HW 4 Details

Topics: files, lists, and objects

In lectures, we explored an application involving a massive collection of UFO sightings. The UFOsighting class stored information about a single sighting and provided methods for accessing that information. The UFOlookup class read in the UFO sighting data from a file, constructed an ArrayList of UFOsighting objects, and then provided methods for selectively looking up sightings based on location or date. For this assignment, you will develop similar classes that store and process information about accessibility tests.

In order to evaluate systems or tools, it is extremely common to create a test suite that can be used to evaluate a single system or compare multiple systems. For this assignment, we will be looking at the results of multiple tools that were run on a test suite created by the United Kingdom's Government Digital Services to test accessibility checkers. An accessible page is one that is usable by people with disabilities, many of whom will be accessing the web with assistive technologies such as [screen readers (Links to an external site.)Links to an external site.](https://www.youtube.com/watch?v=92pM6hJG6Wo&t=123s) or [dictation software (Links to an external site.)Links to an external site.](https://www.youtube.com/watch?v=kJKQmTumFP0&t=17s). The [test suite (Links to an external site.)Links to an external site.](https://alphagov.github.io/accessibility-tool-audit/test-cases.html) has a series of tests, each with a single introduced error. The tests are grouped within categories of related tests.

These tests have been run on 13 different accessibility checkers and the [full results (Links to an external site.)Links to an external site.](https://alphagov.github.io/accessibility-tool-audit/results.html) have been made publicly available. For this assignment, I have created a file, [a11yCheckersResults.txt](https://blueline.instructure.com/courses/1146932/files/folder/Homeworks/HW4?preview=58674226) that contains the results of the tests for 4 of the 13 checkers. The file is organized such that each line contains the results for one test organized as such:

[category] [google result] [wave result] [sortsite result] [aslint result] [test description]

The category and test results for each checker are all a single "word". The test description is of variable length.

The test results will be one of six "words":

* error = the issue was properly found
* error\_paid = the issue was properly found in the paid version
* warning = the tool only gave a warning
* manual = the tool required the user to check
* identified = The tool noticed the issue, but didn't give a warning
* notfound = The tool did not identify the issue

This homework will have you write code to read in the test results list and then use them to display different information about the tests.

Part 1: AccessibilityTest

Similar to the UFOSighting class, you are to design and implement a class named AccessibilityTest, which stores the the category and description of the test and the results of the test for the four checkers. The constructor for the class should take all 6 pieces of information as parameters and store those values in fields. It should have accessor methods for each value, e.g., getCategory, getDescription, getGoogleResult, etc. It should also have a method toString method which presents a readable format of the results of the test.

Part 2: AccessibilityResults

The file a11yCheckersResults.txt has been provided for you, which contains the information about tests used by the UK Government Digital Services to evaluate accessibility checkers.Each line in the file contains the information about a single test organized as:

[category] [google result] [wave result] [sortsite result] [aslint result] [test description]

For instance, the line for the test that is used to see if a person can visually determine which element has keyboard focus looks like this:

Keyboard notfound notfound error error Keyboard focus is not indicated visually

This line indicates that the test is in the Keyboard category of tests, the error was not detected by the Google or WAVE result, but was detected by SortSite and ASLint.

Similar to the UFOlookup class, you are to design and implement a class named AccessibilityResults that reads in the test information from a file of this format, stores them in an ArrayList of AccessibilityTest objects, and provides methods for accessing that information. When you are reading in the information, make sure that you are storing the values in the most appropriate types.

In particular, this class should have the following methods:

* **constructor:**This method should take the filename as a parameter and should read in the file of accessibility tests and store them in an ArrayList. The constructor should use the try-catch exception handling to print an error message if an invalid filename is found
* **numTests:**This method takes no parameters and returns the number of tests that are stored in the ArrayList.
* **showTestResults:** This method takes test details (or a portion of the test details) as a parameter, and displays the test information of all tests that match (or contain) that detail (should be case insensitive). As before, the number of matching tests should be displayed at the end. For example, if the information from a11yCheckersResults.txt was read in and stored in aAccessibilityResults object named a11y, then a11y.showTestResults("Colour") should display the following:

Colour/Contrast: Colour alone is used to convey content Google: notfound WAVE: notfound SortSite: notfound ASLint: manual  
Links: Identifying links by colour alone Google: notfound WAVE: notfound SortSite: notfound ASLint: notfound  
Forms: Errors identified by colour only Google: notfound WAVE: notfound SortSite: notfound ASLint: manual  
Forms: Errors identified with a poor colour contrast Google: warning WAVE: error SortSite: error ASLint: error  
HTML: Inline style adds colour Google: notfound WAVE: notfound SortSite: notfound ASLint: notfound  
  
Total tests matching: 5

* **showByCategory:** This method takes a category (or a portion of the category) as a parameter, and displays the test information of all tests that match (or contain) that category (should be case insensitive). As before, the number of matching tests should be displayed at the end. For example, if the information from a11yCheckersResults.txt was read in and stored in aAccessibilityResults object named a11y, then a11y.showByCategory("keyboard") and a11y.showByCategory("Key")should display the following:

Keyboard: Alert shows for a short time Google: notfound WAVE: notfound SortSite: notfound ASLint: notfound  
Keyboard: Lightbox - close button doesn't receive focus Google: notfound WAVE: notfound SortSite: notfound ASLint: notfound  
Keyboard: Focus order in wrong order Google: notfound WAVE: notfound SortSite: notfound ASLint: notfound  
...  
Keyboard: Fake button is not keyboard accessible Google: notfound WAVE: notfound SortSite: notfound ASLint: notfound  
  
Total tests in category: 16

* **showAllFailed:** This method takes no parameters, and displays the tests that all checkers failed (i.e. a test is only shown if Google, WAVE, ASLint, and SortSite failed it and if even one of the checkers passed the test, it is not shown) and the number of tests that all failed. A failed test is the result "notfound". For example, if the information from a11yCheckersResults.txt was read in and stored in aAccessibilityResults object named a11y, then a11y.showAllFailed() should display the following:

Content: Content identified by location Google: notfound WAVE: notfound SortSite: notfound ASLint: notfound  
...  
HTML: Inline style adds colour Google: notfound WAVE: notfound SortSite: notfound ASLint: notfound  
HTML: PRE element without CODE element inside it Google: notfound WAVE: notfound SortSite: notfound ASLint: notfound  
  
Total tests failed: 51

* **numPass:** This method takes two parameters of the checker name and category and returns the number tests that had a result of either error or error\_paid. (Note this method should work with a partial checker name or partial category name and should be case insensitive). For example, if the information from a11yCheckersResults.txt was read in and stored in a AccessibilityResults object named a11y, then a11y.numPass("Goog","") should both return 23 and a11y.numPass("lint","htm") should return 2.

Be sure to include javadoc comments in your classes, including your name in the top comment block. Avoid redundancy and be conservative in your use of fields - if a data value does not need to persist over the life of the object, declare it to be local to the method that needs it.

Part 3: Write Up

Answer the following questions in a word document:

1. Complete the following table with the number of tests that each checker passes for each category (in the last column, place the number of tests in the category).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category | Google Checker | WAVE Checker | Sortsite Checker | ASLint | Total Number of Tests |
| Content |  |  |  |  |  |
| Colour/Contrast |  |  |  |  |  |
| Typography |  |  |  |  |  |
| Language |  |  |  |  |  |
| Title |  |  |  |  |  |
| Headings |  |  |  |  |  |
| Lists |  |  |  |  |  |
| Tables |  |  |  |  |  |
| Images |  |  |  |  |  |
| Multimedia |  |  |  |  |  |
| Links |  |  |  |  |  |
| Buttons |  |  |  |  |  |
| Forms |  |  |  |  |  |
| Navigation |  |  |  |  |  |
| Keyboard |  |  |  |  |  |
| Frames |  |  |  |  |  |
| CSS |  |  |  |  |  |
| HTML |  |  |  |  |  |
| Total |  |  |  |  |  |

1. Which category/categories are the accessibility checkers best at identifying errors for? Which category/categories are the accessibility checkers worst at identifying errors for? What is the general description of the types of tests in this category?
2. Select one of the tests that every checker fails and describe what the issues is and how it can affect a person with a disability.
3. Reflect: How does this assignment shape your understanding of computer science?