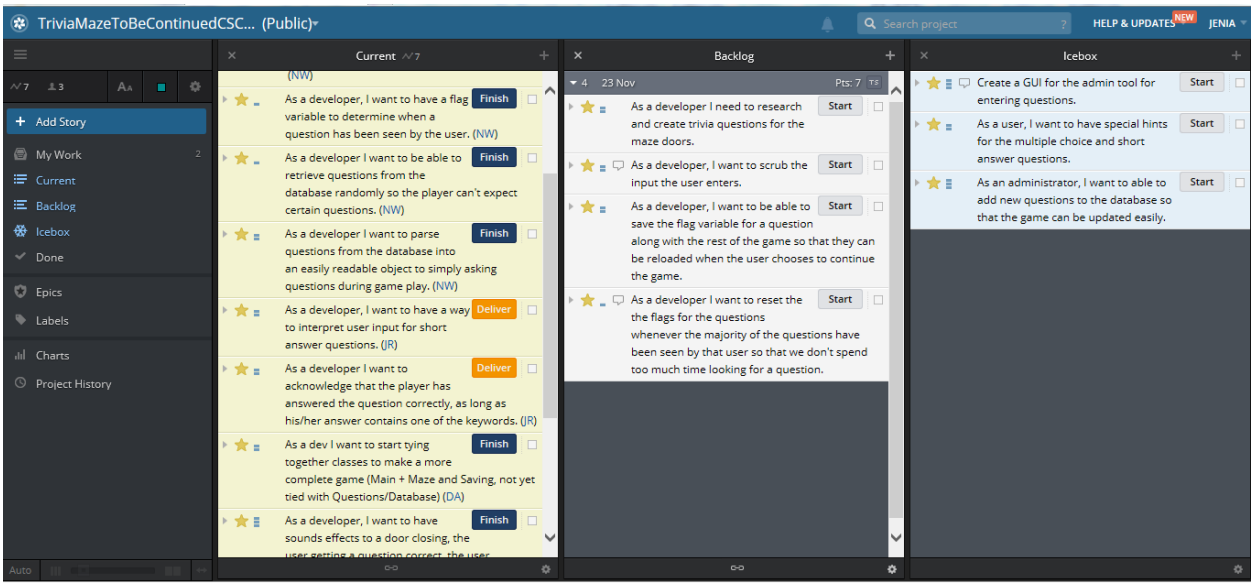
CSCD 350 Software Engineering

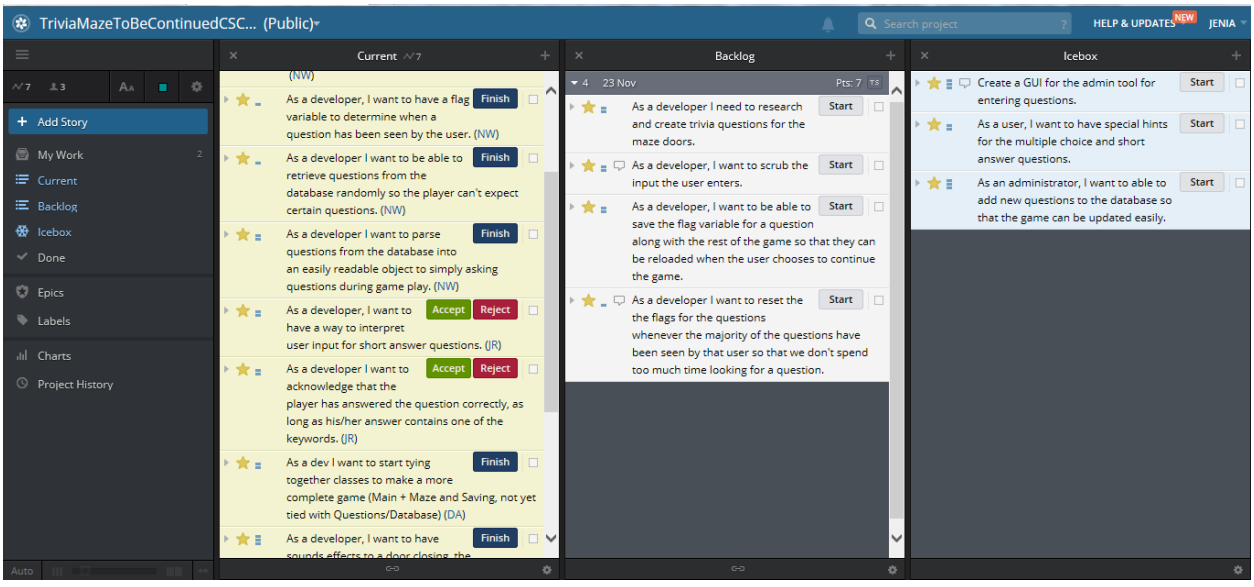
End of 3rd Iteration

Jenia's work for Iteration 3:

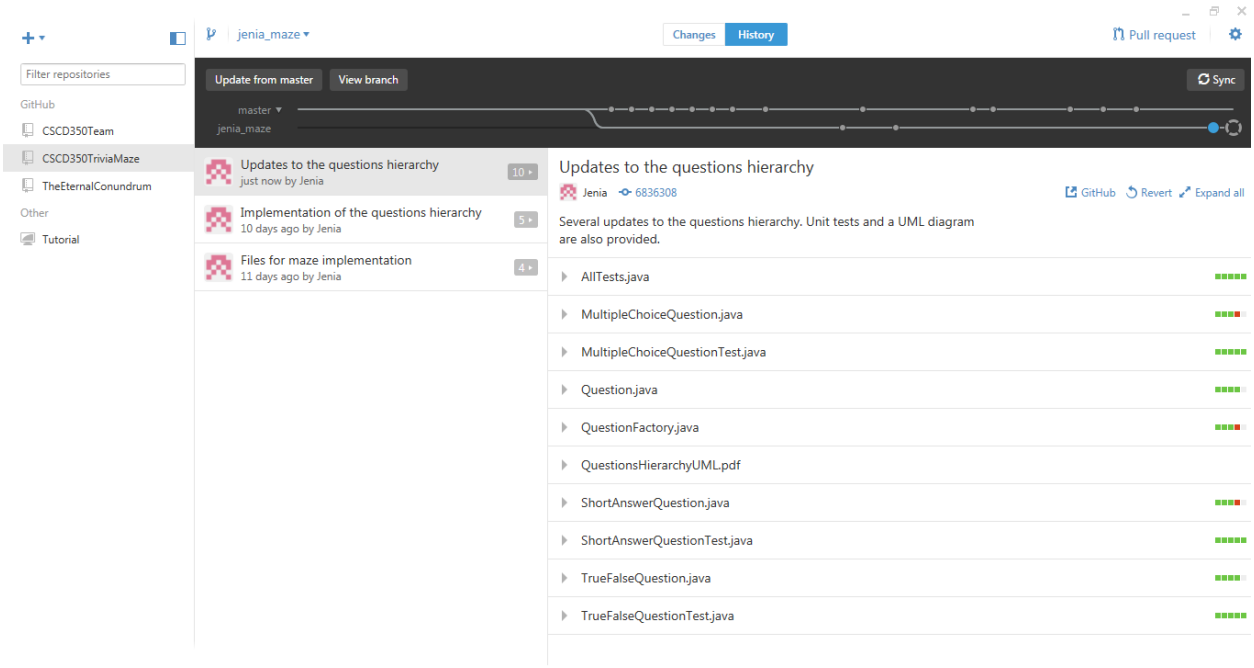
Delivering user stories:

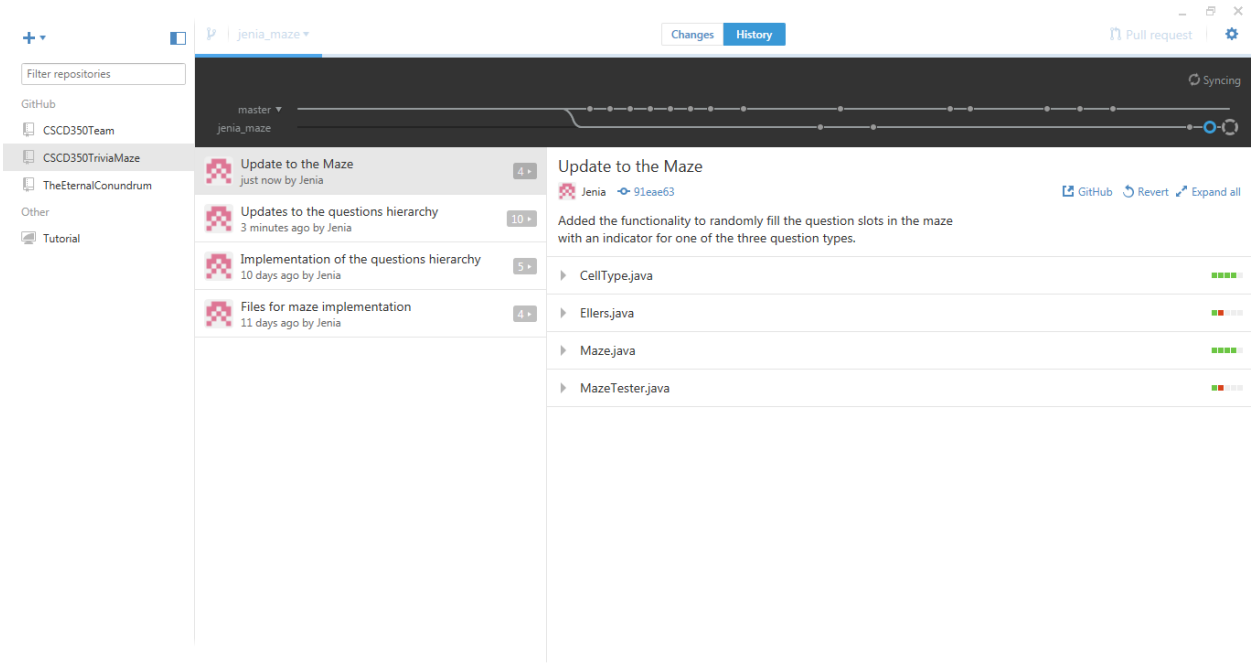


Stories accepted:

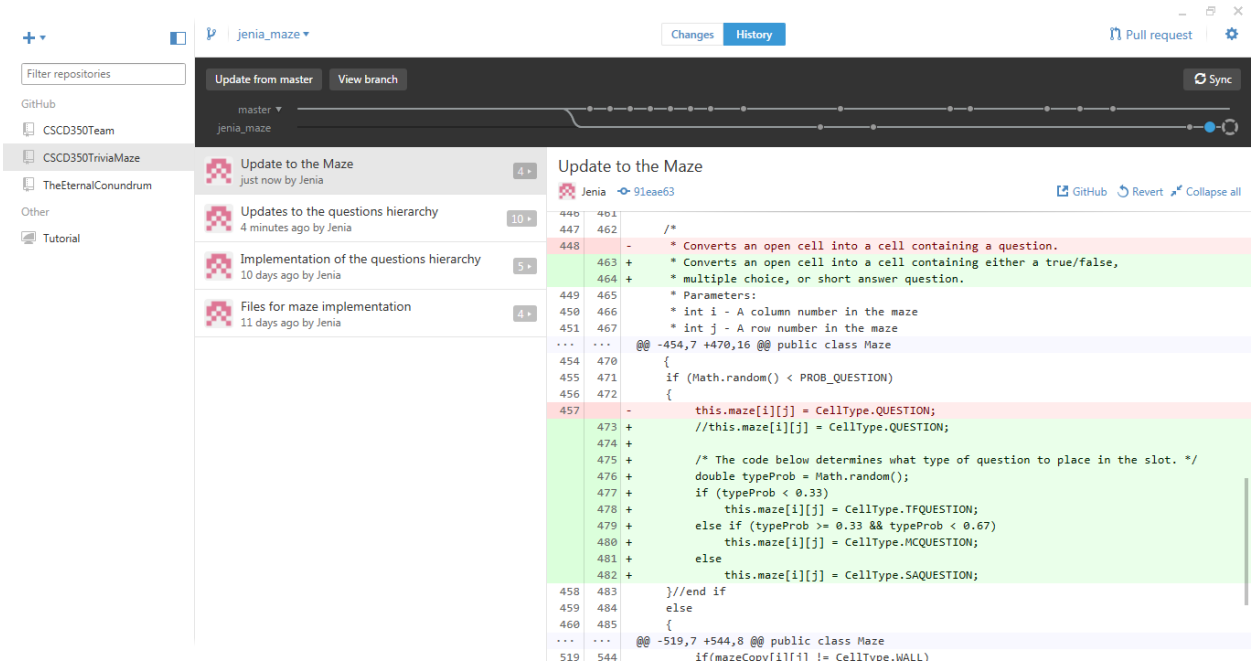


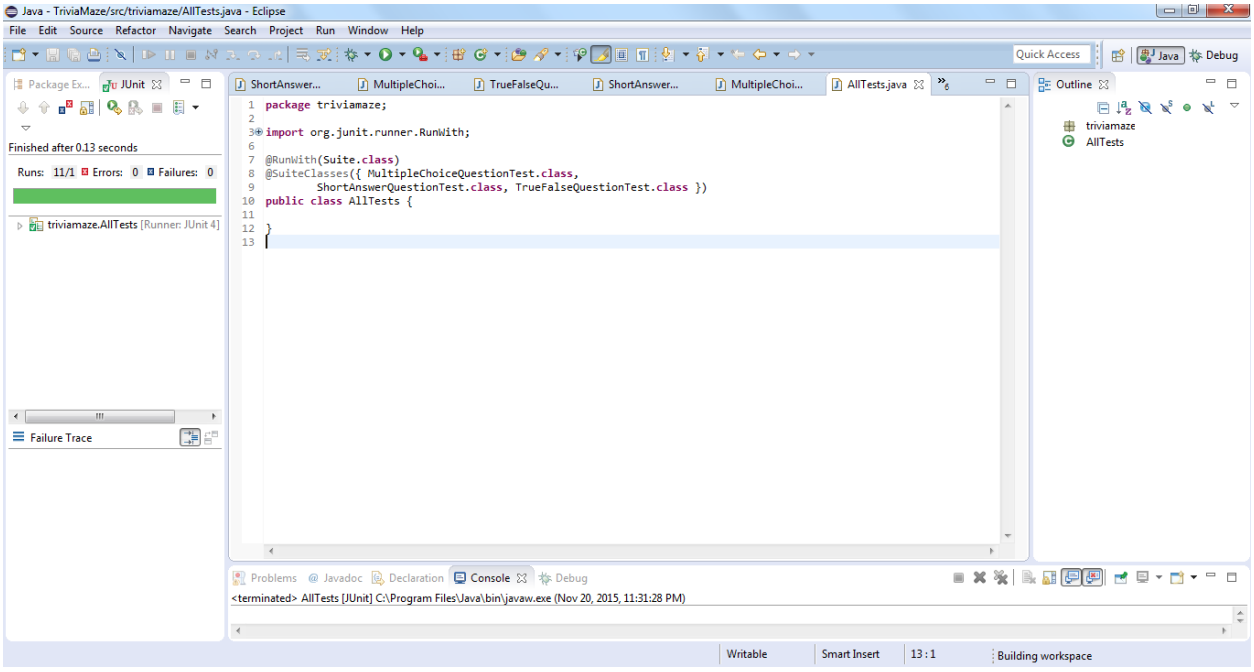
Committing the changes to GitHub:





Code Snippets:





ShortAnswer... | MultipleChoi... | TrueFalseQu... | ShortAnswer... | MultipleChoi... | AllTests.java

```
98      @Override
99      public boolean checkCorrectAnswer(String input)
100     {
101         char ch = input.toUpperCase().charAt(0);
102         String answer = getAnswerAtIndex(ch);
103         return answer.equalsIgnoreCase(this.getCorrectAnswer());
104     }//end checkCorrectAnswer
105
106
107     /*
108      * This overrides the hook for the Question superclass.
109      * Multiple choice questions have the special functionality to have
110      * the answer choices randomly shuffled.
111      */
112
113     @Override
114     public void specialFunction()
115     {
116         String [] choices = this.getChoices();
117         for (int i = 0; i < choices.length; i ++)
118         {
119             int j = (int)(Math.random() * choices.length);
120             String temp = choices[i];
121             choices[i] = choices[j];
122             choices[j] = temp;
123         }//end for
124     }//end specialFunction
125
126
127     /*
128      * Maybe eliminate one or two of the wrong answer choices.
129      * Notify the user of the wrong choice(s).
```

ShortAnswer... | MultipleChoi... | TrueFalseQu... | ShortAnswer... | MultipleChoi... | AllTests.java

```
54      {
55          assertEquals("Olympia", test.getCorrectAnswer());
56      }//end testGetCorrectAnswer
57
58
59      @Test
60      public void testGetChoices()
61      {
62          assertEquals("Seattle", test.getChoices()[0]);
63          assertEquals("Olympia", test.getChoices()[1]);
64          assertEquals("Spokane", test.getChoices()[2]);
65          assertEquals("Washington DC", test.getChoices()[3]);
66      }//end testGetChoices
67
68
69      @Test
70      public void testCheckCorrectAnswer()
71      {
72          /* These are acceptable answers */
73          String a1 = "B";
74          String a2 = "b";
75
76          assertEquals(true, test.checkCorrectAnswer(a1));
77          assertEquals(true, test.checkCorrectAnswer(a2));
78
79          /* These are unacceptable answers */
80          String a3 = "a";
81          String a4 = "C";
82
83          assertEquals(false, test.checkCorrectAnswer(a3));
84          assertEquals(false, test.checkCorrectAnswer(a4));
85      }//end testCheckCorrectAnswer
86
```

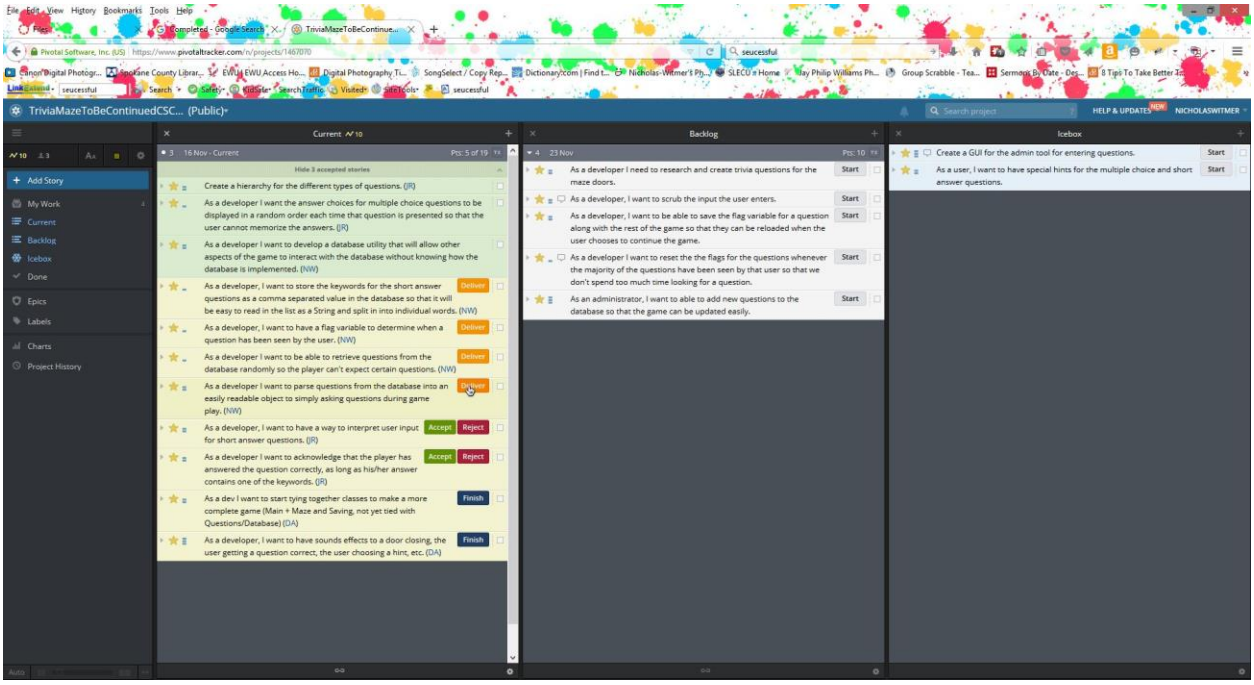
ShortAnswer... | MultipleChoi... | TrueFalseQu... | ShortAnswer... | MultipleChoi... | AllTests.java

```
60      public void testGetKeywords()
61      {
62          assertEquals("red", test.getChoices()[0]);
63          assertEquals("white", test.getChoices()[1]);
64          assertEquals("blue", test.getChoices()[2]);
65      }//end testGetKeywords
66
67
68      @Test
69      public void testCheckCorrectAnswer()
70      {
71          /* These are acceptable answers */
72          String a1 = "red";
73          String a2 = "WHITE";
74          String a3 = "A color is blue.";
75          String a4 = "Red is on the French flag";
76
77          assertEquals(true, test.checkCorrectAnswer(a1));
78          assertEquals(true, test.checkCorrectAnswer(a2));
79          assertEquals(true, test.checkCorrectAnswer(a3));
80          assertEquals(true, test.checkCorrectAnswer(a4));
81
82          /* These are unacceptable answers */
83          String a5 = "RRED is there!";
84          String a6 = "b lue";
85          String a7 = "Whiite";
86          String a8 = " w@hite";
87
88          assertEquals(false, test.checkCorrectAnswer(a5));
89          assertEquals(false, test.checkCorrectAnswer(a6));
90          assertEquals(false, test.checkCorrectAnswer(a7));
91          assertEquals(false, test.checkCorrectAnswer(a8));
92      }//end testCheckCorrectAnswer
93
```

