as well as the “Finish Exam” button if at least one test has been attempted. The user may choose to end the exam early, or begin another test in an uncompleted category.

Screenshot 2.4

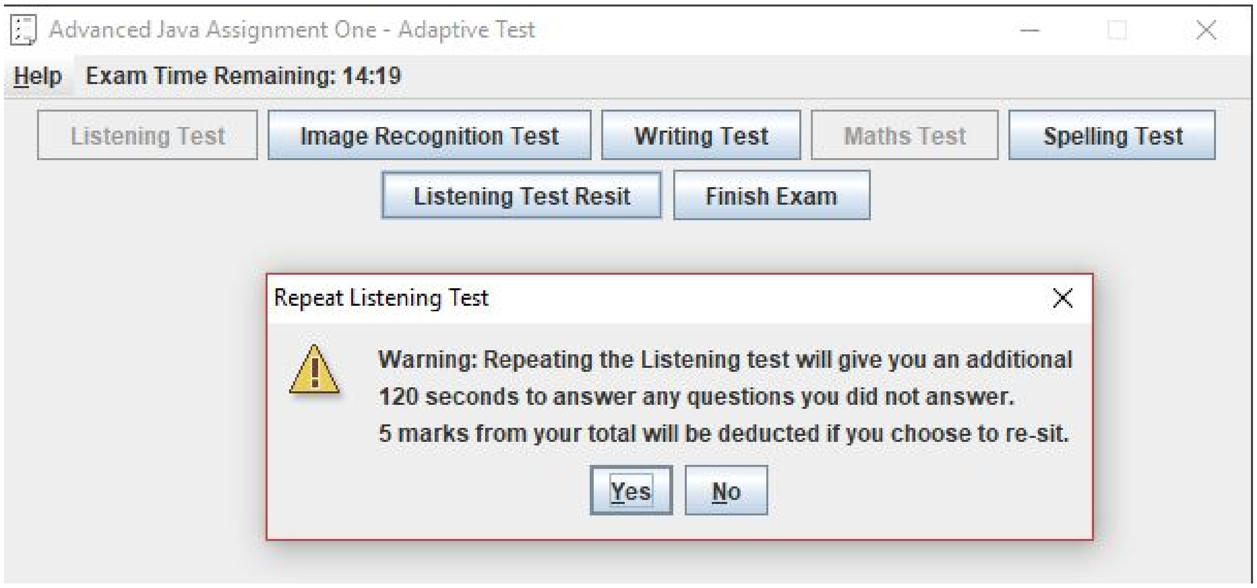


Nice work! You've completed this test. Hover over the button to see how long you took!

Screenshot 2.5



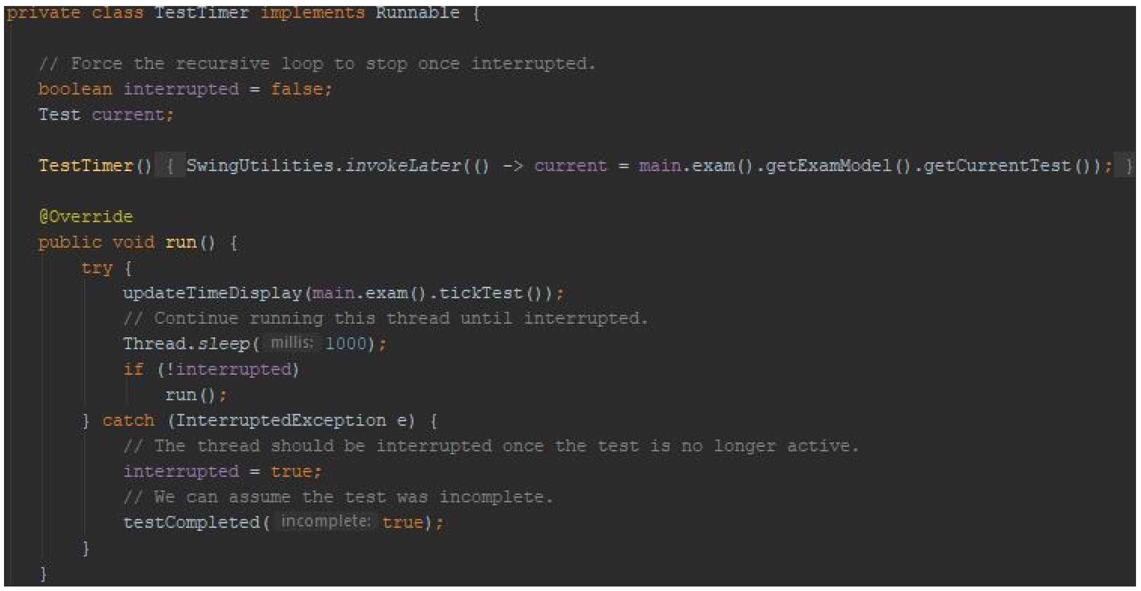
Screenshot 2.6



Warning: Repeating the Listening test will give you an additional 120 seconds to answer any questions you did not answer.
5 marks from your total will be deducted if you choose to re-sit.

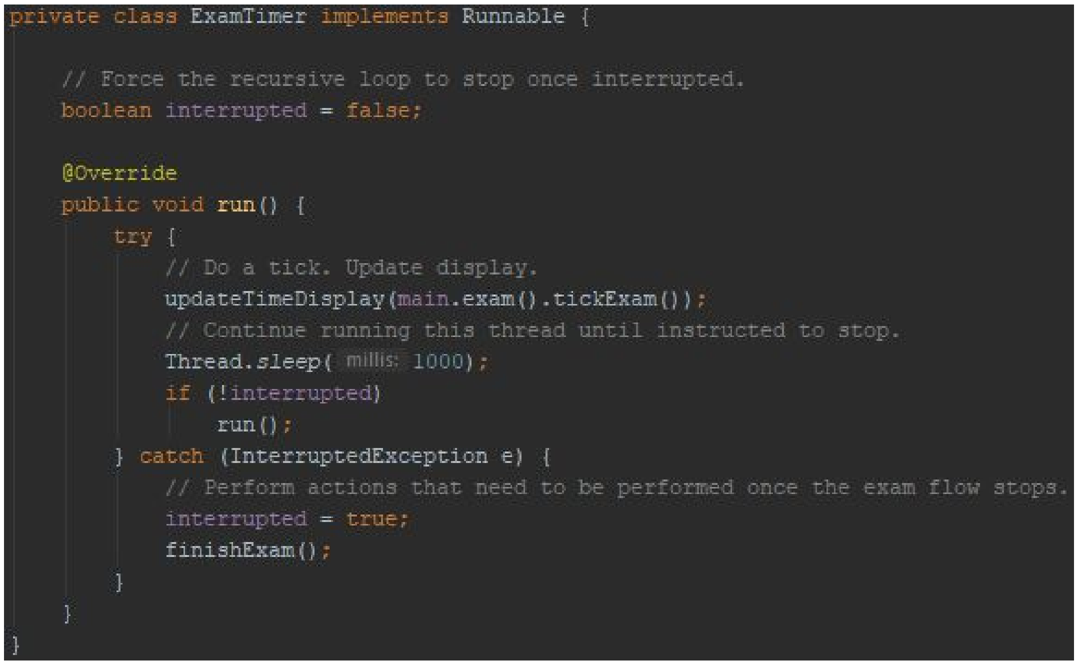
If at any point in the exam did the user not complete the Listening test in time, the “Listening Test Resit” button will appear if the user has accumulated at least five marks in the exam. Clicking it will prompt the user with a warning about the process and consequences of repeating the Listening test. If the user clicks “Yes”, they will be brought to where they left off in the Listening test, with an additional 120 seconds at hand. The software is written in a way that allows any test to be repeated, but only once, and only if the Test is marked as “repeatable”. The resit button and prompt are shown in screenshots 2.5 and 2.6.

