



# DATA STRUCTURES.

[Document subtitle]

## ABSTRACT

This report is about the applications `GenericsKbBSTApp` and `GenericsKbArrayApp` which both store Generic truths but use different methods to store them

Kagiso Dube

Computer Science

## **Introduction**

GenericsKbArrayApp and the GenericsKbBSTApp both query the GenericKB database which contains over 3 million generic sentences about general truths. Both applications store the facts but in different way like the GenericsKbArrayApp stores the facts in a traditional array while the GenericsKbBSTApp stores the facts in a binary search tree.

## **Object-Oriented Design**

### **GenericsKbArrayApp:**

Six classes were created in total if we count both apps. The first class is called Item which separated each fact into three parts the term, the statement and the confidence score. The class makes it easier for the classes to differentiate between the term, the statement and the confidence score and it makes comparison between facts easier. The following class is called ReadFile which reads the file and stores the facts (which are now Item objects) into a traditional array so it is easier for us to update, search and insert a fact. The third class is the main class of the first application. It deals with the front end of the program; it allows the user to enter the source file and provides the user with options that allow the user to update knowledge base, add to the knowledge base or search for a fact through the knowledge base. The main class performs all these things by using methods that are in the ReadFile class and item class.

### **GenericsKbBSTApp:**

This application utilizes the Item class, Term, BinarySearchTree (BST) class and the GenericsKbBSTApp (main class). The term class which you can consider as the node class, it stores the fact but the difference here is that this class allows the fact to have a link to the other facts that follow it, and this forms the foundation for our BinarySearchTree class. The BinarySearchTree class or BST for short stores the facts in a sorted manner, it uses the term class by storing the node and setting the value of the following nodes to null until another node is entered which will take place for the null value. The class compares the term of the fact that is being added to the term of the facts that are already in the tree and inserts the fact in the proper position. This makes it easier to search and I don't have to go through all the facts to see where a fact belongs. The final class GenericsKbBSTApp handles the front end of the program and provides the user with multiple option like to search for a fact, enter knowledge base, update the knowledge base and what's different about this program is that the main class reads the knowledge base.

## **BinarySearchTree vs Traditional Array Implementation**

For the binarysearchtree I created the term class which acts like a node, it stores the generic truth and allows it to have links to the next generic truths. The tree had the instance variable root which formed the bases of our code. We would add a term/node at the root if the root was empty because that meant we had no tree, but if there was a tree we checked if the term of the generic truth comes before the root or after the root(In this case we did it alphabetically). If term came before previous alphabetically then we

check the left subtree to see where we can the term but if it comes after the previous term then we check the right subtree to see where the node should be added.

For the traditional array I just made the array to store the type of Item. Item is a class that extracts the term and saves it in its own variable and the other parts of the generic truth in their own variables. The array stores all the generics truths without sorting them.

### **Creative aspects and Non-functional Requirements**

For the front end of the application, I use java graphical user interface. I used the Optionpane to display the options to the user and accept user input. If ever the user does something that is not accounted for in the application or I want to inform them of something, the application displays a message to the user. The applications prevent the user from using the cancel button to end the program, but they must use one of the options provided to them to end the program. Both the pane and the message dialog appear on a frame which acts like a background for the applications.

The application accounts for many things that could go wrong. Firstly, if the user was to ever enter the incorrect option (option that was not displayed to them.) Like the using entering a word or sentence or a whitespace. The user is notified if they enter a name of a file that doesn't exist or wrong file location. When the user is inserting a new generic truth or trying to update an already existing generic truth the application doesn't allow the user to enter whitespaces only, they must insert an actual word (any sequence of letters) or sentence and for entering the confidence score the user cannot continue with the processes until they enter a real number, and the number must be between 0 and 1.

When ever the user activates the program it firsts tells them what they should do in order for the program to function correctly. It tells them that they must first insert a knowledge base before they can choose the other options, or they can just quit the program.

### **Discussion and Conclusion**

When comparing the two applications the genericsKbBST app seems to be more efficient than the other one. Firstly, when it comes to searching for a generic truth the BST app got the traditional array beat since in the worst case where the generic truth were looking for is at the very end we don't have to go through the entire dataset looking for it but we can just focus on one subtree which contains fewer generic truths than the dataset, but in the traditional array we would have to go through the entire array just to get the last generic truth.

The binary search tree also sorts the generic truth, so the searching process becomes a lot easier. The only time where the array has the binary tree beat is when we insert a new generic truth, the array just jumps to the very end and inserts the generic truth while in

the binary tree it still must compare the new generic truth to the others so it can find a proper position for the truth.

