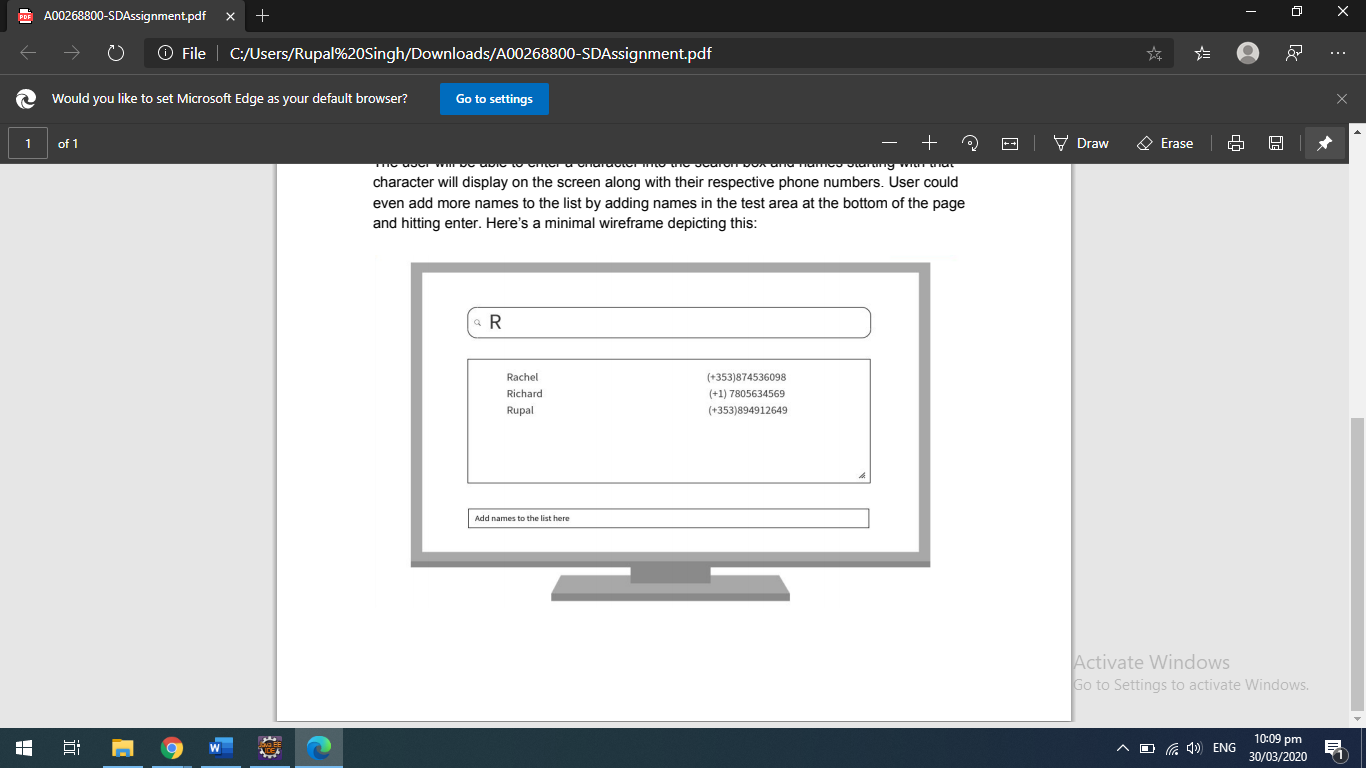
**Software Design 4.2 Assignment | A00268800**

**Implementing TRIE, LinkedList and HashMap**

**PHONE BOOK DIRECTORY**

**Brief Description:**

Created a Phone Book Directory using Java Swing and Collection Framework. TRIE data structure, LinkedList and HashMap have been used to store and retrieve contacts by entering names in the search bar. New entries can be saved by adding the name and phone number. An existing entry/phone number can be searched by writing prefix of the respective name to enable the search query to find names with same prefix, that can later be displayed on a table in the centre of the screen.



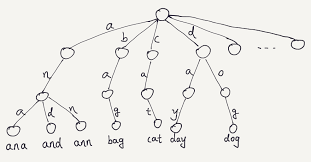
**Crux of the Trie Data Structure:**

Based on the concept of prefix retrieval TRIE appeared the best data structure for the chosen use case as it allows us to work with large data or groups of strings. Clearly, a naïve approach would cost a lot of time and effort while comparing each string with the required one. This would be an expensive process in terms of the time involved. Tries solve this issue by providing the result in a much more efficient way.

The **complexity** of creating a **trie** is O(W\*L), where W is the number of words, and L is an average length of the word: you need to perform L lookups on the average for each of the W words in the set.

To contrast this, let’s consider Computer Science’s poster child: the Binary Tree. Performance of a binary tree for let’s say operation x is O(log(n)) ie log base 2. But what if, instead of a Binary Tree, we used a ternary tree, where every node has three children. Then, we’d be talking log base 3. (That’s a performance improvement, albeit only by a constant factor.) Thus, our tree would become wider and shorter so we could perform lesser lookups without having to descend quite so deep.