Screenshot 3.1



```
private class TestTimer implements Runnable {

    // Force the recursive loop to stop once interrupted.
    boolean interrupted = false;
    Test current;

    TestTimer() { SwingUtilities.invokeLater(() -> current = main.exam().getExamModel().getCurrentTest()); }

    @Override
    public void run() {
        try {
            updateTimeDisplay(main.exam().tickTest());
            // Continue running this thread until interrupted.
            Thread.sleep( millis: 1000);
            if (!interrupted)
                run();
        } catch (InterruptedException e) {
            // The thread should be interrupted once the test is no longer active.
            interrupted = true;
            // We can assume the test was incomplete.
            testCompleted( incomplete: true);
        }
    }
```

Screenshot 3.2



```
private class ExamTimer implements Runnable {

    // Force the recursive loop to stop once interrupted.
    boolean interrupted = false;

    @Override
    public void run() {
        try {
            // Do a tick. Update display.
            updateTimeDisplay(main.exam().tickExam());
            // Continue running this thread until instructed to stop.
            Thread.sleep( millis: 1000);
            if (!interrupted)
                run();
        } catch (InterruptedException e) {
            // Perform actions that need to be performed once the exam flow stops.
            interrupted = true;
            finishExam();
        }
    }
}
```

Timer Threads

These are special, repeating procedures that run at certain intervals and are responsible for keeping track of how much timer the user has left to complete individual tests, as well as for the entire exam.

The **Test Timer** runs every second, and updates the “Test Time Remaining” display on the menu bar based on the number returned from the method tickTest() in ExamController. tickTest() essentially decreases the test time remaining, and forces the thread to stop once it reaches zero. Once the thread has stopped (been interrupted), it forces the test to finish, marking it as incomplete. This is the same as if the user were to click the “Finish” button during a test. The code for the Test Timer can be seen in screenshot 3.1.

The **Exam Timer** runs every second, and updates the “Exam Time Remaining” display on the menu bar based on the number returned from the method tickExam() in ExamController. tickExam() decreases the total exam time remaining, and forces the thread to stop once it reaches zero. Once the thread has stopped, it forces the exam to stop. This is the same as if the user were to click the “Finish Exam” button on the navbar. Any test that is being taken is interrupted. The code for the Exam Timer can be seen in screenshot 3.2.

It should be noted that once either the exam or test counters reach 30 seconds remaining, a warning message is played to notify the user. A metronome sound is played when there is 3 seconds left.

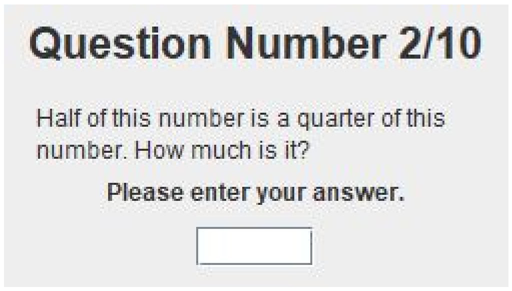
Questions

Each question type has a unique way of being prompted, answered, and marked. That being said, all question types share a lot in common, such as:

- What type the question actually is.
- What difficulty the question is.
- Being marked as ‘answered’.
- Being marked as ‘correct’.
- Having a String prompt.

While each question does have a unique way of doing the stuff above, each question type must be able to present itself, be answered, and mark itself. At the Question level, these details are abstracted away to the child classes. There are also a few intermediate classes that serve to house common code between similar question types- Choice and Immutable Questions.

Choice Question
Choice questions are a type of Many-To-One question, and encompass both Math and Listening questions. Since both Math and Listening Questions required one or more strings as an answer, and can have zero or more choices to select from, we can make them share a common class, Choice Question. This cuts down a lot of code that is needed to display both kinds of questions.
The way the question is marked and answered can also be implemented in this way.
Screenshot 4.1

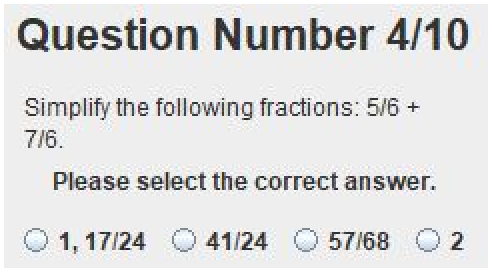


Question Number 2/10

Half of this number is a quarter of this number. How much is it?

Please enter your answer.

Screenshot 4.2



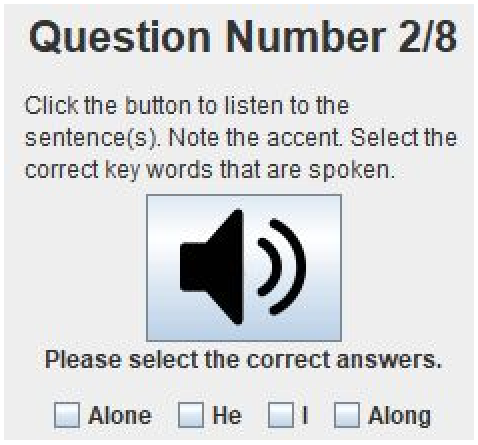
Question Number 4/10

Simplify the following fractions: 5/6 + 7/6.

Please select the correct answer.

○ 1, 17/24 ○ 41/24 ○ 57/68 ○ 2

Screenshot 4.3


Question Number 2/8

Click the button to listen to the sentence(s). Note the accent. Select the correct key words that are spoken.

Please select the correct answers.

☐ Alone ☐ He ☐ I ☐ Along

Choice Questions are always answered in the same way, by giving one or more strings as an answer. For example, a Math or Listening Question may require you to give an answer from a text box or radio selection (screenshots 4.1 and 4.2), or from a selection of checkboxes (screenshot 4.3). No matter what configuration the question is in, the answer always ends up being the same- an array of strings. When a Choice Question is marked, it compares the array of strings against the answer, which is also an array of strings. The answer is correct if all the elements of the actual answer array are also in the user’s answer array.

When presenting a Choice Question to a user (either Listening or Math), the form of input may vary. If there is many choices and more than one correct answer, it displays as a group of checkboxes (screenshot 4.3). If there is many choices and only one correct answer, it displays as a group of radio buttons (screenshot 4.2). If there are no choices and only one correct answer, it displays a text box, for which the user must enter their answer (screenshot 4.1). Questions with no choices and more than one correct answer are not supported.

Immutable Question

Immutable Questions are a type of One-To-Many question, and encompass both Spelling and Writing questions. Since both Spelling and Writing questions can have infinitely many answers, it should be ensured that the user does not enter the same correct answer twice to get an easy “correct” mark. Immutable Question holds common methods to compare its answer to other Immutable Questions. Immutable Questions hold a reference to the test they belong to, so they can iterate through previous questions and compare their answer to previous answers to ensure they did not enter the same answer twice.

In the case that the user does enter the same correct answer twice, the Immutable Question will prevent the question from being marked as correct. This problem is solved. It should also be noted that Immutable Questions require a specific type (T) answer. This is so the user’s answer can be readily retrieved and compared to other Immutable Questions.

Question Presentation

Each question prompt panel is presented in a Grid Bag layout, using weights and additional JPanels to determine sizes. The first JPanel contains the question number, as shown in numerous Question prompt screenshots, e.g. 4.1, 4.2, 4.3. The actual String prompt is displayed in an uneditable, transparent JTextArea below the question number panel. Below the question number and prompt, the actual JPanel for answering the question is generated. Every type of question has a different answering panel, which is explained below. Below the question prompt panel is the buttons panel, which contains all the relevant buttons needed for the question. Most of the time these are just the “Answer” and “Finish” buttons.

Maths, **Listening**

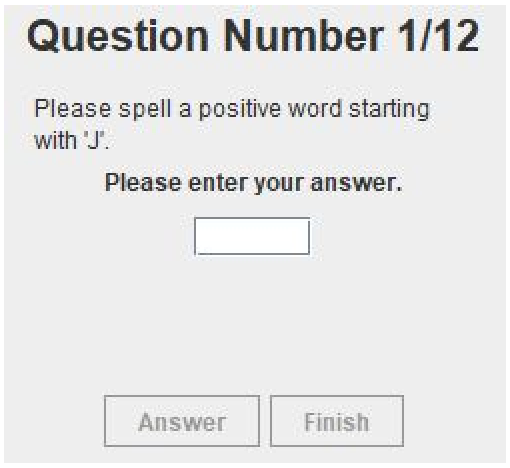
Since both questions are displayed in the same way, we can discuss the presentation of both questions through the means of Choice Question. Screenshot 4.3 depicts a listening question. Screenshots 4.1 and 4.2 depict a maths question. The elements of a Choice Question have already been discussed as above, but a quick summary will be provided here:

- Zero choices = Text Box Written Answer
- One answer, many choices = Radio Selection Answer
- Many answers, many choices = Checkbox Selection Answer

It should also be noted that depending on difficulty, the apparent toughness of the math question will increase. The number of sentences that are spoken in Listening questions will increase depending on the difficulty too.

