**Overview**

In this practical assignment, we were tasked to write a program that performs a word frequency analysis on the data returned from Bored API. Bored API is a simple API that suggests random activities to users, which can be filtered using the number of participants specified by the activities, the type of activity, the price of the activity etc. Apart from making queries to the online Bored API URLs, our program also needs to be able to read data from local files that form a local cache.

My solution achieves all parts of the specification. It successfully implements classes to create the required functionality as stated in the specification.

**Code Design**

There was a total of five distinct tasks that our program should be able to accomplish:

1. Search for a random activity using a Bored API URL.
2. Search for an activity of a type specified by the user using a Bored API URL.
3. Search for an activity with a specified number of participants using a Bored API URL.
4. Retrieve an activity based on its key, which would be specified by the user.
5. Iterate over all the files given in the cache directory and print the 10 most frequent words that appear the “activity” text field in each file together with the number of occurrences.

**Testing**

I started testing my program by ensuring that all the automated stacscheck tests passed. As shown in the stacscheck output below, my program satisfied all of the provided tests.

|  |
| --- |
| Testing CS1003 Assignment #1 - Bored API  - Looking for submission in a directory called 'src': Already in it!  \* BUILD TEST - build : pass  \* COMPARISON TEST - badargs/0arg/prog-expected.out : pass  \* COMPARISON TEST - badargs/1arg/prog-expected.out : pass  \* COMPARISON TEST - badargs/1argwrong-cache/prog-expected.out : pass  \* COMPARISON TEST - badargs/2arg2-location/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-key/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-participants/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-random-wrong-cache/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-summary-wrong-cache/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-type/prog-expected.out : pass  \* COMPARISON TEST - badargs/3args-random/prog-expected.out : pass  \* COMPARISON TEST - badargs/3args-summary/prog-expected.out : pass  \* COMPARISON TEST - badargs/3args-type-wrong/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-1638604/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-2790297/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-3305912/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-4101229/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-5881028/prog-expected.out : pass  \* TEST - wellformed/participants-1/test : pass  \* TEST - wellformed/participants-2/test : pass  \* TEST - wellformed/participants-3/test : pass  \* TEST - wellformed/participants-4/test : pass  \* TEST - wellformed/participants-5/test : pass  \* TEST - wellformed/random/test : pass  \* COMPARISON TEST - wellformed/summary/prog-expected.out : pass  \* TEST - wellformed/type-busywork/test : pass  \* TEST - wellformed/type-charity/test : pass  \* TEST - wellformed/type-cooking/test : pass  \* TEST - wellformed/type-diy/test : pass  \* TEST - wellformed/type-education/test : pass  \* TEST - wellformed/type-music/test : pass  \* TEST - wellformed/type-recreational/test : pass  \* TEST - wellformed/type-relaxation/test : pass  \* TEST - wellformed/type-social/test : pass  34 out of 34 tests passed |

Figure 1: All stacscheck tests were successfully passed.

**Evaluation**

Overall, my program successfully implements all the requirements in the assignment. I have run my program extensively to ensure that it works as intended and also included comments as a form of documentation within each Java file.

**Conclusion**

This assignment provided an enjoyable and insightful test of my programming skills. It served as a nice consolidation of what I have learnt over the past few weeks. Among the various concepts I had to implement, I found that inheritance was the trickiest as it was a relatively new concept to me. However, I ultimately understood it and managed to utilize it within my solution successfully.

If given more time to come up with a solution, I would like to attempt to use generics within my code. While I am currently not very familiar with the concept and syntax, I feel like it could be a viable approach when expanding the code to deal with other types of grids that a user might want to implement.

Overall, I viewed this assignment as a meaningful conclusion for all the concepts that I have learnt over the course of CS1002, and I'm content with the final solution I came up with.

**References**

1. Do a Simple HTTP Request in Java. (2024, February 8). Baeldung.com. <https://www.baeldung.com/java-http-request>
2. How to Sort ArrayList using Comparator? (2020, December 14). GeeksforGeeks. <https://www.geeksforgeeks.org/how-to-sort-arraylist-using-comparator/>
3. JsonObject (Java(TM) EE 7 Specification APIs). (2015, June). Oracle.com. <https://docs.oracle.com/javaee%2F7%2Fapi%2F%2F/javax/json/JsonObject.html>
4. JsonParser (Java(TM) EE 7 Specification APIs). (2015, June). Oracle.com. https://docs.oracle.com/javaee%2F7%2Fapi%2F%2F/javax/json/stream/JsonParser.html