CSE 360

Individual Homework 3

Darren Fernandes

1. A Listing of the Five Automated Tests

The five tests I am implementing are:
testSetQuestion()
testUpdateQuestion()
testGetAnswer()
testUpdateAnswer()
testGetAllAnswers()

2. A Description of the Five Automated Test

testSetQuestion: -

This test will run several methods within the question-and-answer database and try and ensure that the methods in their work correctly. The first method it tests is the getQuestion, which returns the entire question object based on the question number, in this case, 1. The next method is the getText method. This method returns the question as a string. Since this is a test, we have pre-populated the fields with information we already know. If the getText method as well as the getQuestion method both work correctly, it should return the text we specified i.e. "Where are the user stories for HW2 located? Are they the same ones we were working on for TP1?". If it does return this, we know that the test is working fine. However, if it does not, then the function is not working and needs to be checked. However, it can also fail when trying to connect to the database. In this case, it should enter the catch block, which will let us know that we have used this already tested code wrong and need to adjust how we call it.

testUpdateQuestion: -

This test will also run several methods within the question-and-answer database and to try and ensure that the methods in their work are correct. The first method it tests is getQuestion, which returns the entire question object based on the question number, in this case, 1. thisThise, we do not return the text, but the object itself. When the object does return, we use the setText method. This method is a setter in the Question class. When the question is set, we then try and update the database. This update should update the question in the database. We then call the method from the last test and compare the string we get to the one we tried to set it as. If it does set it correctly, The test passes. If it does not, we fail the test. However, it can also fail when trying to connect to the database. In this case, it should enter the catch block, which will let us know that we have used this already tested code wrong and need to adjust how we call it.

testGetAnswer: -

This test will run several methods within the question-and-answer database and to try and ensure that the methods in their work correctly. The first method it tests is the getAnswer, which returns the entire answer object based on the answer number, in this case, 3. The next method is the getText method. This method returns the answer as a string. Since this is a test, we have pre-populated the fields with information we already know. If the getText method as well as the getAnswer method both work correctly, it should return the text we specified i.e. "Makes sense to me.". If it does return this, we know that the test is working fine. However, if it does not, then the function is not working and needs to be checked. However, it can also fail when trying to connect to the database. This case, it should enter the catch block, which will let us know that we have used this already tested code wrong and need to adjust how we call it.

testUpdateAnswer: -

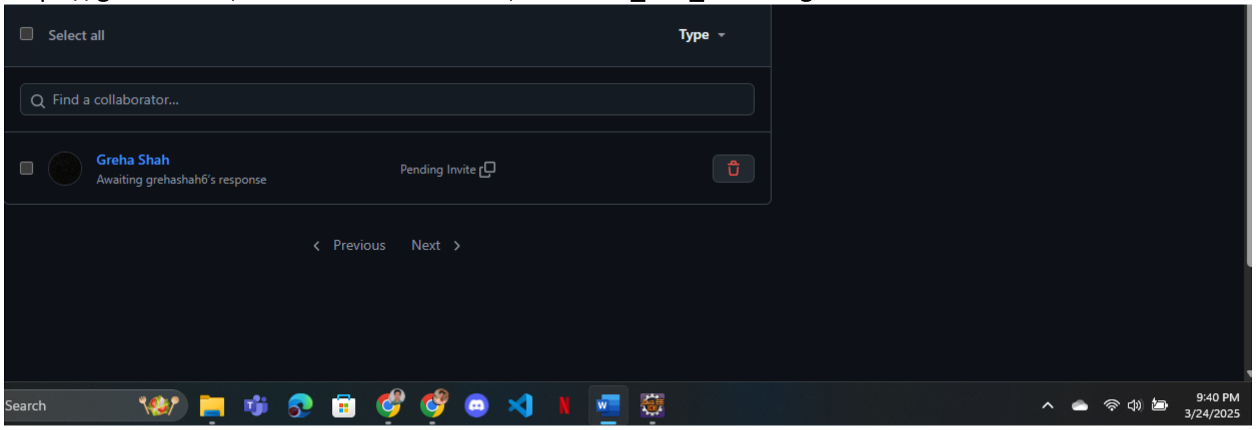
This test will also run several methods within the question-and-answer database and to try and ensure that the methods in their work are correct. The first method it tests is the getAnswer, which returns the entire answer object based on the answer number, in this case, 3. This time, we do not return the text, but the object itself. When the object does return, we use the setText method. This method is a setter in the Question class. When the answer is set, we then try and update the database. This update should update the answer in the database. We then call the method from the last test and compare the string we get to the one we tried to set it as. If it does set it correctly, The test passes. If it does not, we fail the test. However, it can also fail when trying to connect to the database. In this case, it should enter the catch block, which will let us know that we have used this already tested code wrong and need to adjust how we call it.