Spelling

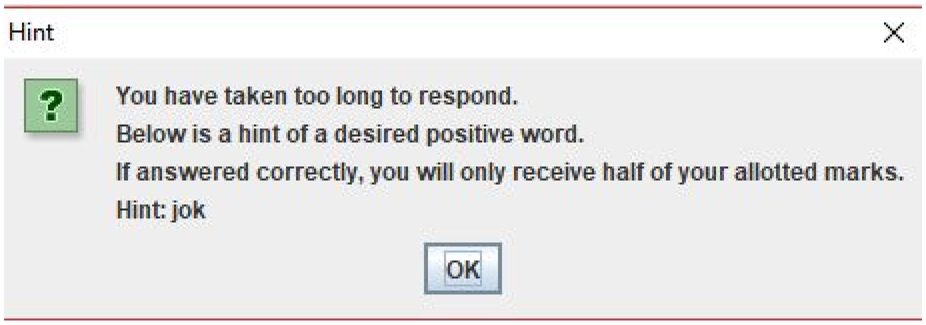
Screenshot 4.4


Question Number 1/12

Please spell a positive word starting with 'J'.

Please enter your answer.

Screenshot 4.5


Hint

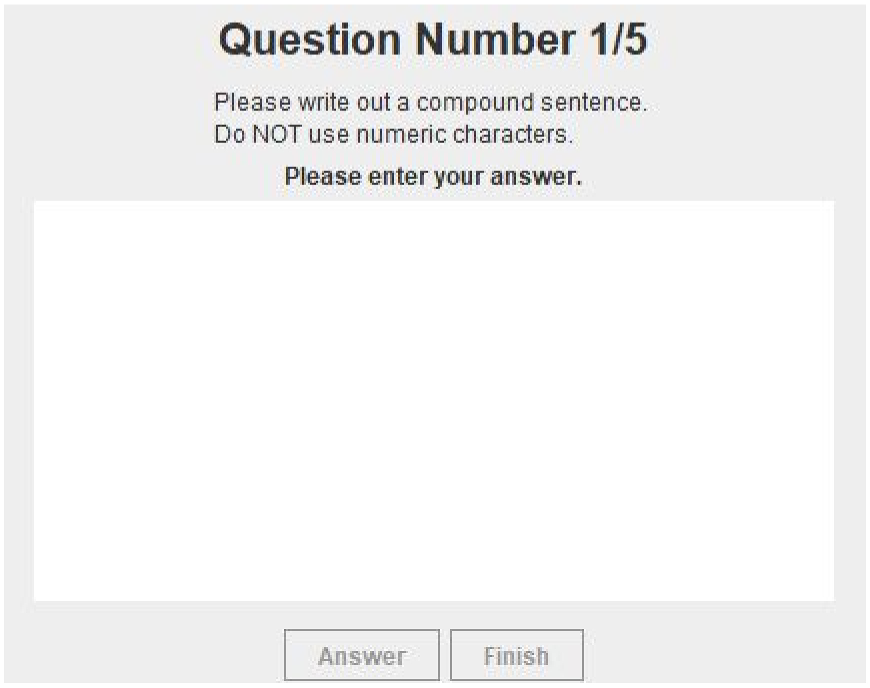
You have taken too long to respond.
Below is a hint of a desired positive word.
If answered correctly, you will only receive half of your allotted marks.
Hint: jok

Spelling questions use a single text box to receive a user's answer. Depending on the difficulty of the question, it will be harder to devise a "positive" word starting with a specific letter. A medium spelling question is depicted in screenshot 4.4. If a user takes longer than 10 seconds to answer, they will be given a hint as shown in screenshot 4.5. After receiving the hint, the user will not be penalised with a difficulty reduction, but will only receive half the marks that the user would have originally received if they are correct.

In order to accomplish this 10 second delay, Spelling Question makes a Thread wait ten seconds after the question has been prompted to the user, and then prompts the dialog to the user if they have not answered the question, and if the test is still active.

The positive word must start with 'G', 'J', or 'Z' for easy, medium, and hard questions respectively.

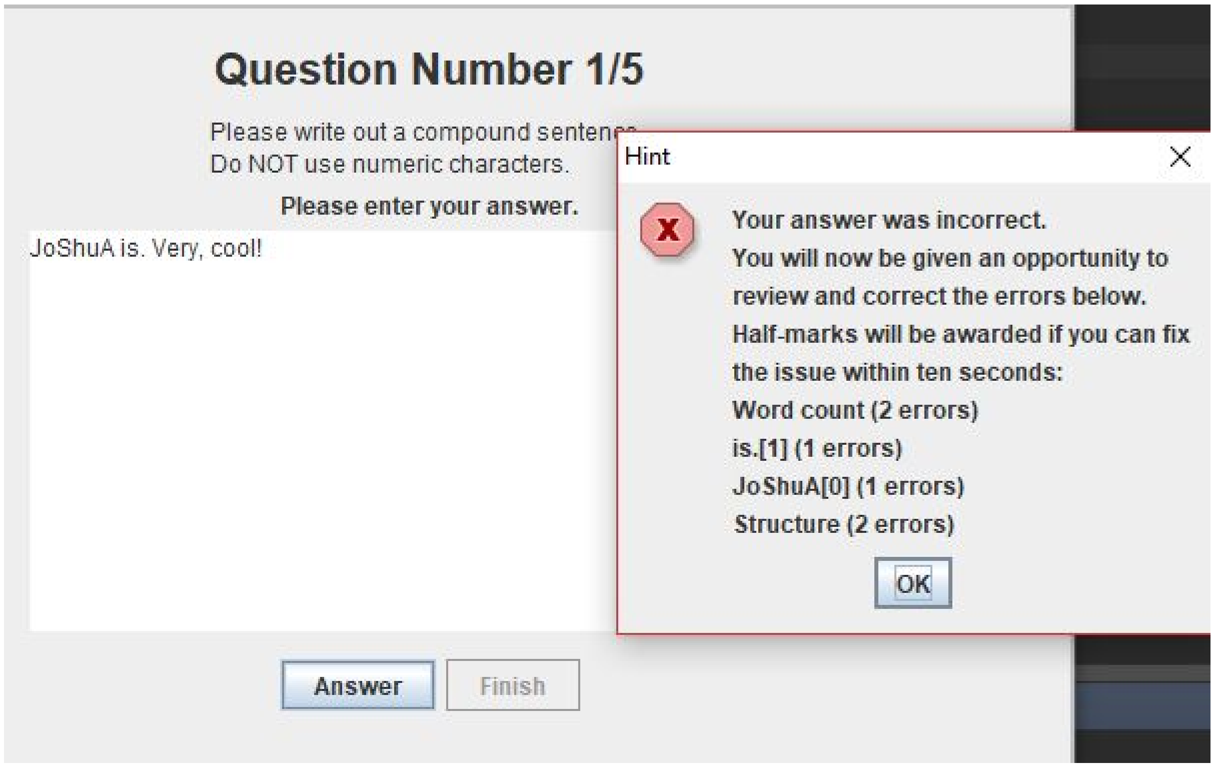
Writing
Screenshot 4.6


Question Number 1/5

Please write out a compound sentence.
Do NOT use numeric characters.

Please enter your answer.

Screenshot 4.7


Question Number 1/5

Please write out a compound sentence.
Do NOT use numeric characters.

Please enter your answer.

JoShuA is. Very, cool!