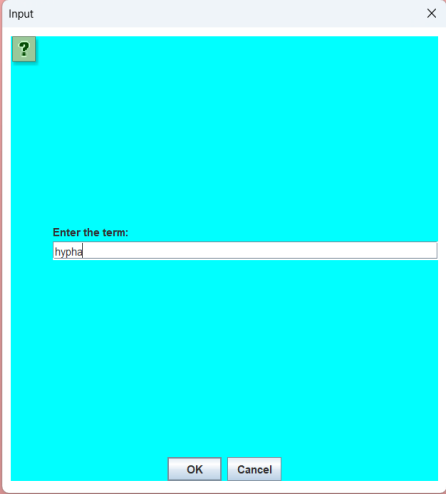
The binary tree uses less space since for the traditional array we have to pre define the number of truths that it store but knowing that the truths could be more than a million we have to set a very huge number which ends using a lot of space but the binary uses the exact amount of space needed to fit all the generic truths even when the user enters a new one the generic truth will account for that new addition.

## **Test Values**

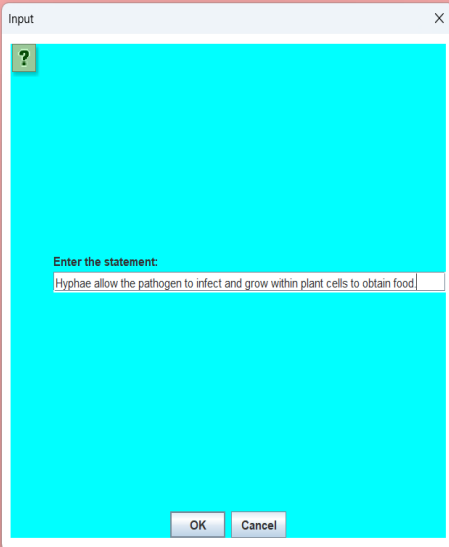
Testing when user chooses option 1:



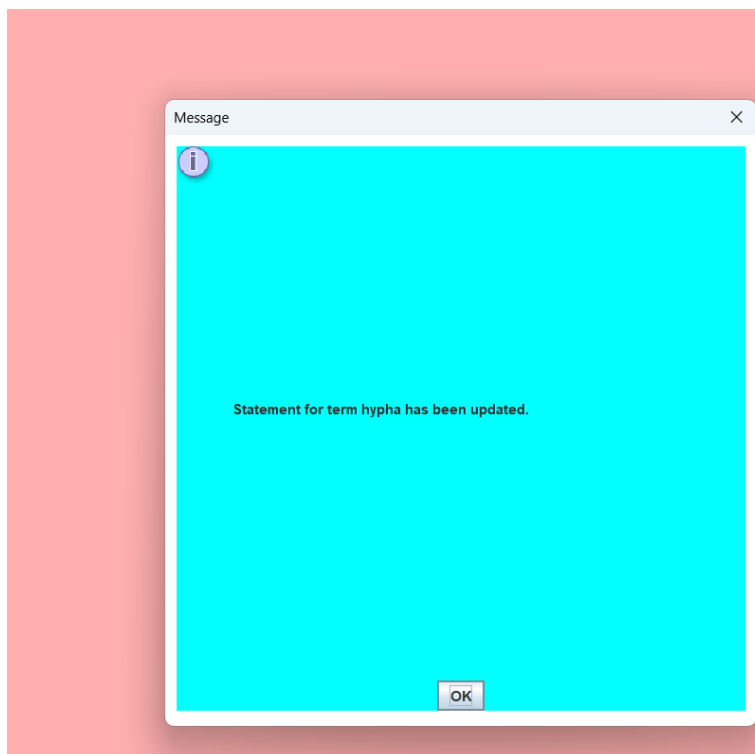
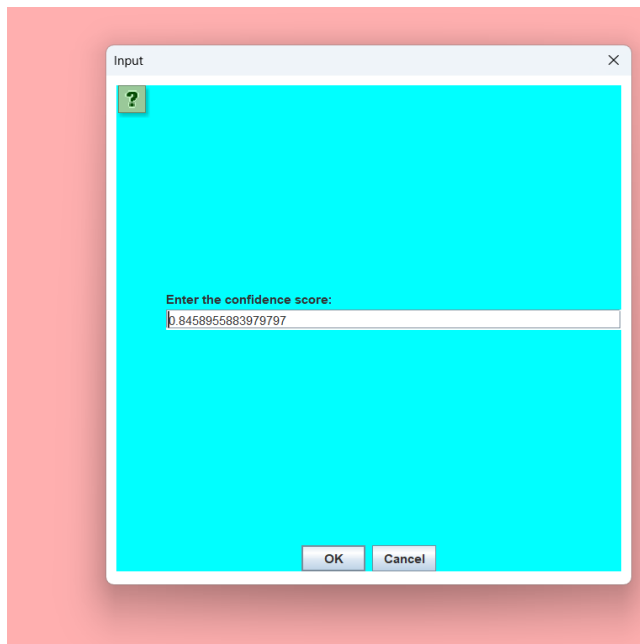
**Testing option 2: User enters a new fact for already existing term.**