testGetAllAnswers: -

This is a test to test our ability to get all the answers from the database. We know how we stored them along with the number of answers, what the answers are, and their related id's. With this information, we test a few things that we expect and see if our list has the required things. The first thing we do is of course connect to the database, which should be done without issues, but is placed in a try catch block just in case. We then get all the answers from the database and store them in a list for a particular question, 1 in this case. We then compare the number of answers in the list, 2, versus the number found. We also check the text of the field and check if that also works out to the same text expected, "Oh well another test happened", in this case. If both are fine, then the test passes, and we exit out successfully

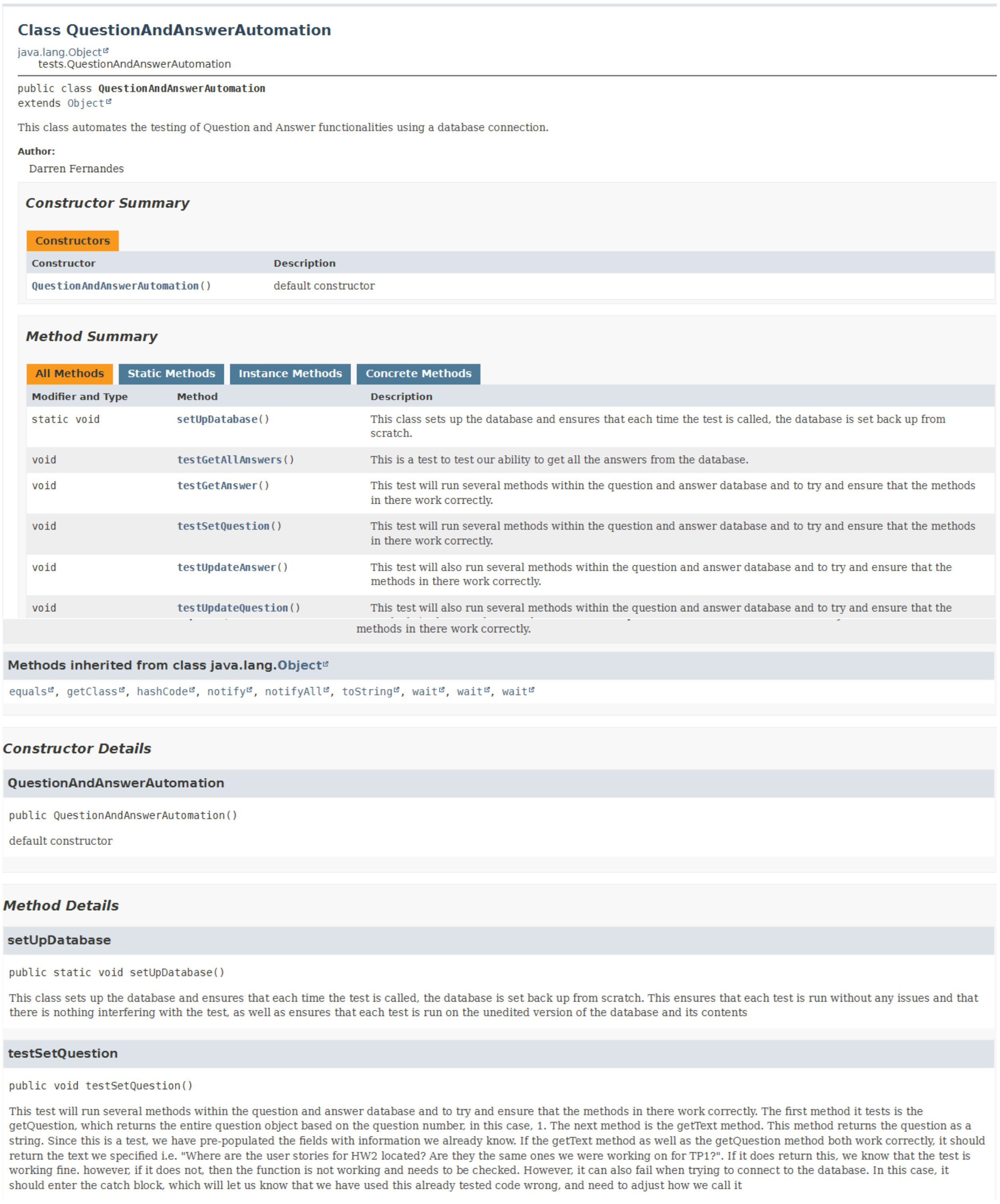
3. The Link to the HW3 Code in Your Personal Repository