Hint

Your answer was incorrect.
You will now be given an opportunity to review and correct the errors below.
Half-marks will be awarded if you can fix the issue within ten seconds:
Word count (2 errors)
is.[1] (1 errors)
JoShuA[0] (1 errors)
Structure (2 errors)

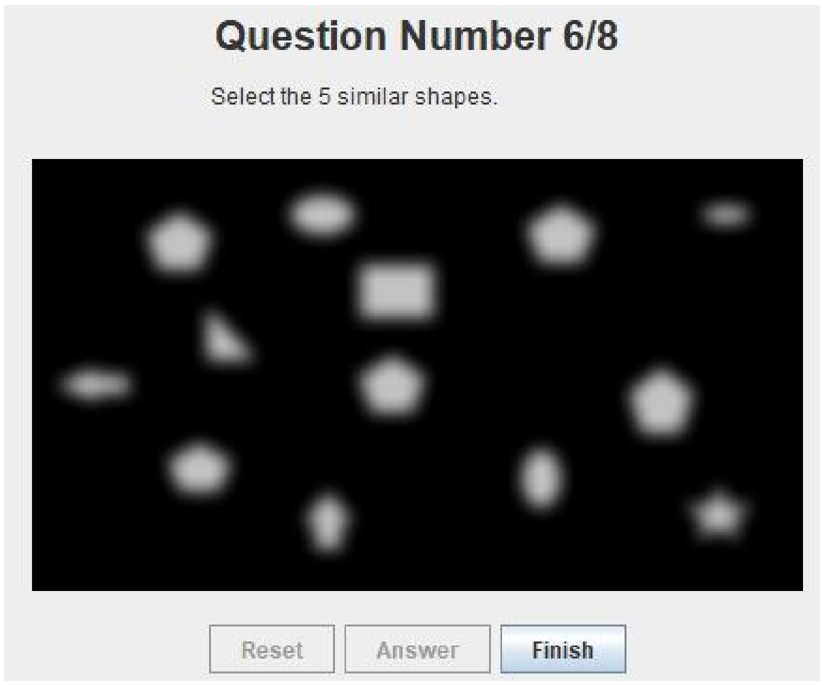
Writing questions use a text area to receive a user's answer. Depending on the difficulty of the question, the complexity of the question will increase. A medium writing question prompt is depicted in screenshot 4.6. Once the user presses “answer”, a preliminary check-over of their answer will be performed before submitting it for marking. If the Writing Question detects and errors in the user’s answer, they will be prompted with a dialog explaining what needs to be fixed in order for the answer to be considered correct.

If a user’s answer is incorrect and they are prompted with a correction dialog, they will only have 10 seconds to receive half marks. If they take longer than 10 seconds to correct the errors or if the answer is still incorrect, they will receive zero marks.

In order to accomplish the ten second delay, much like the spelling question, Writing Question makes a Thread wait ten seconds after the correction dialog has been prompted to the user. If it has been longer than 10 seconds and the question is still not answered, the amount of marks given for correctly answering the question is changed to zero.

Listening (additional, separate from Choice Question)
Listening Questions are their own child class, whereas Maths Questions are simply just an instance of Choice Question with a Maths question category. This is because Listening Questions hold an extra string pointing to the file name that plays the audio track necessary for the question. When displaying a Listening Question, an extra button also appears that allows the user to play the audio file. Maths Questions do not have this button.

Image
Screenshot 4.8



Question Number 6/8

Select the 5 similar shapes.

Screenshot 4.9

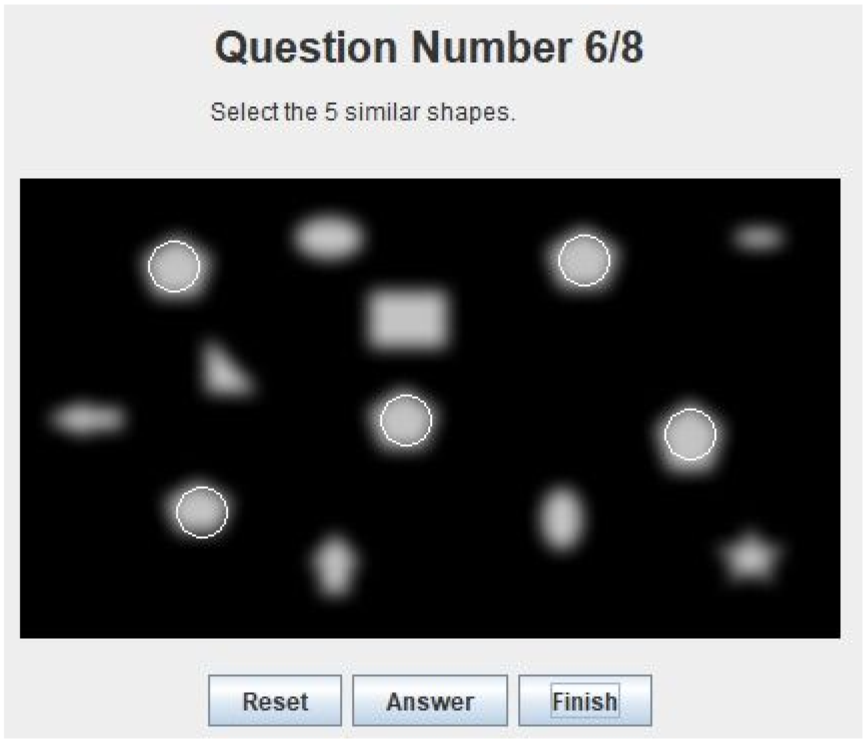


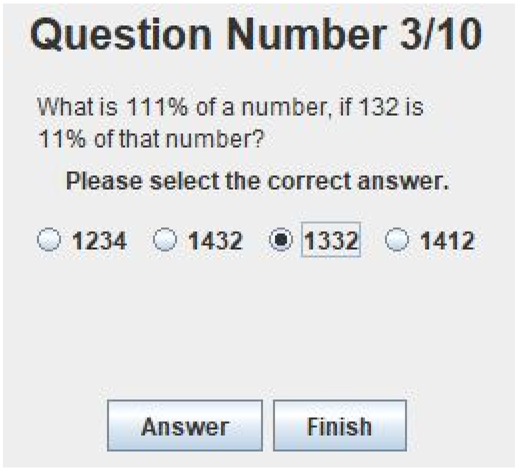
Image Questions are different to the other 4 types of questions. While they all use Action Listeners to process the input from checkboxes, buttons, or text boxes, Image Question uses a mouse listener to process clicks on an image. In screenshot 4.8, there is an image with a prompt to “select the 5 similar shapes”. In screenshot 4.9, five clicks have been made, shown by the white circles encompassing the click points.

When Image Questions are loaded, there are certain points you have to click, and as long as the actual answer’s point falls within the circle, it is considered correct. You can only click as many times as there are correct points to be clicked. For example, if a question requires you to select two things, you can, at most, only click twice to place two circles on the image. It should also be noted that an extra “Reset” button is included in the button panel by the Image Question so the user can clear the placed circles and try again.

Lastly, Image Question is processed differently from every other question. When “Answer” is clicked on any other question, the answer is given to the question at the same time the question is **finalised**. Finalising a question basically means that the user is done with their answer and it is ready to be checked. With Image Question, every “click” counts as an additional answer given to the question, whereas pressing “Answer” **finalises** your choices.

Checking Question Answers

Maths, Listening
Screenshot 5.1



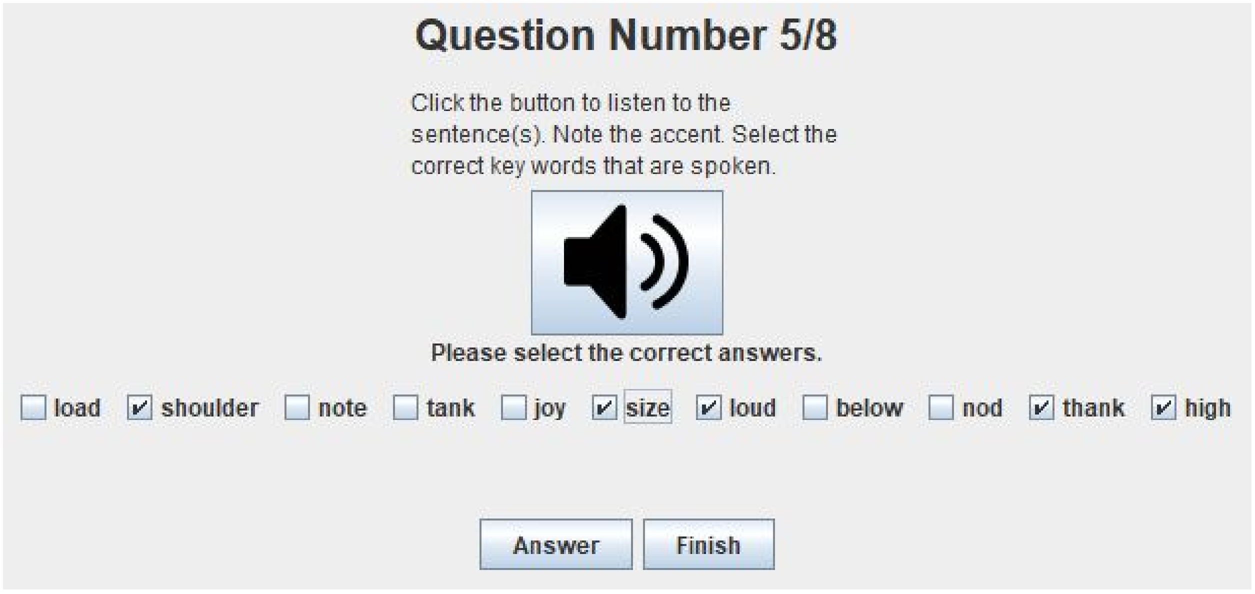
Question Number 3/10

What is 111% of a number, if 132 is 11% of that number?