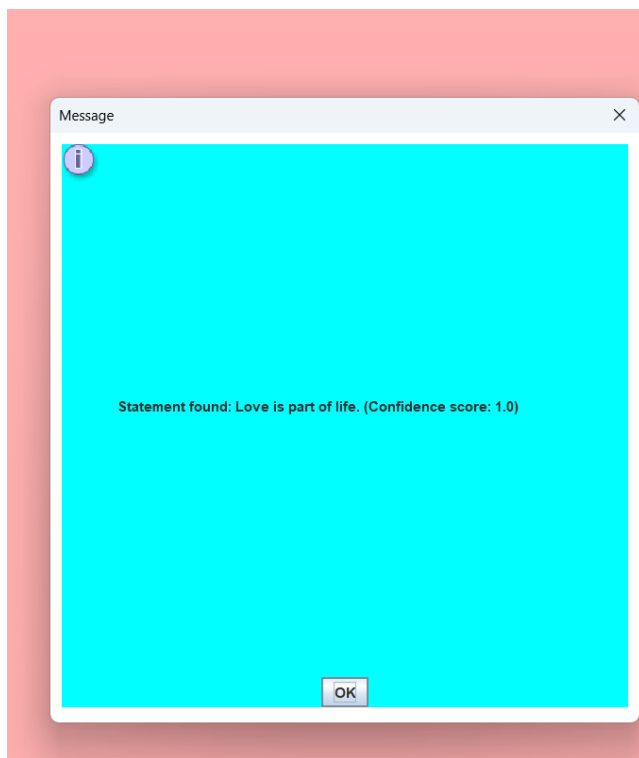
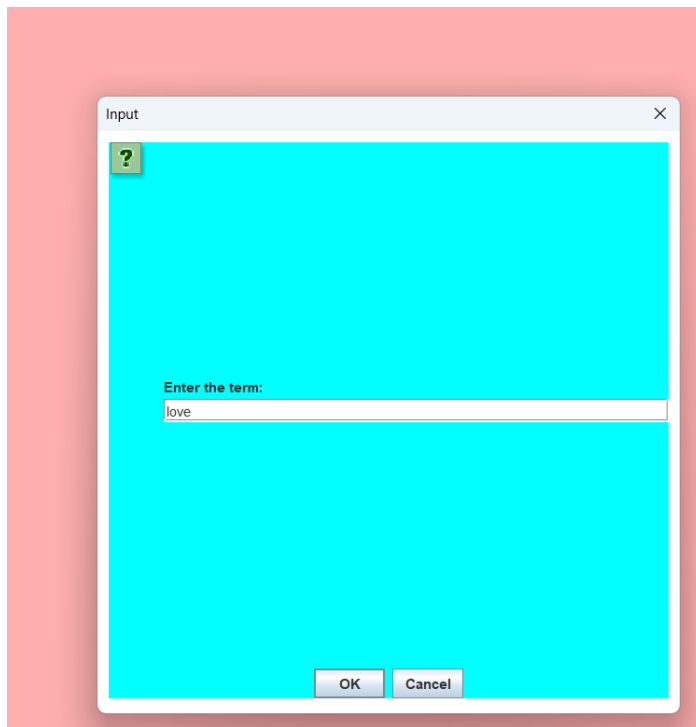
The screenshot shows a dialog box titled "Input" with a close button (X) in the top right corner. Inside the dialog, there is a green question mark icon in the top left. Below it, the text "Enter the term:" is displayed above a text input field. The input field contains the word "hypha". At the bottom of the dialog, there are two buttons: "OK" and "Cancel".



The screenshot shows a dialog box titled "Input" with a close button (X) in the top right corner. Inside the dialog, there is a green question mark icon in the top left. Below it, the text "Enter the statement:" is displayed above a text input field. The input field contains the sentence "Hyphae allow the pathogen to infect and grow within plant cells to obtain food". At the bottom of the dialog, there are two buttons: "OK" and "Cancel".



**Testing Option 3: User searching for fact using the term.**



**Test Option 4:** Using statement and term to search for generic truth.

Input

?

