This report is based on a program I created which allows users load data from a knowledge base into an array or binary search tree then query the loaded information, while also incorporating new knowledge or updating the existing knowledge base.

CSC2001F 2024 Data Structures Assignment 1

MSHPRI017

Prince Mashava

Mshpri017@myuct.ac.za

# Appropriate design and implementation of OOP:

The data read in from textfiles was formatted in the following way, (term, sentence, confidence score) all separated by tabs. So, I created a class called, Kb which would have these three fields as instance variables and can be used to create an object of type, Kb. The datatypes of the term and sentence are String, and the confidence is of type double.

The constructor of the class takes in a line from the textfile which has a term, sentence, and score. From there the line is sliced into its three parts and the instance variables are assigned their appropriate values.

The class, Kb has get methods that return the term, sentence and confidence score of the object. As well as set methods to modify the instance variables.

There is a toString() method that returns the term, sentence and confidence score altogether separated by tabs.

I also created two methods to search for statements. One is called, matchTerm which takes in a String argument and checks if it matches the term of Kb object exactly. And another is called, matchTermPartial which takes in a String argument and checks if it partially matches the name of the Kb object.

I created a similar method called matchSenPartial which checks if an entered sentence matches the description of a Kb object.

There is a method called replace which will takes in a Kb object as an argument and will replace the term, description and confidence score of the current object.

# Traditional Array & BST implementation

The user gets to choose if they want to load the database into an array or BST.

If they choose an array, then an array of Kb objects is used. If the file entered by the user exists then each line from the file will be read using a scanner and for each line, a Kb object is created. Before inserting a new object, the array is checked for an object with the same term. If an object with the same term exists, and it has a lower confidence score than the new Kb object then the object in the array is replaced by the new object with a higher confidence score using the replace method. If an object with the same term is not found, then the object is added to the array. To find a term, a for-loop is used to traverse through the array and check for an entered term.

If they choose a BST, then a BST of Kb objects is used. If the file entered by the user exists then each line from the file will be read using a scanner and for each line, a Kb object is created. Before inserting a new object, the BST is checked for an object with the same term. If an object with the same term exists, and it has a lower confidence score than the new Kb object then the object in the array is replaced by the new object with a higher confidence score using the replace method. If an object with the same term is not found, then the object is added to the BST. To find a term, a recursive is used to traverse through the BST and check for an entered term.

# Experimental tests

To test Java program for querying and updating the knowledge base, I looked at various test cases to ensure that the program behaves correctly and handles different scenarios effectively. Those include:

## Loading the Knowledge Base:

* Test loading a valid knowledge base file

A screenshot of a computer program

Description automatically generated

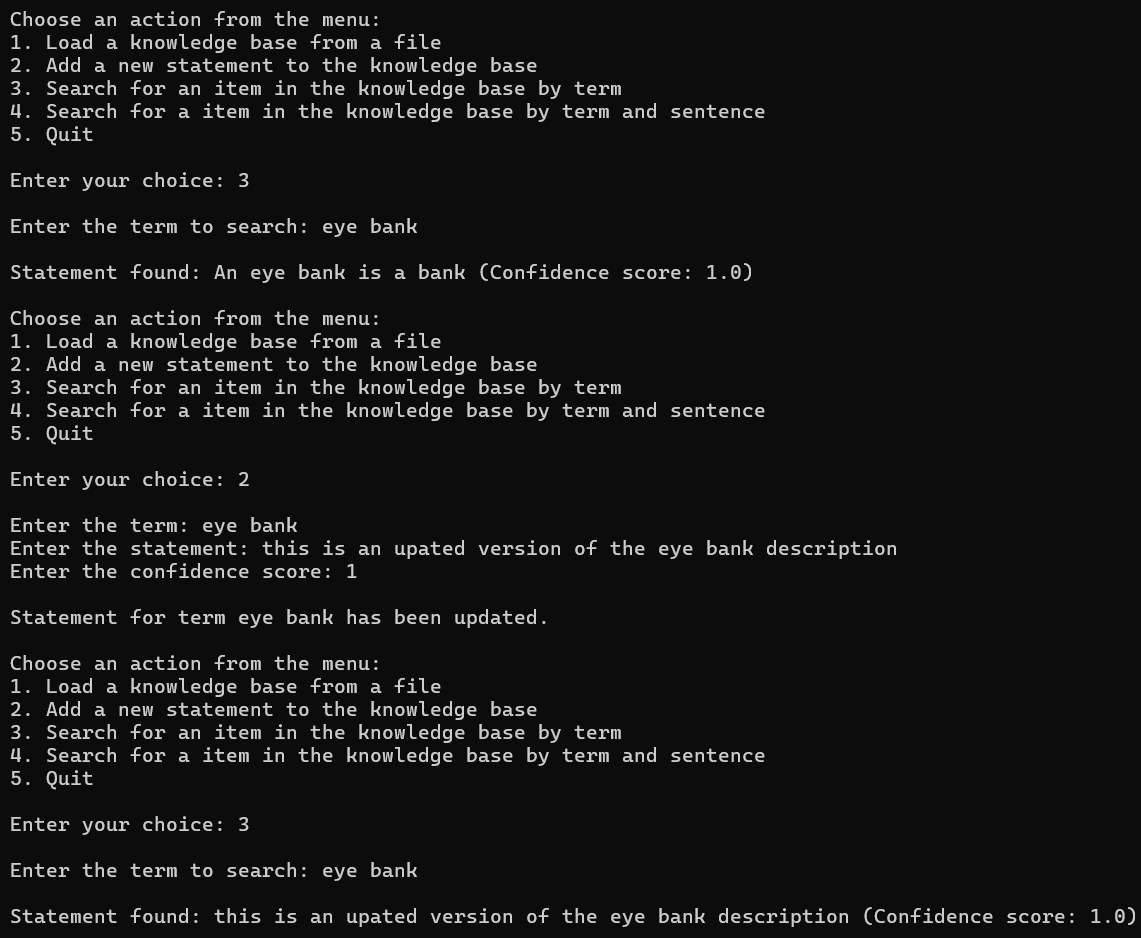
* Test loading an invalid knowledge base file

A screen shot of a computer

Description automatically generated

## Adding New Statements:

* Test adding a new statement for an existing item with a higher confidence score (Array)

****

* Test adding a new statement for an existing item with a higher confidence score (BST)

**A screenshot of a computer program

Description automatically generated**

* Test adding a new statement for an existing item with a lower confidence score (Array)