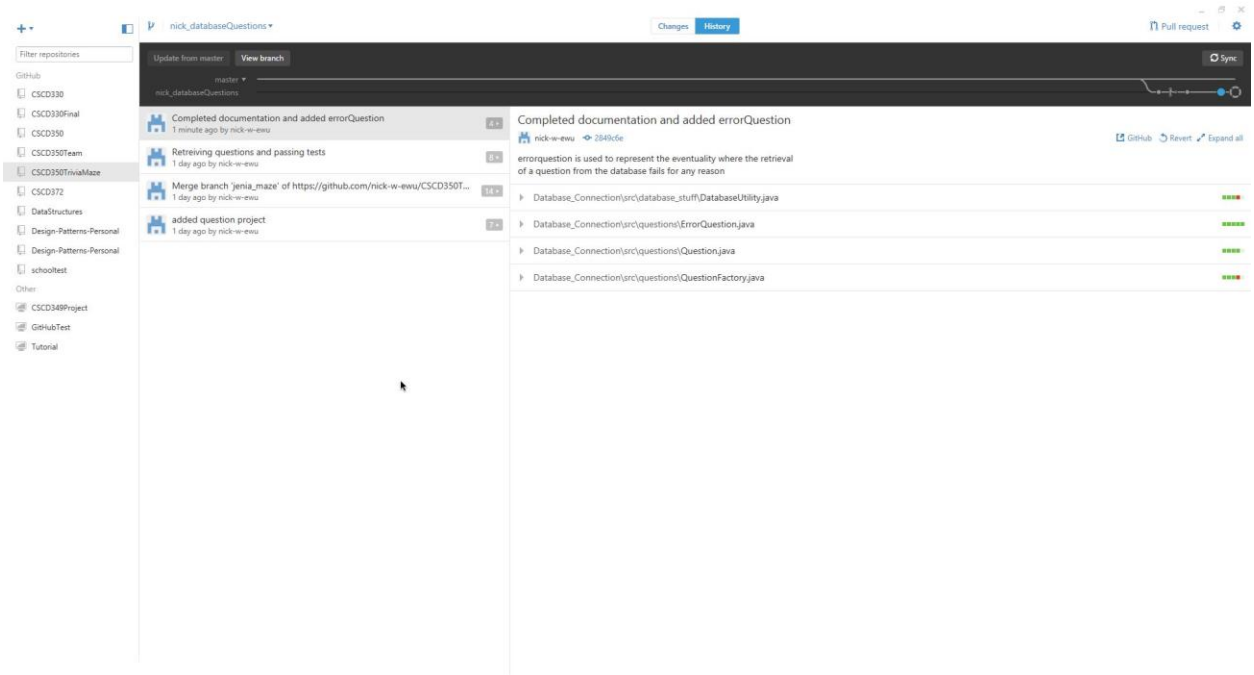
MultipleChoi... | TrueFalseQu... | ShortAnswer... | MultipleChoi... | AllTests.java | TrueFalseQu...

```
35          test.setQuestion("The length of a year on earth is 365 days.");
36          test.setCorrectAnswer("F");
37      }//end setUp
38
39
40      @After
41      public void tearDown() throws Exception {}
42
43
44      @Test
45      public void testGetQuestion()
46      {
47          assertEquals("The length of a year on earth is 365 days.", test.getQuestion());
48      }//end testGetQuestion
49
50
51      @Test
52      public void testGetCorrectAnswer()
53      {
54          assertEquals("F", test.getCorrectAnswer());
55      }//end testGetCorrectAnswer
56
57
58      @Test
59      public void testCheckCorrectAnswer()
60      {
61          assertEquals(true, test.checkCorrectAnswer("f"));
62          assertEquals(true, test.checkCorrectAnswer("F"));
63          assertEquals(false, test.checkCorrectAnswer("false"));
64          assertEquals(false, test.checkCorrectAnswer("T"));
65      }//end testCheckCorrectAnswer
66
67  }//end TrueFalseQuestionTest
68
```