Please select the correct answer.

○ 1234 ○ 1432 [● 1332] ○ 1412

Screenshot 5.2



Question Number 5/8

Click the button to listen to the sentence(s). Note the accent. Select the correct key words that are spoken.

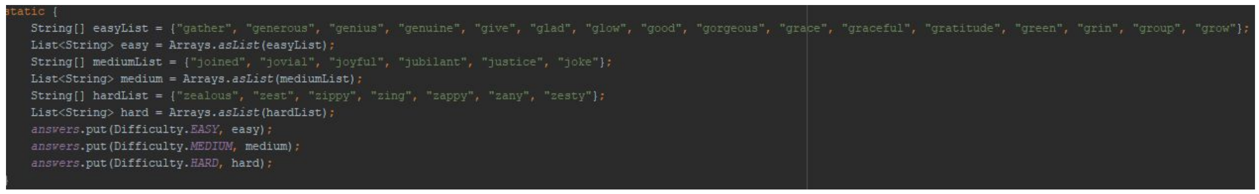
Please select the correct answers.

☐ load [✓ shoulder] ☐ note ☐ tank ☐ joy [✓ size] [✓ loud] ☐ below ☐ nod [✓ thank] [✓ high]

An example user answer for a choice question (maths category) is depicted in screenshot 5.1. When passed into the Choice Question through the Exam Controller, it is treated as a string array with one element, or in this case [“1332”]. Let’s suppose the correct answer is 1332, that means the actual answer is [“1332”] as well. When checking if a choice question is correct, we compare the array of user answers to the array of actual answers, to make sure each element is in both arrays.

Another example user answer for a choice question (listening category) is depicted in screenshot 5.2. When passed into Choice Question through the Exam Controller, it is treated as a string array with 5 elements, or in this case [“shoulder”, “size”, “loud”, “thank”, “high”]. The correct answer is [“load”, “shoulder”, “note”, “tank”, “joy”, “size”]. The Question can automatically assume this answer is not correct, because it does not even match the array length of the actual answer. If it were the same length, it would check that every element of the user answer is also in the actual answer. If it finds a user answer that isn’t in the actual answer, or if not all actual answers were found in the user answer, the answer would be marked as wrong.

Spelling
Screenshot 5.3



```
tatic {
    String[] easyList = {"gather", "generous", "genius", "genuine", "give", "glad", "glow", "good", "gorgeous", "grace", "graceful", "gratitude", "green", "grin", "group", "grow"};
    List<String> easy = Arrays.asList(easyList);
    String[] mediumList = {"joined", "jovial", "joyful", "jubilant", "justice", "joke"};
    List<String> medium = Arrays.asList(mediumList);
    String[] hardList = {"zealous", "zest", "zippy", "zing", "zappy", "zany", "zesty"};
    List<String> hard = Arrays.asList(hardList);
    answers.put(Difficulty.EASY, easy);
    answers.put(Difficulty.MEDIUM, medium);
    answers.put(Difficulty.HARD, hard);
```

The way spelling questions are marked are blissfully simple. Since there is a finite amount of positive words that start with ‘G’, ‘J’, or ‘Z’, simply hard-coding them into static arrays would be more than sufficient for the requirements of this question. This is shown in screenshot 5.3.

When checking if a user’s answer is correct, two things happen:

- It checks that the answer given has not already been used (Immutable Question!)
- It checks the list pertaining to the current difficulty for the user’s answer.

If the user’s answer is unique (immuted), and is inside the list of correct words, then the question is marked as correct. The list of correct words are static because One-To-Many questions are created on the spot as they are needed.

Writing
The marking procedure is different for Writing Questions than it is for Spelling Questions, as there is theoretically infinitely many answers. Still, it follows similar rules:

- Checks if the answer given has not already been used (immutable)
- Checks if the answer conforms to the complexity the difficulty requires

Requirements for an easy (simple) writing prompt:

- The first letter starts with a capital.
- No words contain capital letters, except for the first letter.
- There is a full stop of the end of the sentence.
- Only one full stop is used.
- Numeric characters are not used.
- At least five words are used.

Requirements for a medium (compound) writing prompt:

- The existing sentence satisfies the “simple” requirements.

- At least one common conjunction is used.
- At least one comma is used.
- At least ten words are used.

Requirements for a hard (complex) writing prompt:
- The existing sentence satisfies the “compound” requirements.
- At least one common relative pronoun is used.
- At least one word is capitalised that isn’t the first word.
- At least fifteen words are used.

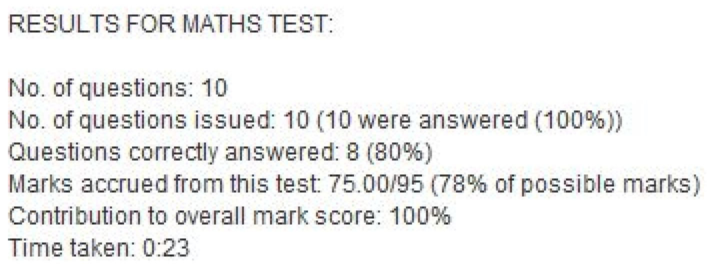
If the user’s answer is unique (immuted), and is the correct level of sentence complexity is met for the difficulty given, then the question is marked as correct. The common conjunctions and common relative pronouns lists are static because One-To-Many questions are created on the spot as needed.

Image
The way Image Questions are marked is straightforward. It checks if each user answer is within a required point on the image. The way it goes about this, however, is a little complex, but is very robust. Firstly, if the amount of points given is not the same as the amount of points required, the Image Question will automatically assume that the question is wrong and not mark it as correct.

For each answer given, it checks each required answer. If it is within the range of a required point, it marks off that required point as being selected. It does this for all user answer points. If every required answer point has been selected, then the question is marked as correct.

Calculating Results
Test Results
When calculating results, a bunch of variables are calculated and displayed for each test panel:
- Total number of questions for the test
- Number of questions actually issued on the test
- Number of questions answered
- Number of questions correctly answered
- Time taken, in minutes and seconds.
- Marks accrued from this test.
- The maximum number of marks possible that could have been received on this test.
- Percentage of possible marks obtained
- Percentage of questions answered
- Percentage of issued questions answered correctly
- Percentage of contribution to total mark from this test
- ((these can be seen in screenshot 6.1 below)

Screenshot 6.1

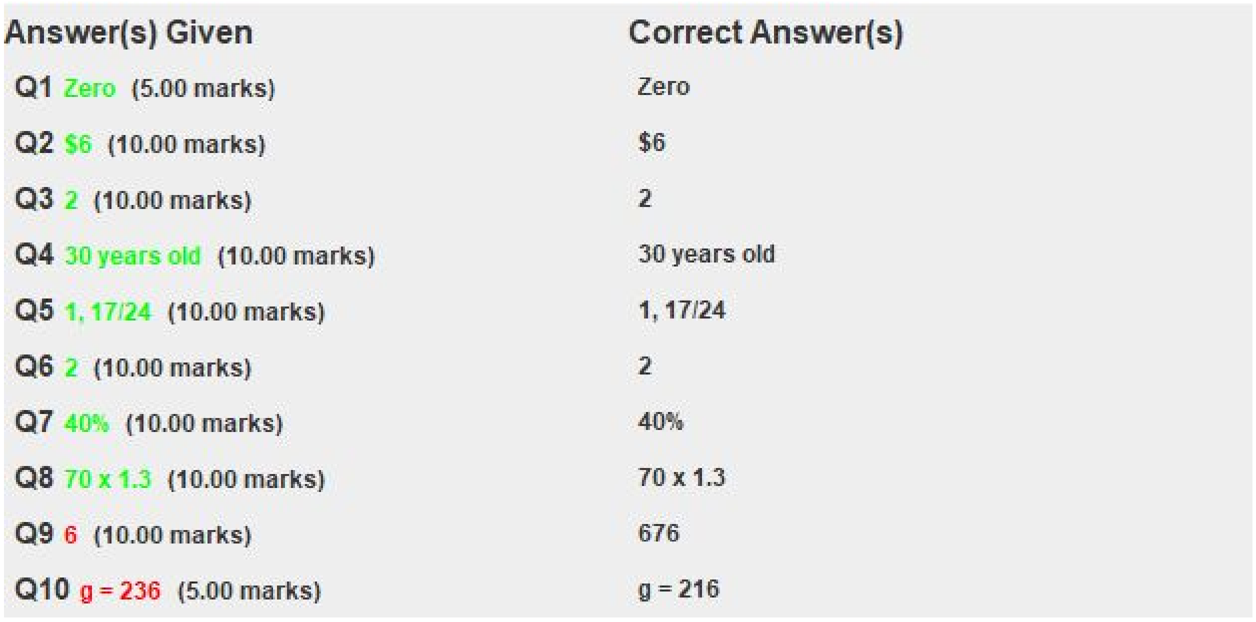


```
RESULTS FOR MATHS TEST:

No. of questions: 10
No. of questions issued: 10 (10 were answered (100%))
Questions correctly answered: 8 (80%)
Marks accrued from this test: 75.00/95 (78% of possible marks)
Contribution to overall mark score: 100%
Time taken: 0:23
```