A screenshot of a computer program

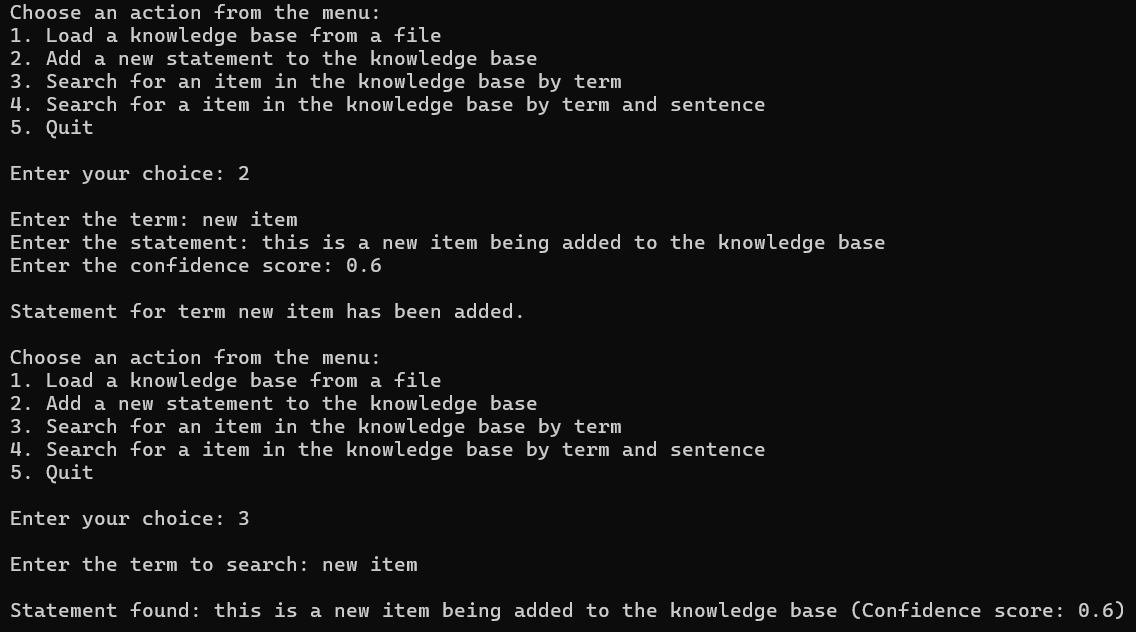
Description automatically generated

* Test adding a new statement for an existing item with a lower confidence score(BST)

A black screen with white text

Description automatically generated

* Test adding a new statement for a new item (Array)



* Test adding a new statement for a new item (BST)

A screenshot of a computer program

Description automatically generated

## Search Results:

* Test displaying information for an existing term (Array)

A black screen with white text

Description automatically generated

* Test displaying information for an existing term (BST)

A black screen with white text

Description automatically generated

* Test displaying information for a non-existing term (Array)

A black screen with white text

Description automatically generated

* Test displaying information for a non-existing term (BST)

A screenshot of a computer program

Description automatically generated

* Test searching for a statement by term and sentence(Array)

A black screen with white text

Description automatically generated

* Test searching for a statement by term and sentence(BST)

A black screen with white text

Description automatically generated

* Test searching for a non-existent statement by term and sentence(Array)

A screen shot of a computer

Description automatically generated

* Test searching for a non-existent statement by term and sentence(BST)

A screenshot of a computer program

Description automatically generated

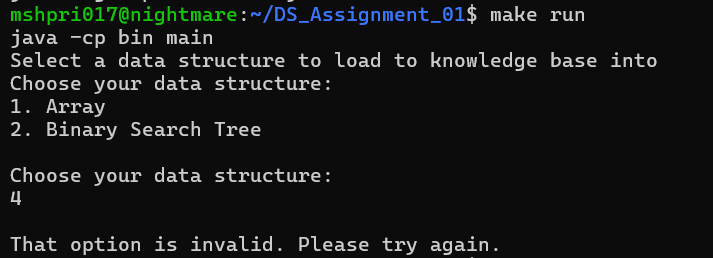
## Error Handling

* Test handling invalid user inputs (e.g., a confidence score that’s not in the 0-1 range)

A screenshot of a computer program

Description automatically generated

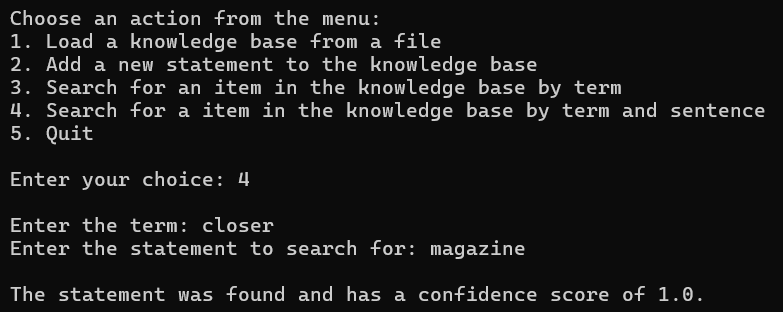
* Test handling invalid user inputs (e.g., a data structure that’s not either 1 or 2)



## Creativity:

I created methods in the Kb class that allows users to search for statements using partial keywords. They can enter a word that appears in the term and a word that appears in the description and if the entered term and sentence are part of a statement’s term and description, then a result will be returned to them. I also made all the searches case insensitive so that the user can receive results even if they enter the wrong cases to make the program easier to use.

* Test searching for a statement by partial term and sentence(Array)



* Test searching for a non-existent statement by partial term and sentence(Array)

A screenshot of a computer program

Description automatically generated

* Test searching for a statement by partial term and sentence(BST)

A screen shot of a computer

Description automatically generated

* Test searching for a non-existent statement by partial term and sentence(BST)

A screenshot of a computer program

Description automatically generated

## Git usage log

A black screen with white text

Description automatically generated