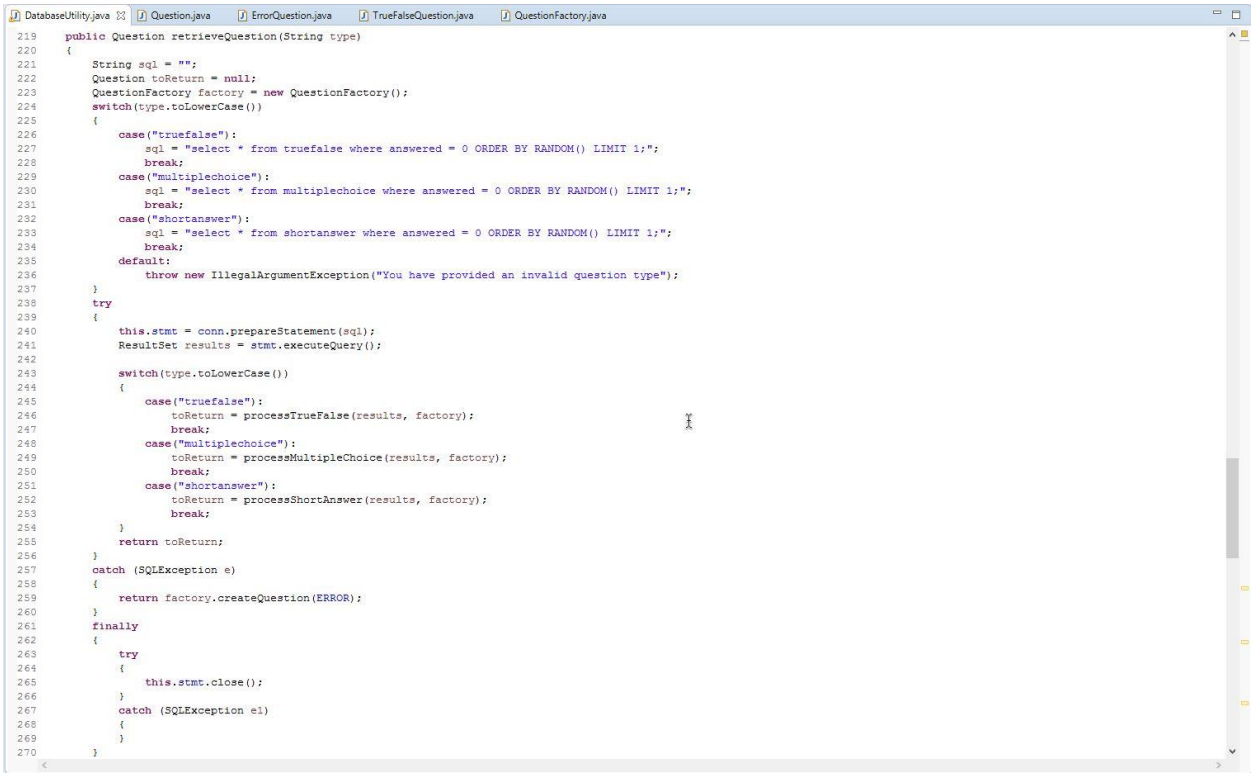
Pivotal Tracker Nick's stories finished



GitHub branch, created a new one to resolve some issues caused by the power outage and trying to continue working on the project



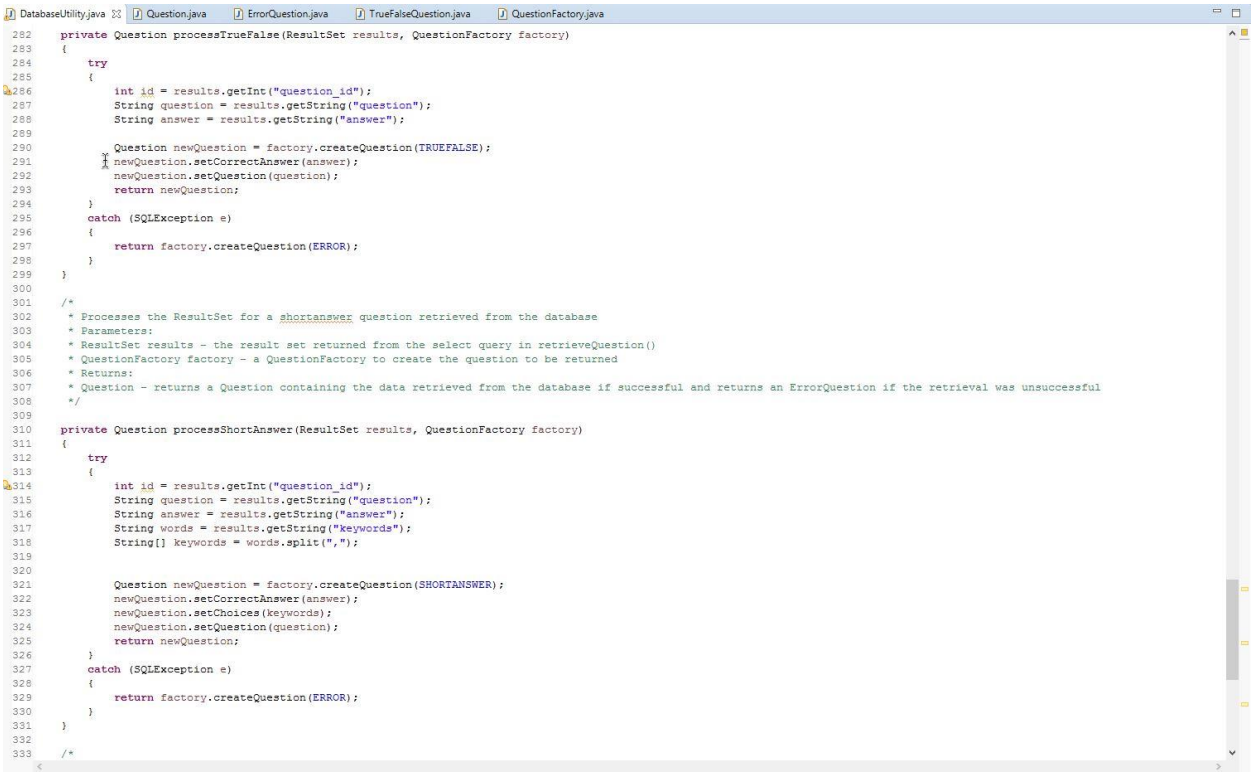
Retrieval of questions from the database in the DatabaseUtility



```
public Question retrieveQuestion(String type)
{
    String sql = "";
    Question toReturn = null;
    QuestionFactory factory = new QuestionFactory();
    switch(type.toLowerCase())
    {
        case("truefalse"):
            sql = "select * from truefalse where answered = 0 ORDER BY RANDOM() LIMIT 1;";
            break;
        case("multiplechoice"):
            sql = "select * from multiplechoice where answered = 0 ORDER BY RANDOM() LIMIT 1;";
            break;
        case("shortanswer"):
            sql = "select * from shortanswer where answered = 0 ORDER BY RANDOM() LIMIT 1;";
            break;
        default:
            throw new IllegalArgumentException("You have provided an invalid question type");
    }
    try
    {
        this.stmt = conn.prepareStatement(sql);
        ResultSet results = stmt.executeQuery();

        switch(type.toLowerCase())
        {
            case("truefalse"):
                toReturn = processTrueFalse(results, factory);
                break;
            case("multiplechoice"):
                toReturn = processMultipleChoice(results, factory);
                break;
            case("shortanswer"):
                toReturn = processShortAnswer(results, factory);
                break;
        }
        return toReturn;
    }
    catch (SQLException e)
    {
        return factory.createQuestion(ERROR);
    }
    finally
    {
        try
        {
            this.stmt.close();
        }
        catch (SQLException e1)
        {
        }
    }
```