Question Results

Every test result panel uses a GridLayout of (n + 1) x 2 dimension, where n is the number of questions issued on the test. The left column is used for displaying the user's answer, and whether or not is was correct or not. The right side is used for displaying the actual answer. The entire layout is contained within a JScrollPane so that if the results is lengthier than the screen allows, the user will be able to scroll down to see more information.

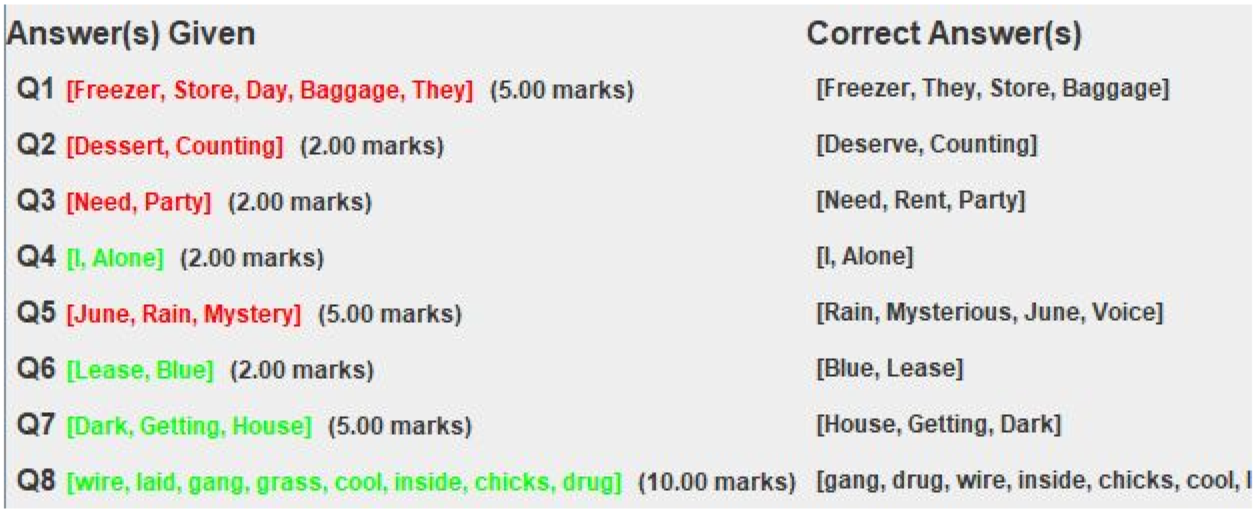
Maths Results
Screenshot 6.2



Answer(s) Given	Correct Answer(s)
Q1 Zero (5.00 marks)	Zero
Q2 $6 (10.00 marks)	$6
Q3 2 (10.00 marks)	2
Q4 30 years old (10.00 marks)	30 years old
Q5 1, 17/24 (10.00 marks)	1, 17/24
Q6 2 (10.00 marks)	2
Q7 40% (10.00 marks)	40%
Q8 70 x 1.3 (10.00 marks)	70 x 1.3
Q9 6 (10.00 marks)	676
Q10 g = 236 (5.00 marks)	g = 216

For each maths question on the maths test, the left side is populated with the question number, the answer given (colored in green if correct, red if incorrect), and the marks that are received if the answer was correct.

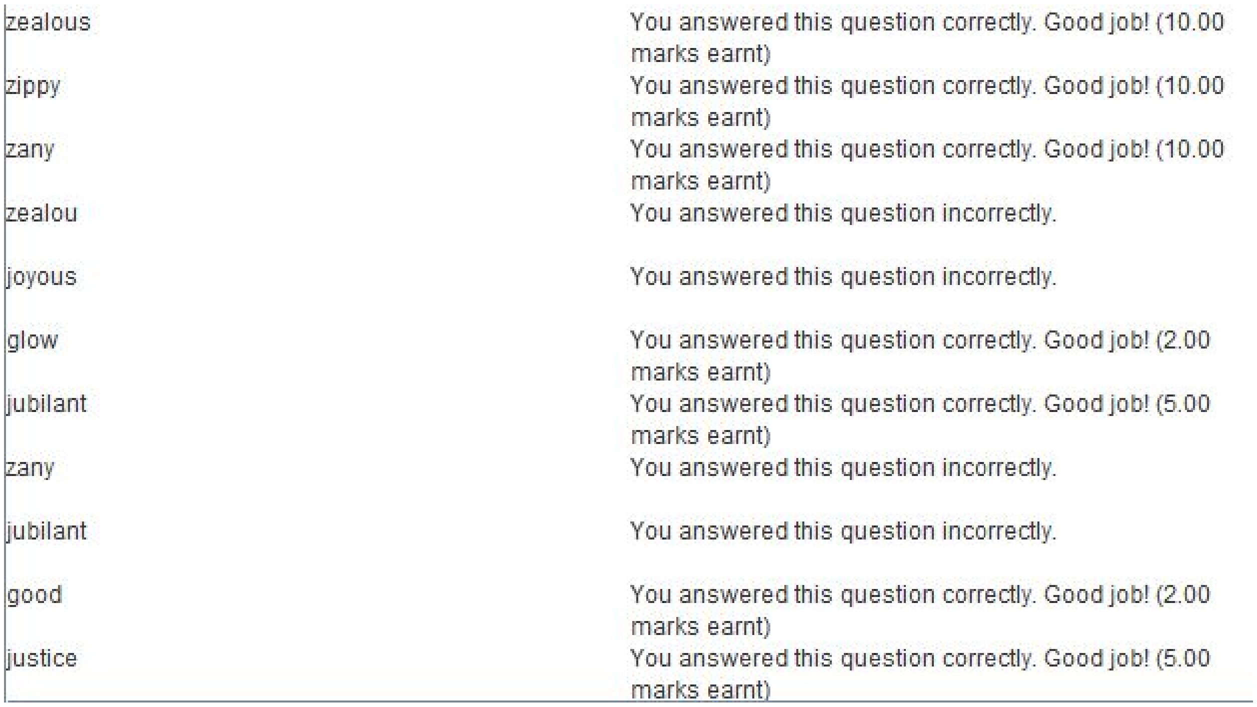
Listening Results
Screenshot 6.3



Answer(s) Given		Correct Answer(s)
Q1 [Freezer, Store, Day, Baggage, They]	(5.00 marks)	[Freezer, They, Store, Baggage]
Q2 [Dessert, Counting]	(2.00 marks)	[Deserve, Counting]
Q3 [Need, Party]	(2.00 marks)	[Need, Rent, Party]
Q4 [I, Alone]	(2.00 marks)	[I, Alone]
Q5 [June, Rain, Mystery]	(5.00 marks)	[Rain, Mysterious, June, Voice]
Q6 [Lease, Blue]	(2.00 marks)	[Blue, Lease]
Q7 [Dark, Getting, House]	(5.00 marks)	[House, Getting, Dark]
Q8 [wire, laid, gang, grass, cool, inside, chicks, drug]	(10.00 marks)	[gang, drug, wire, inside, chicks, cool, l

Results calculation for Listening questions is the same as Maths questions, as they are both Choice Questions.