Enter the term:

hypha

OK Cancel

Input

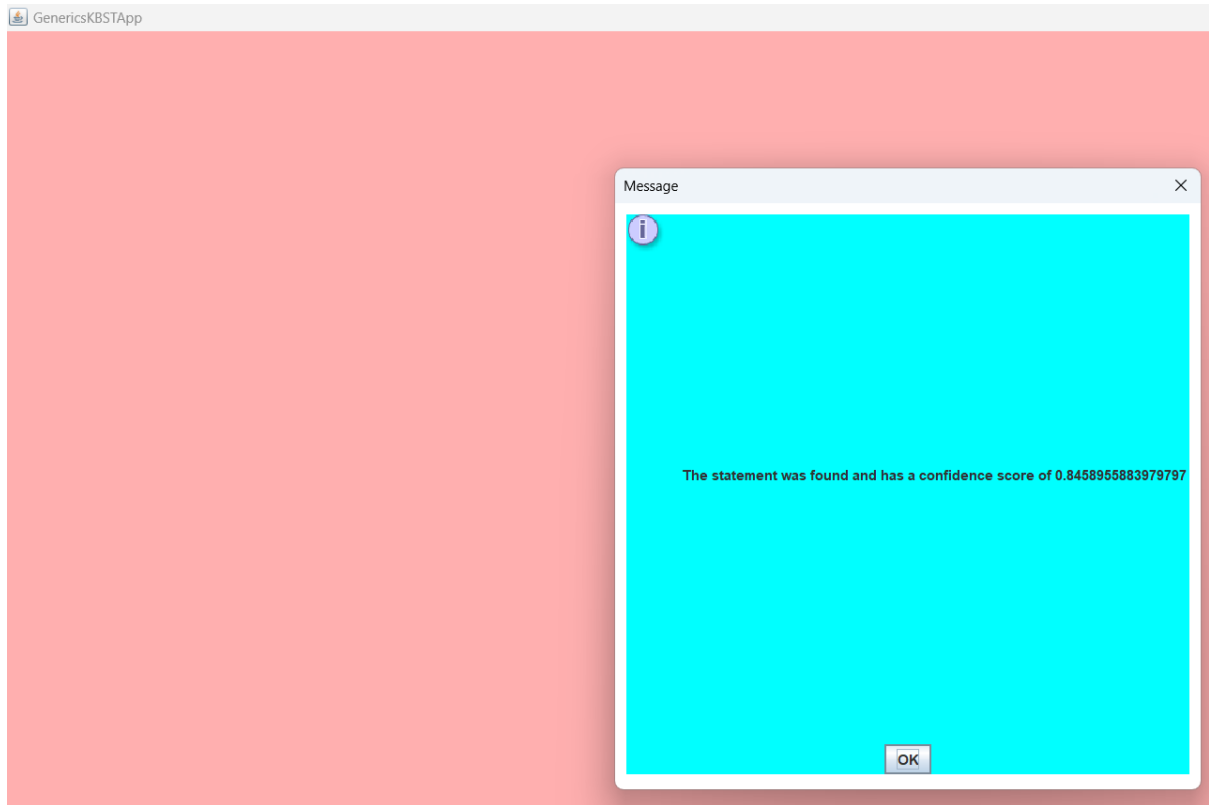
?

Enter the statement:

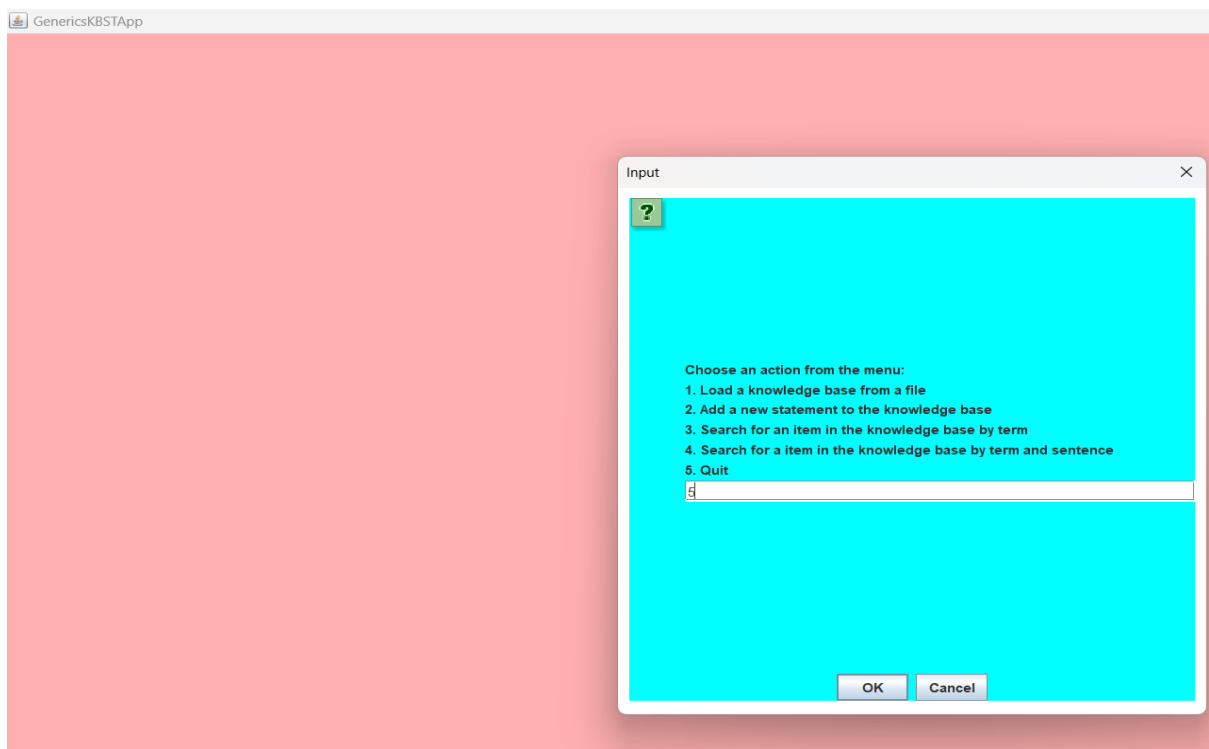
Hyphae allow the pathogen to infect and grow within plant cells to obtain food.