Processing of ResultSets retrieved from the database in above function



```
private Question processTrueFalse(ResultSet results, QuestionFactory factory)
{
    try
    {
        int id = results.getInt("question_id");
        String question = results.getString("question");
        String answer = results.getString("answer");

        Question newQuestion = factory.createQuestion(TRUEFALSE);
        newQuestion.setCorrectAnswer(answer);
        newQuestion.setQuestion(question);
        return newQuestion;
    }
    catch (SQLException e)
    {
        return factory.createQuestion(ERROR);
    }
}

/*
 * Processes the ResultSet for a shortanswer question retrieved from the database
 * Parameters:
 * ResultSet results - the result set returned from the select query in retrieveQuestion()
 * QuestionFactory factory - a QuestionFactory to create the question to be returned
 * Returns:
 * Question - returns a Question containing the data retrieved from the database if successful and returns an ErrorQuestion if the retrieval was unsuccessful
 */

private Question processShortAnswer(ResultSet results, QuestionFactory factory)
{
    try
    {
        int id = results.getInt("question_id");
        String question = results.getString("question");
        String answer = results.getString("answer");
        String words = results.getString("keywords");
        String[] keywords = words.split(",");


        Question newQuestion = factory.createQuestion(SHORTANSWER);
        newQuestion.setCorrectAnswer(answer);
        newQuestion.setChoices(keywords);
        newQuestion.setQuestion(question);
        return newQuestion;
    }
    catch (SQLException e)
    {
        return factory.createQuestion(ERROR);
    }
}

/*
```

Testing for question retrieval

DatabaseTests.java

```
23    @Test
24    public void testInsertTrueFalse()
25    {
26        assertTrue(test.insertQuestion("Will it rain today", "t"));
27        assertFalse(test.insertQuestion(null, "t"));
28    }
29
30    @Test
31    public void testInsertShortAnswer()
32    {
33        assertTrue(test.insertQuestion("Will it rain today", "Maybe, it could", "maybe,probably"));
34        assertFalse(test.insertQuestion("What is my name", null, null));
35    }
36
37    @Test
38    public void testInsertMultipleChoice()
39    {
40        assertTrue(test.insertQuestion("How many children are in Mary Poppins", "2", "55", "2", "7", "25"));
41        assertFalse(test.insertQuestion("What is my name", "3", "Nick", "Fish", null, "Nothing"));
42    }
43
44    @Test
45    public void testInsertHint()
46    {
47        assertTrue(test.insertHint(id++, "shortanswer", "its possible"));
48        assertFalse(test.insertHint(8, null, null));
49    }
50
51    //Will fail on first run if the database is empty
52    @Test
53    public void testretrieveQuestions()
54    {
55        Question q1 = test.retrieveQuestion("shortanswer");
56        Question q2 = test.retrieveQuestion("truefalse");
57        Question q3 = test.retrieveQuestion("multiplechoice");
58        String[] choices = {"maybe", "probably"};
59        assertArrayEquals(choices, q1.getChoices());
60        assertEquals("Will it rain today", q1.getQuestion());
61        assertEquals("Maybe, it could", q1.getCorrectAnswer());
62
63        assertEquals("t", q2.getCorrectAnswer());
64        assertEquals("Will it rain today", q2.getQuestion());
65
66        String[] options = {"55", "2", "7", "25"};
67        assertEquals("How many children are in Mary Poppins", q3.getQuestion());
68        assertEquals("2", q3.getCorrectAnswer());
69        assertArrayEquals(options, q3.getChoices());
70    }
71
72
73}
74
```