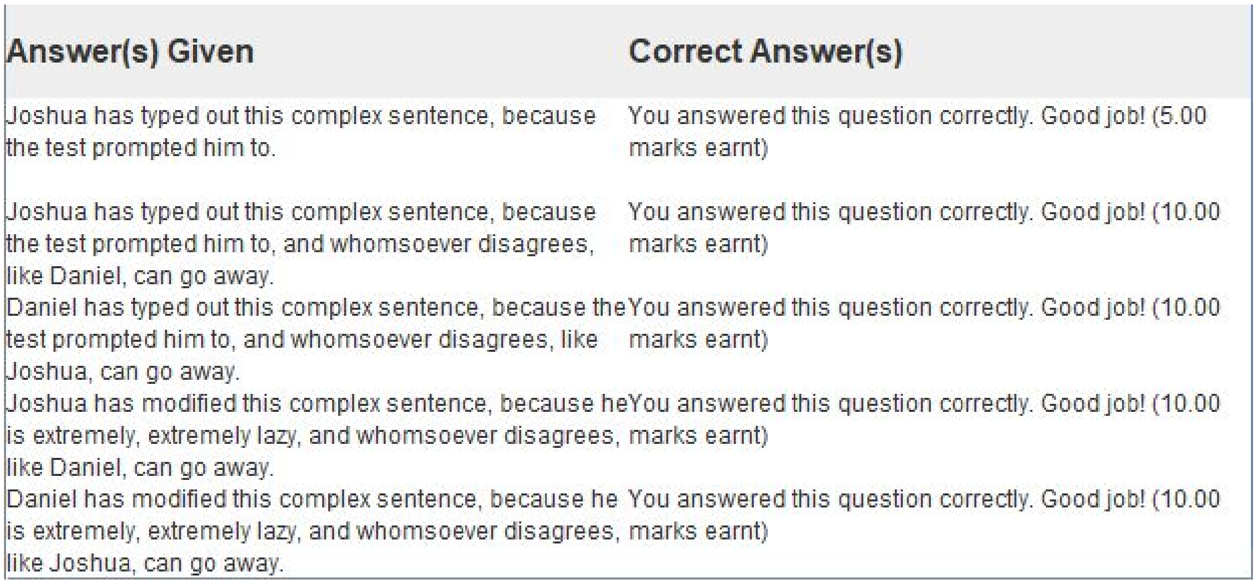
Spelling Results
Screenshot 6.4



zealous	You answered this question correctly. Good job! (10.00 marks earnt)
zippy	You answered this question correctly. Good job! (10.00 marks earnt)
zany	You answered this question correctly. Good job! (10.00 marks earnt)
zealou	You answered this question incorrectly.
joyous	You answered this question incorrectly.
glow	You answered this question correctly. Good job! (2.00 marks earnt)
jubilant	You answered this question correctly. Good job! (5.00 marks earnt)
zany	You answered this question incorrectly.
jubilant	You answered this question incorrectly.
good	You answered this question correctly. Good job! (2.00 marks earnt)
justice	You answered this question correctly. Good job! (5.00 marks earnt)

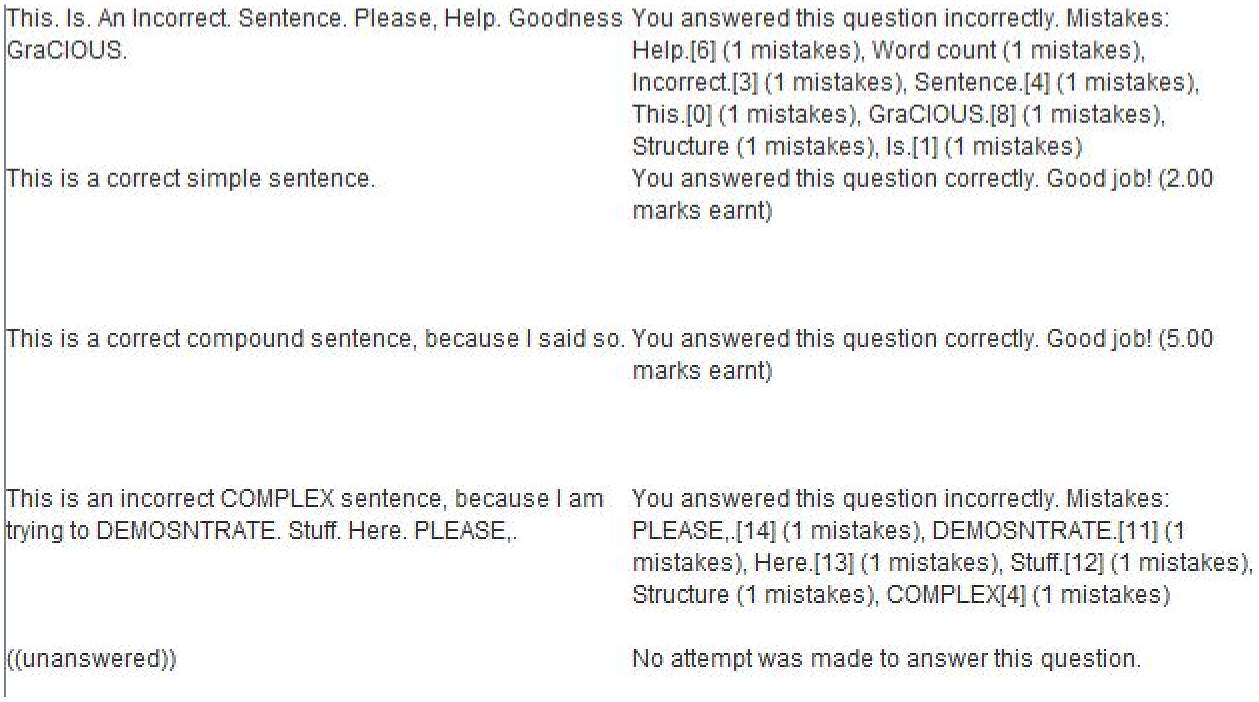
Spelling results are displayed more simply than Maths Questions, as they are a One-To-Many question. On the left is the user's answer, and feedback is given on the right side this time.

Writing Results
Screenshot 6.5



Answer(s) Given	Correct Answer(s)
Joshua has typed out this complex sentence, because the test prompted him to.	You answered this question correctly. Good job! (5.00 marks earnt)
Joshua has typed out this complex sentence, because the test prompted him to, and whomsoever disagrees, like Daniel, can go away.	You answered this question correctly. Good job! (10.00 marks earnt)
Daniel has typed out this complex sentence, because the test prompted him to, and whomsoever disagrees, like Joshua, can go away.	You answered this question correctly. Good job! (10.00 marks earnt)
Joshua has modified this complex sentence, because he is extremely, extremely lazy, and whomsoever disagrees, like Daniel, can go away.	You answered this question correctly. Good job! (10.00 marks earnt)
Daniel has modified this complex sentence, because he is extremely, extremely lazy, and whomsoever disagrees, like Joshua, can go away.	You answered this question correctly. Good job! (10.00 marks earnt)

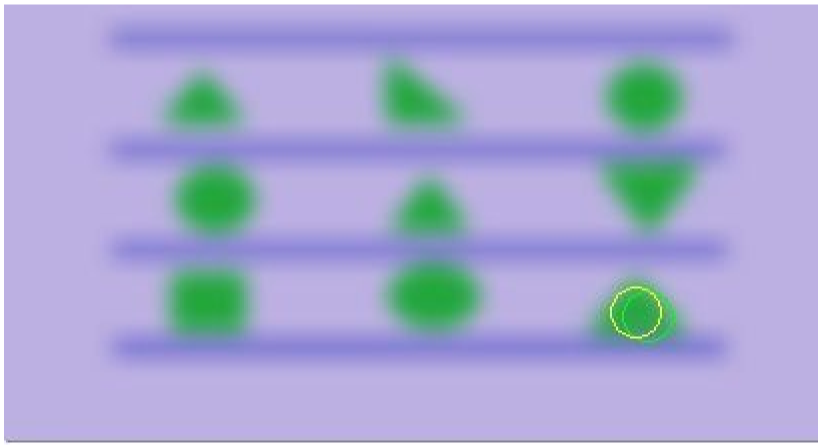
Screenshot 6.6



This. Is. An Incorrect. Sentence. Please, Help. Goodness GraCIOUS.	You answered this question incorrectly. Mistakes: Help.[6] (1 mistakes), Word count (1 mistakes), Incorrect.[3] (1 mistakes), Sentence.[4] (1 mistakes), This.[0] (1 mistakes), GraCIOUS.[8] (1 mistakes), Structure (1 mistakes), Is.[1] (1 mistakes)
This is a correct simple sentence.	You answered this question correctly. Good job! (2.00 marks earnt)
This is a correct compound sentence, because I said so.	You answered this question correctly. Good job! (5.00 marks earnt)
This is an incorrect COMPLEX sentence, because I am trying to DEMOSNTRATE. Stuff. Here. PLEASE,.	You answered this question incorrectly. Mistakes: PLEASE,.[14] (1 mistakes), DEMOSNTRATE.[11] (1 mistakes), Here.[13] (1 mistakes), Stuff.[12] (1 mistakes), Structure (1 mistakes), COMPLEX[4] (1 mistakes)
((unanswered))	No attempt was made to answer this question.