I have invited the grader (git username grehashah6). Please accept this and view this document, the README, as well as my code, including the Javadoc and the screencast. The Javadoc can be found in the folder HW3->src-> doc. This folder contains all the applications generated when creating the Javadoc.
https://github.com/DarrenFernandes0402/Dkferna1_HW_CSE360.git


Select all
Type
Find a collaborator…
Greha Shah
Awaiting grehashah6's response
Pending Invite

The following is my Javadoc:

Individual Homework 3



Class QuestionAndAnswerAutomation

java.lang.Object
    tests.QuestionAndAnswerAutomation

```
public class QuestionAndAnswerAutomation
extends Object
```

This class automates the testing of Question and Answer functionalities using a database connection.

Author:
Darren Fernandes

Constructor Summary

Constructors

Constructor	Description
`QuestionAndAnswerAutomation()`	default constructor

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
`static void`	`setUpDatabase()`	This class sets up the database and ensures that each time the test is called, the database is set back up from scratch.
`void`	`testGetAllAnswers()`	This is a test to test our ability to get all the answers from the database.
`void`	`testGetAnswer()`	This test will run several methods within the question and answer database and to try and ensure that the methods in there work correctly.
`void`	`testSetQuestion()`	This test will run several methods within the question and answer database and to try and ensure that the methods in there work correctly.
`void`	`testUpdateAnswer()`	This test will also run several methods within the question and answer database and to try and ensure that the methods in there work correctly.
`void`	`testUpdateQuestion()`	This test will also run several methods within the question and answer database and to try and ensure that the methods in there work correctly.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

QuestionAndAnswerAutomation

```
public QuestionAndAnswerAutomation()
```

default constructor

Method Details

setUpDatabase

```
public static void setUpDatabase()
```

This class sets up the database and ensures that each time the test is called, the database is set back up from scratch. This ensures that each test is run without any issues and that there is nothing interfering with the test, as well as ensures that each test is run on the unedited version of the database and its contents

testSetQuestion

```
public void testSetQuestion()
```

This test will run several methods within the question and answer database and to try and ensure that the methods in there work correctly. The first method it tests is the getQuestion, which returns the entire question object based on the question number, in this case, 1. The next method is the getText method. This method returns the question as a string. Since this is a test, we have pre-populated the fields with information we already know. If the getText method as well as the getQuestion method both work correctly, it should return the text we specified i.e. "Where are the user stories for HW2 located? Are they the same ones we were working on for TP1?". If it does return this, we know that the test is working fine. however, if it does not, then the function is not working and needs to be checked. However, it can also fail when trying to connect to the database. In this case, it should enter the catch block, which will let us know that we have used this already tested code wrong, and need to adjust how we call it