OK Cancel



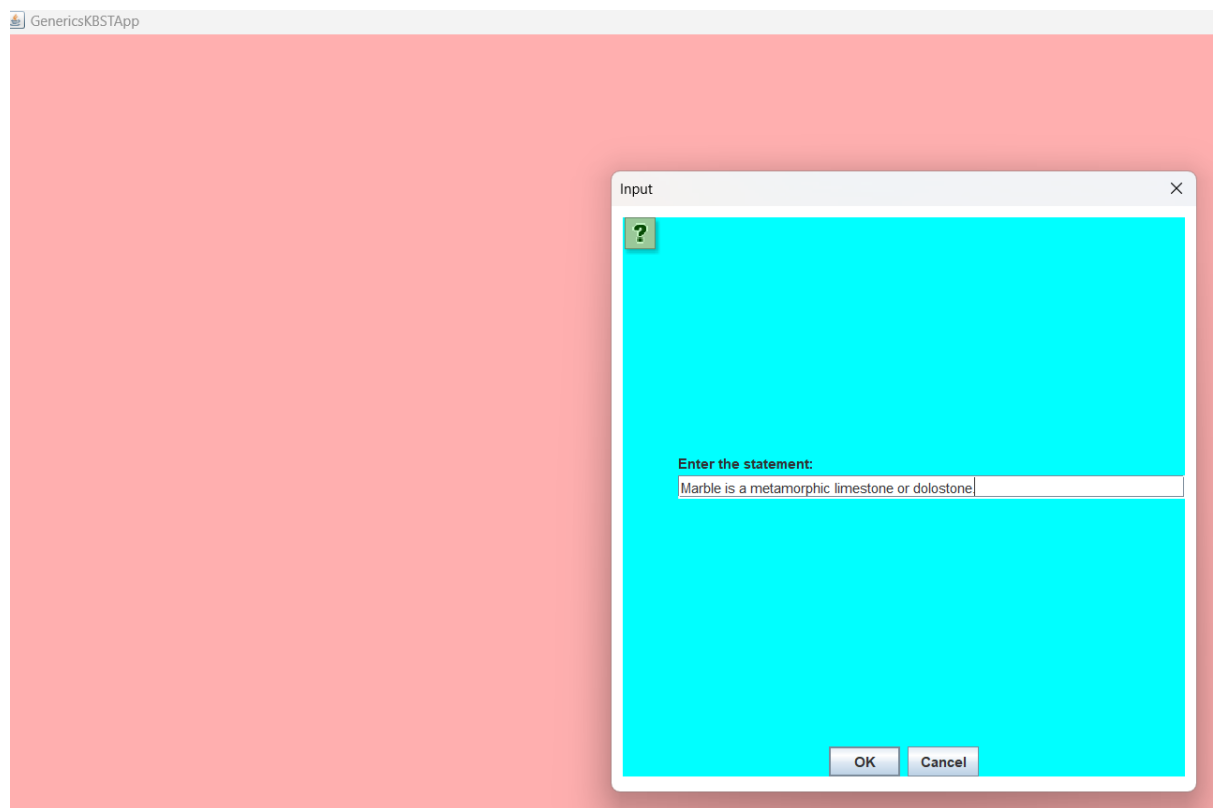
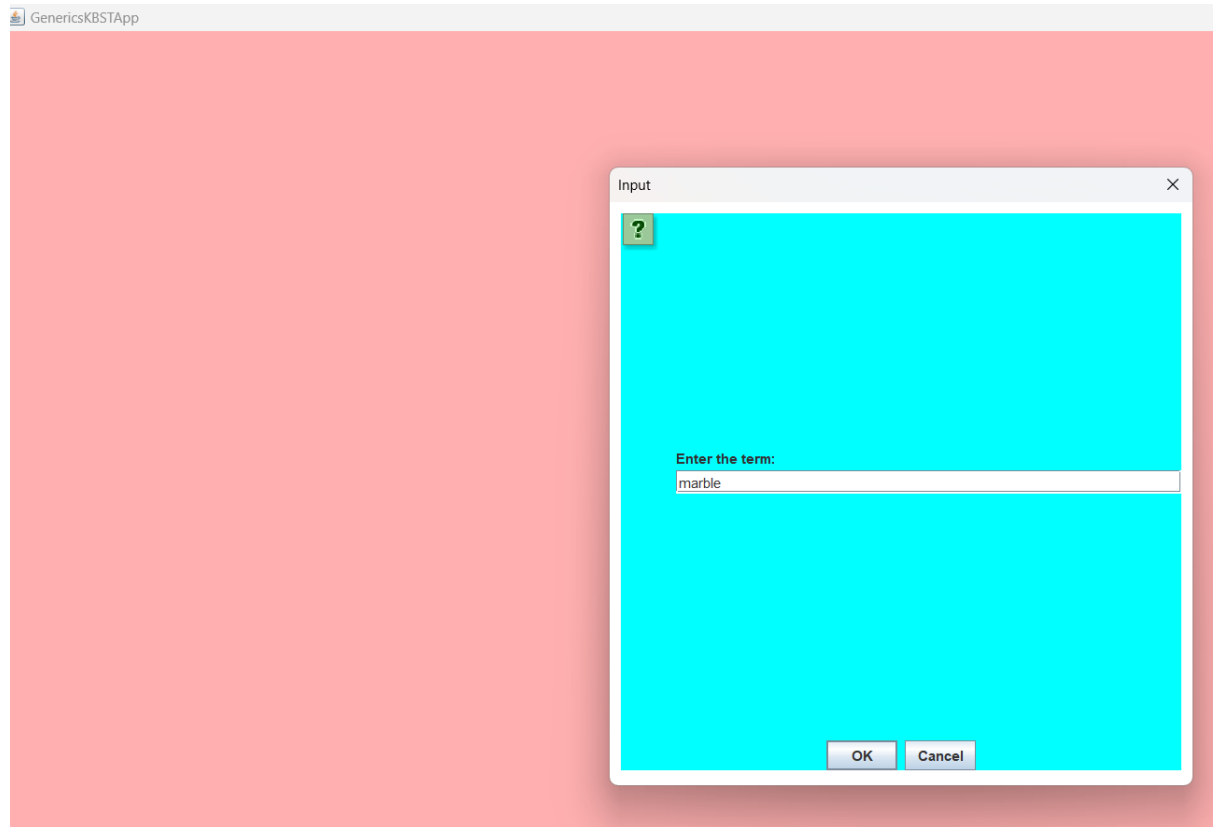