Much like Spelling Question, the results of Writing Questions are done the same way. The user answer is recorded on the left and feedback/marks earnt is recorded on the right. However, with writing questions, any errors that weren't corrected, or that arose after, are displayed too.

Image Results
Screenshot 6.7



Screenshot 6.8

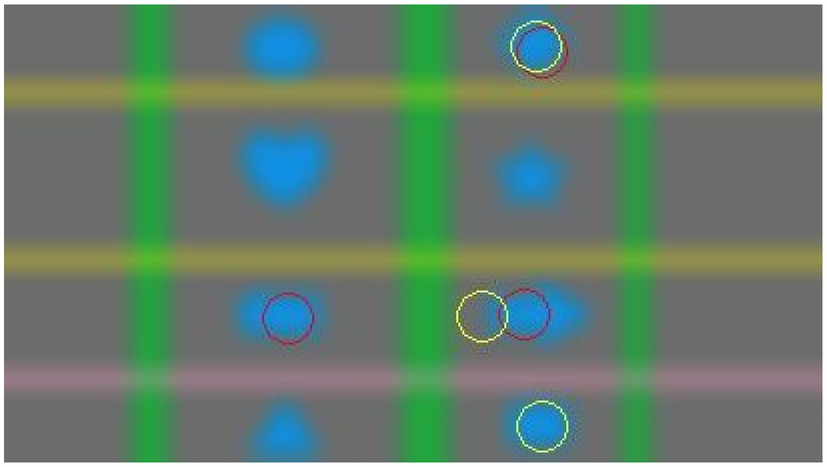
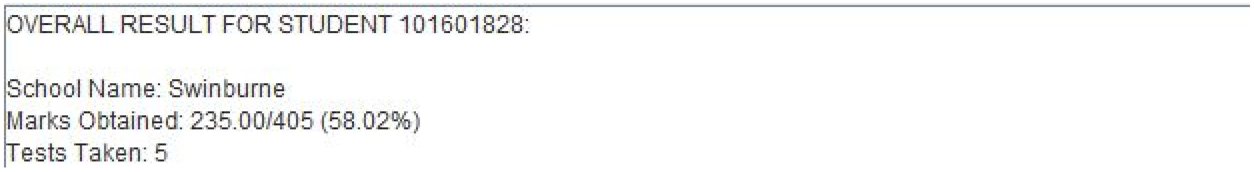


Image Question results are displayed differently, and only in one column. The image from the prompt is used, and is painted on with circles, much like during the exam. Different circles represent different things:

- Yellow Circle: Required Answer (for comparison)
- Red Circle: User Answer (if answer is incorrect)
- Green Circle: User Answer (if answer is correct)

Overall Results

Screenshot 6.9



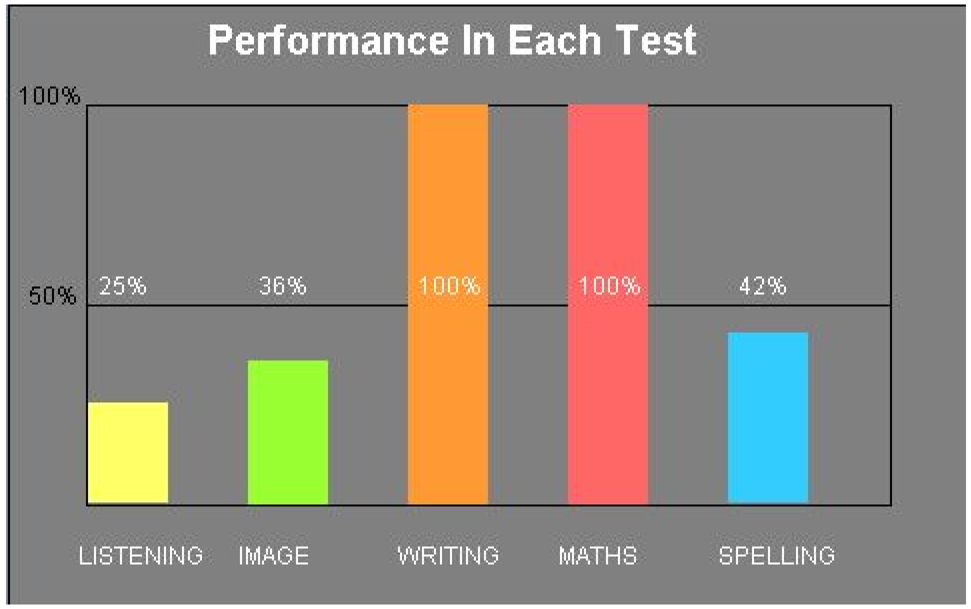
```
OVERALL RESULT FOR STUDENT 101601828:

School Name: Swinburne
Marks Obtained: 235.00/405 (58.02%)
Tests Taken: 5
```

Overall results are displayed in much the same way, using an uneditable JTextArea to display calculations. The content displayed consists of:

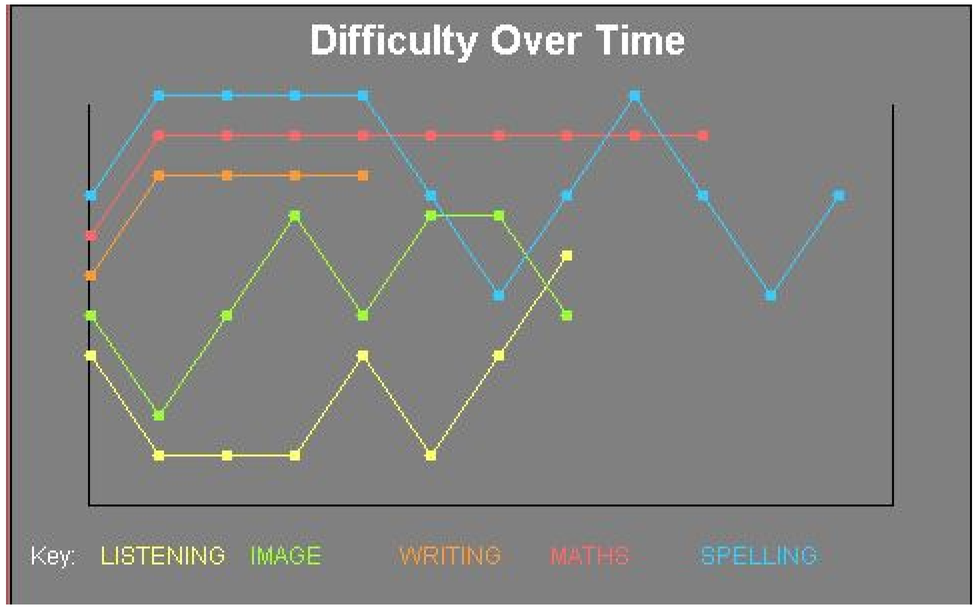
- Student’s Name

- Student’s School Name
- Marks Obtained
- Total number of possible marks that could be obtained
- Number of tests taken
- Percentage result of exam

Screenshot 6.10



Screenshot 6.11



Two statistics graphs are also displayed: a bar graph of test performance (screenshot 6.10) and a line graph of question difficulty over time in each test (screenshot 6.11). These are drawn into JPanels using hard-coded position variables, but regardless, work with any amount of tests. We opted to use Graphics2D instead of JavaFX Applications/Scenes as our program was lacking in Graphics2D usage.

That is all the features in this program. There are a few more things that are worth mentioning:

Model-View-Controller

Our program conforms to MVC, meaning that it is split up into three sections:
Models: (Exam, Test, Question, etc.)
Controllers: (ExamController, Event Handlers, etc.)
Views: (MainView, TestView, ResultsView, ExamView, etc. etc.)

Views display the information from models on the screen, for which the user’s inputs are translated into instructions for the controllers, for which the controllers modify the model data.

Help/About Dialog Menu

For those who are new to the software, it may help to consult the Help menu from the top left of the screen while it is running. It contains useful information, such as how to generate your PIN and start the exam, as well as how to answer questions and interpret your results.