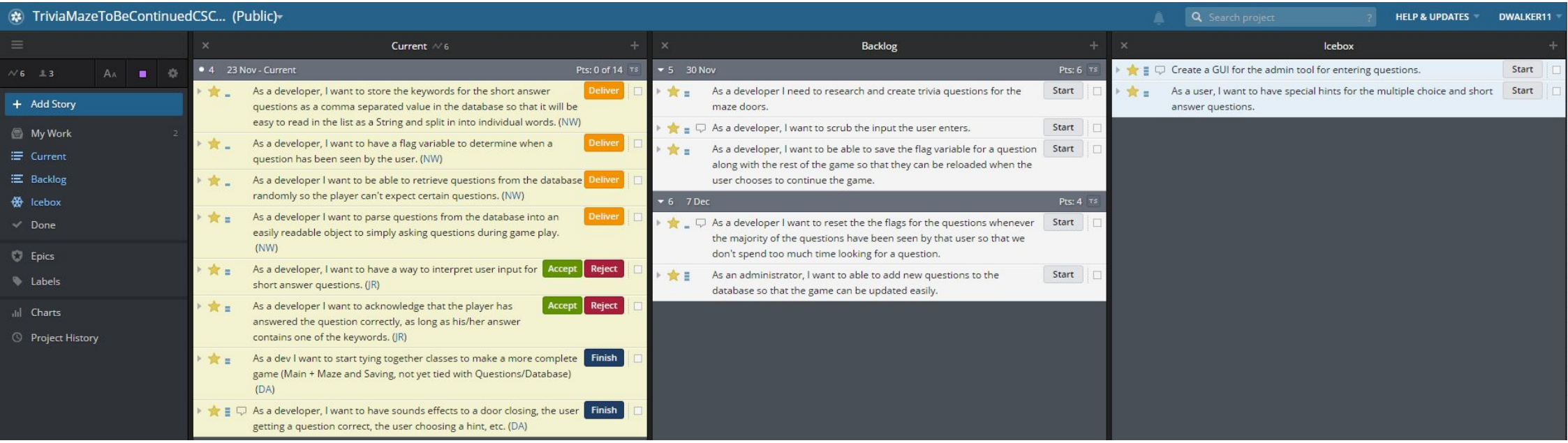
15 commits | 5 branches | 0 releases | 3 contributors

Branch: **davidBranch** **CSCD350TriviaMaze / +**

This branch is 7 commits ahead, 9 commits behind master. Pull request Compare

David2Walker Merging Class together, Saving + Main + Maze + Players (No Question/DB) Latest commit `7db74d4` 17 hours ago

.idea	Merging Class together, Saving + Main + Maze + Players (No Question/DB)	17 hours ago
out/production/CSCD350Trivi...	Added Maze Files	21 hours ago
CSCD350TriviaMaze.iml	merged to work with intellij (hopefully)	8 days ago
CellType.java	Added Maze Files	21 hours ago
Ellers.java	Added Maze Files	21 hours ago
Item.java	Added Trivial Util + save feature	8 days ago
Maze.java	Added Maze Files	21 hours ago
MazeTester.java	Added Maze Files	21 hours ago
Player.java	Merging Class together, Saving + Main + Maze + Players (No Question/DB)	17 hours ago
README.md	Revert "Revert "Initial commit""	17 days ago
TriviaMain.java	Merging Class together, Saving + Main + Maze + Players (No Question/DB)	17 hours ago
saved.ser	merged to work with intellij (hopefully)	8 days ago
triviaUtil.java	Merging Class together, Saving + Main + Maze + Players (No Question/DB)	17 hours ago


TriviaMazeToBeContinuedCSC... (Public)
Search project
HELP & UPDATES
DWALKER11
6
3
Add Story
My Work
2
Current
Backlog
Icebox
Done
Epics
Labels
Charts
Project History
Current 6
4 23 Nov - Current
Pts: 0 of 14
As a developer, I want to store the keywords for the short answer questions as a comma separated value in the database so that it will be easy to read in the list as a String and split in into individual words. (NW)
Deliver
As a developer, I want to have a flag variable to determine when a question has been seen by the user. (NW)
Deliver
As a developer I want to be able to retrieve questions from the database randomly so the player can't expect certain questions. (NW)
Deliver
As a developer I want to parse questions from the database into an easily readable object to simply asking questions during game play. (NW)
Deliver
As a developer, I want to have a way to interpret user input for short answer questions. (JR)
Accept
Reject
As a developer I want to acknowledge that the player has answered the question correctly, as long as his/her answer contains one of the keywords. (JR)
Accept
Reject
As a dev I want to start tying together classes to make a more complete game (Main + Maze and Saving, not yet tied with Questions/Database) (DA)
Finish
As a developer, I want to have sounds effects to a door closing, the user getting a question correct, the user choosing a hint, etc. (DA)
Finish
Backlog
5 30 Nov
Pts: 6
As a developer I need to research and create trivia questions for the maze doors.
Start
As a developer, I want to scrub the input the user enters.
Start
As a developer, I want to be able to save the flag variable for a question along with the rest of the game so that they can be reloaded when the user chooses to continue the game.
Start
6 7 Dec
Pts: 4
As a developer I want to reset the the flags for the questions whenever the majority of the questions have been seen by that user so that we don't spend too much time looking for a question.
Start
As an administrator, I want to able to add new questions to the database so that the game can be updated easily.
Start
Icebox
Create a GUI for the admin tool for entering questions.
Start
As a user, I want to have special hints for the multiple choice and short answer questions.
Start