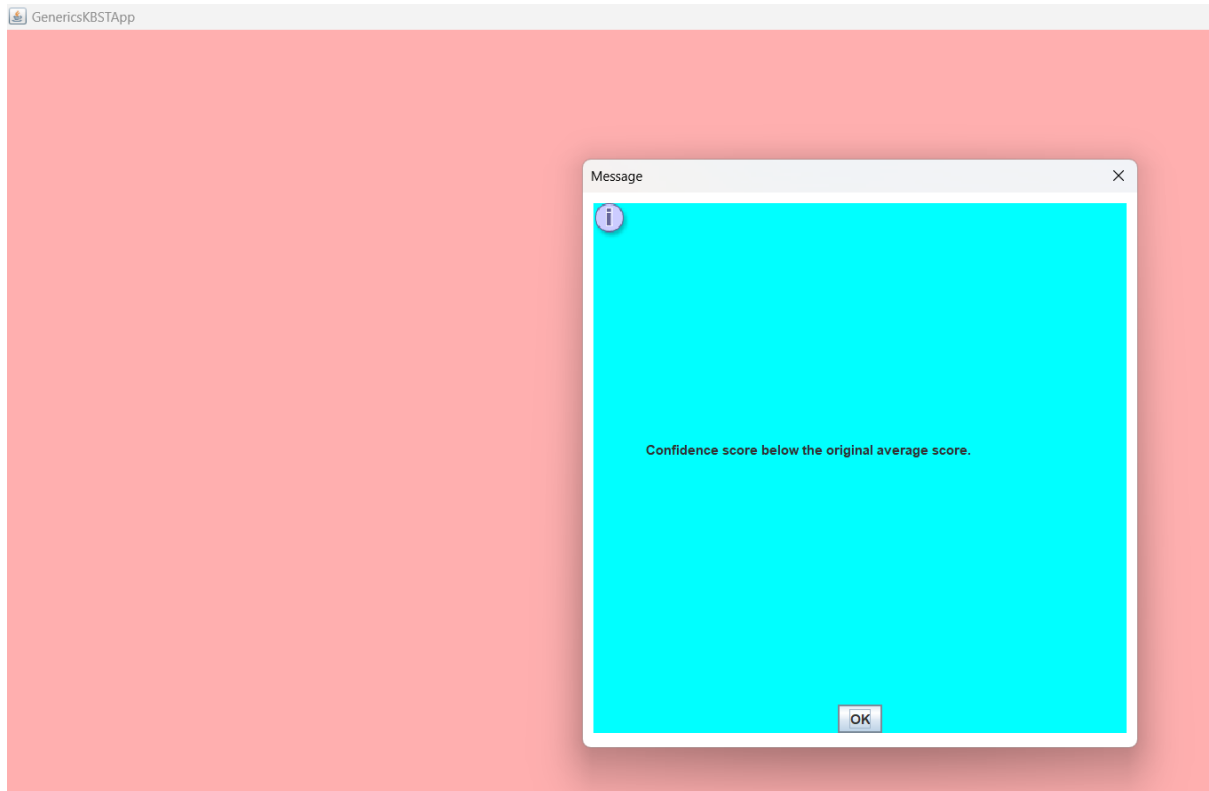
**Test Option 5:** Using the quit option.