testUpdateQuestion

```
public void testUpdateQuestion()
```

This test will also run several methods within the question and answer database and to try and ensure that the methods in there work correctly. The first method it tests is the getQuestion, which returns the entire question object based on the question number, in this case, 1. this time, we do not return the text, but the object itself When the object does return, we use the setText method. This method is a setter in the Question class. When the question is set, we then try and update the database. This update should update the question in the database. We then call the method from the last test, and compare the string we get to the one we tried to set it as. If it does set it correctly, The test passes. If it does not, we fail the test. However, it can also fail when trying to connect to the database. In this case, it should enter the catch block, which will let us know that we have used this already tested code wrong, and need to adjust how we call it.

testGetAnswer

```
public void testGetAnswer()
```

This test will run several methods within the question and answer database and to try and ensure that the methods in there work correctly. The first method it tests is the getAnswer, which returns the entire answer object based on the answer number, in this case, 3. The next method is the getText method. This method returns the answer as a string. Since this is a test, we have pre-populated the fields with information we already know. If the getText method as well as the getAnswer method both work correctly, it should return the text we specified i.e. "Makes sense to me.". If it does return this, we know that the test is working fine. however, if it does not, then the function is not working and needs to be checked. * However, it can also fail when trying to connect to the database. In this case, it should enter the catch block, which will let us know that we have used this already tested code wrong, and need to adjust how we call it.

testUpdateAnswer

```
public void testUpdateAnswer()
```

This test will also run several methods within the question and answer database and to try and ensure that the methods in there work correctly. The first method it tests is the getAnswer, which returns the entire answer object based on the answer number, in this case, 3. this time, we do not return the text, but the object itself When the object does return, we use the setText method. This method is a setter in the Question class. When the answer is set, we then try and update the database. This update should update the answer in the database. We then call the method from the last test, and compare the string we get to the one we tried to set it as. If it does set it correctly, The test passes. If it does not, we fail the test. However, it can also fail when trying to connect to the database. In this case, it should enter the catch block, which will let us know that we have used this already tested code wrong, and need to adjust how we call it.

testGetAllAnswers

```
public void testGetAllAnswers()
```

This is a test to test our ability to get all the answers from the database. We know how we stored them along with the number of answers, what the answers are, and their related id's. With this information, we test a few things in there that we expect, and see if our list has the required things. The first thing we do is ofcourse connect to the database, which should be done without issues, but is placed in a try catch block just incase. We then get all the answers from the database and store it in a list for a particular question, 1 in this case. We then compare the number of answers in the list, 2, versus the number found. We also check the text of the field and check if that also works out to the same text expected, "Oh well another test happened", in this case. If both of these are fine, then the test passes and we exit out successfully

4. The Link to the Source for Your Javadoc Inspiration

My Javadoc took inspiration from Oracle’s how to write Doc comments for Javadoc tool:

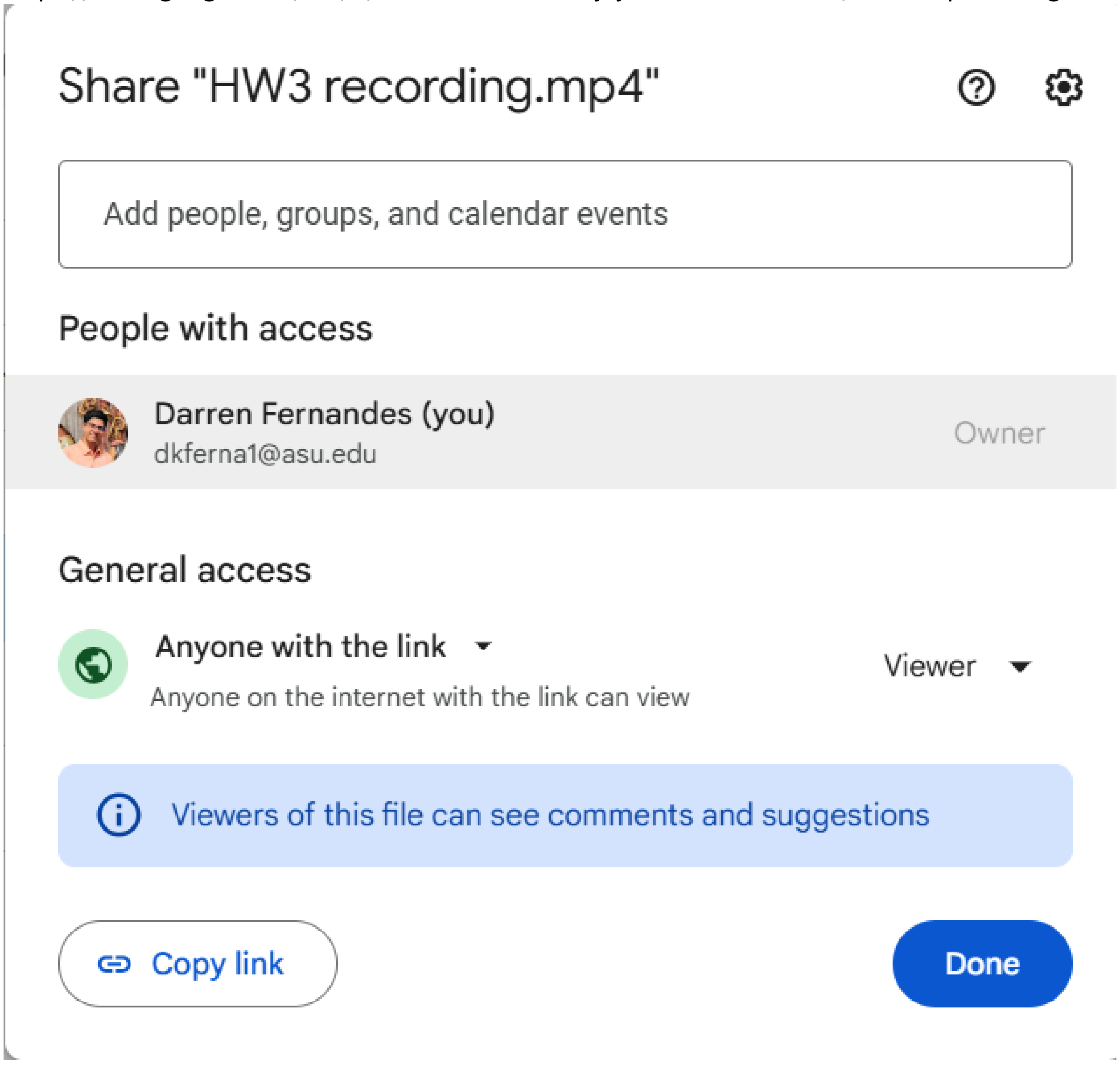
https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html

5. The Link to Your Screencast

https://github.com/DarrenFernandes0402/Dkferna1_HW_CSE360/blob/main/HW3%20recording.mp4

This is the link to the screen recording. I also have it in my drive and it can also be accessed there through the following link:

https://drive.google.com/file/d/15WFk4IdHtJ5wfwjnjb2vMk8Ph6o2vUkl/view?usp=sharing



Share "HW3 recording.mp4"

Add people, groups, and calendar events

People with access

Darren Fernandes (you)
dkferna1@asu.edu
Owner

General access

Anyone with the link
Anyone on the internet with the link can view
Viewer

Viewers of this file can see comments and suggestions