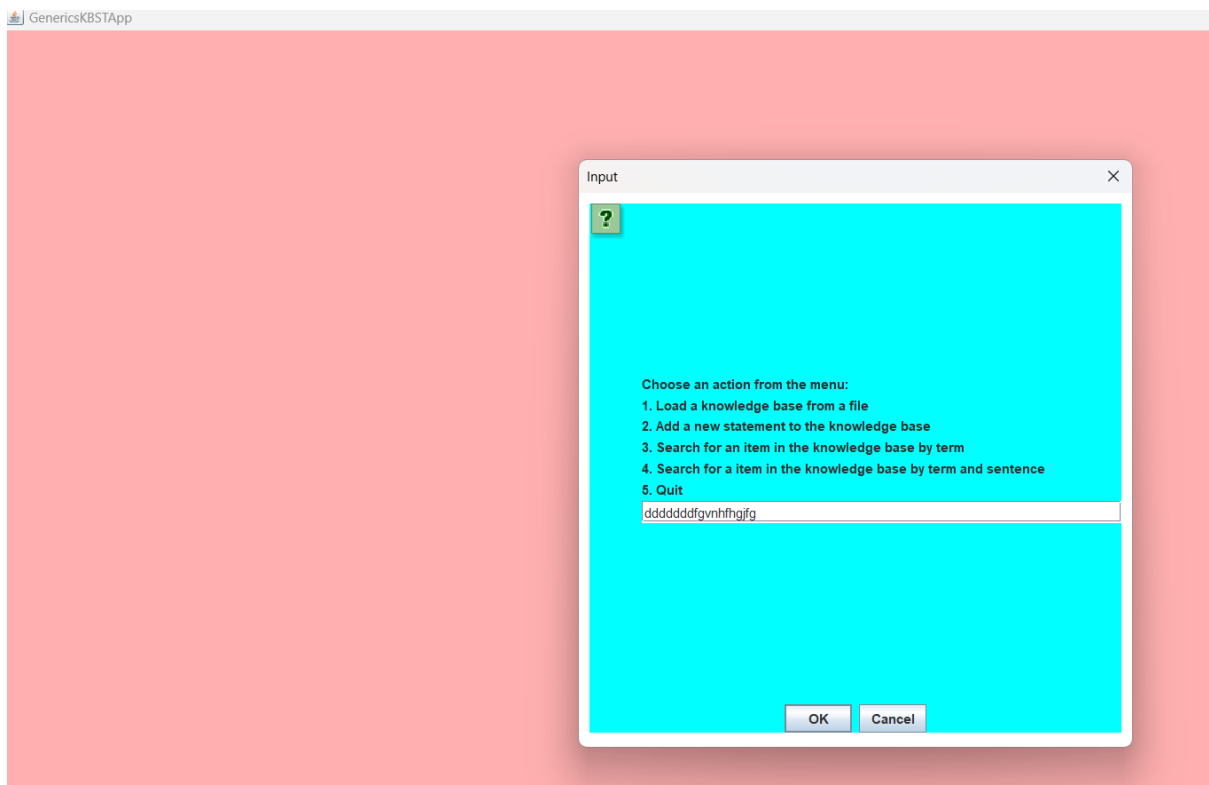
**Test Option 2:** The confidence score of the new Generics fact is below the old one.

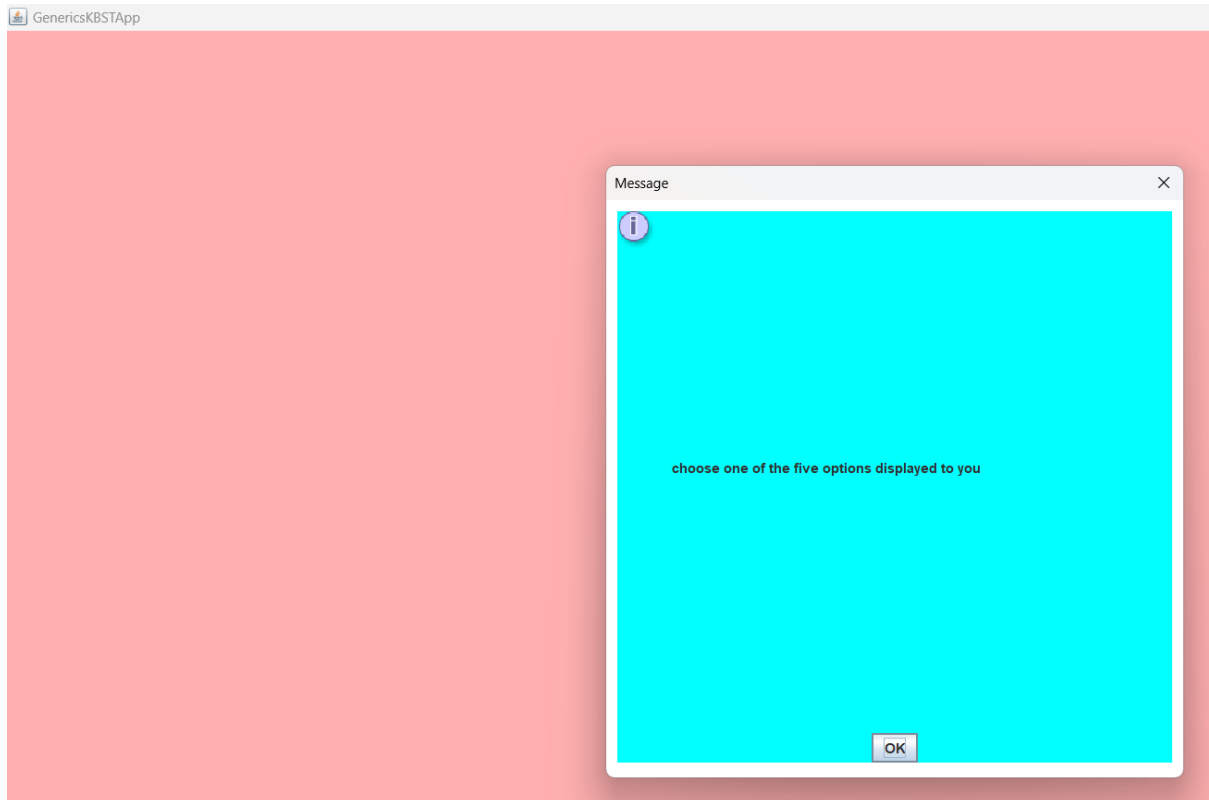
The term marble already exists with a perfect confidence score.





**Test:** When user doesn't choose any of the five options given to them.





## Git Log

```
1 commit 8f670d7b3ee460299afd56701fcf3527ecd904e5
2 Author: k11logram <161375355+k11logram@users.noreply.github.com>
3 Date: Fri Mar 8 00:01:50 2024 +0200
4
5 Add files via upload
6
7 commit 3a6ef3a8037c7b6a33e84daa5f528d99abab3517
8 Author: k11logram <161375355+k11logram@users.noreply.github.com>
9 Date: Thu Mar 7 23:31:29 2024 +0200
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55 commit ec928c59e6aeb0c7e675017b8a8f6123a49f4ea9
56 Author: k11logram <161375355+k11logram@users.noreply.github.com>
57 Date: Mon Feb 26 21:47:09 2024 +0200
58
59 Add files via upload
60
61 commit dfe92fa709d557ce44f1a4bfe13d0fd793f4d8fe
62 Author: k11logram <161375355+k11logram@users.noreply.github.com>
63 Date: Mon Feb 26 21:43:12 2024 +0200
64
65 Initial commit
```