This is the motivation behind the trie. Yes, it’s a tree. A trie tree.



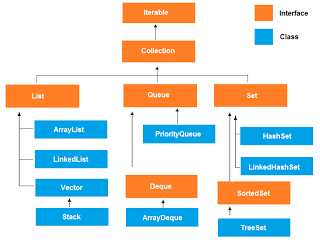
**Features:**

One can add numbers to the directory using LinkedList. Contacts are added as key value pair that is being dealt with the HashMap. All available contacts are displayed in the table. Any contacted can be looked up by typing the prefix of the name who’s contact one wants to find. This retrieval is done using Trie.

**Why Collection Framework?**

The purpose of this project is to implement data structures in Java as a part of the Collection Framework. Collection Framework are being used to:

* Reduce Programming effort
* Increase performance
* Provide Interoperability between unrelated APIs
* Reduces effort required to learn APIs
* And promotes Software Reuse



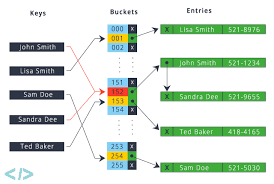
**Data Structure Explained**

1. Linked List



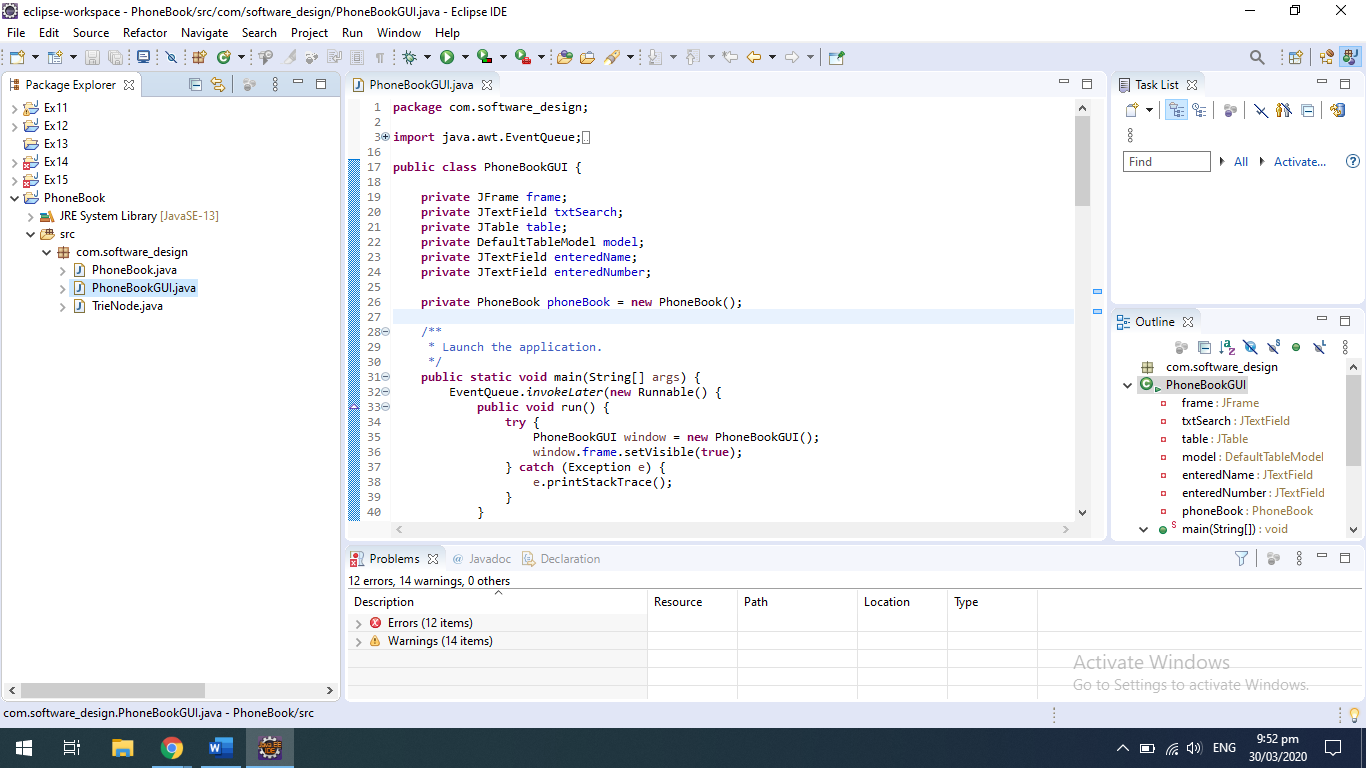
Thus linear data structure lets you store data in contiguous locations and every elememt is a separate object. Its been used to store enteries in the phonebook.

1. HashMap



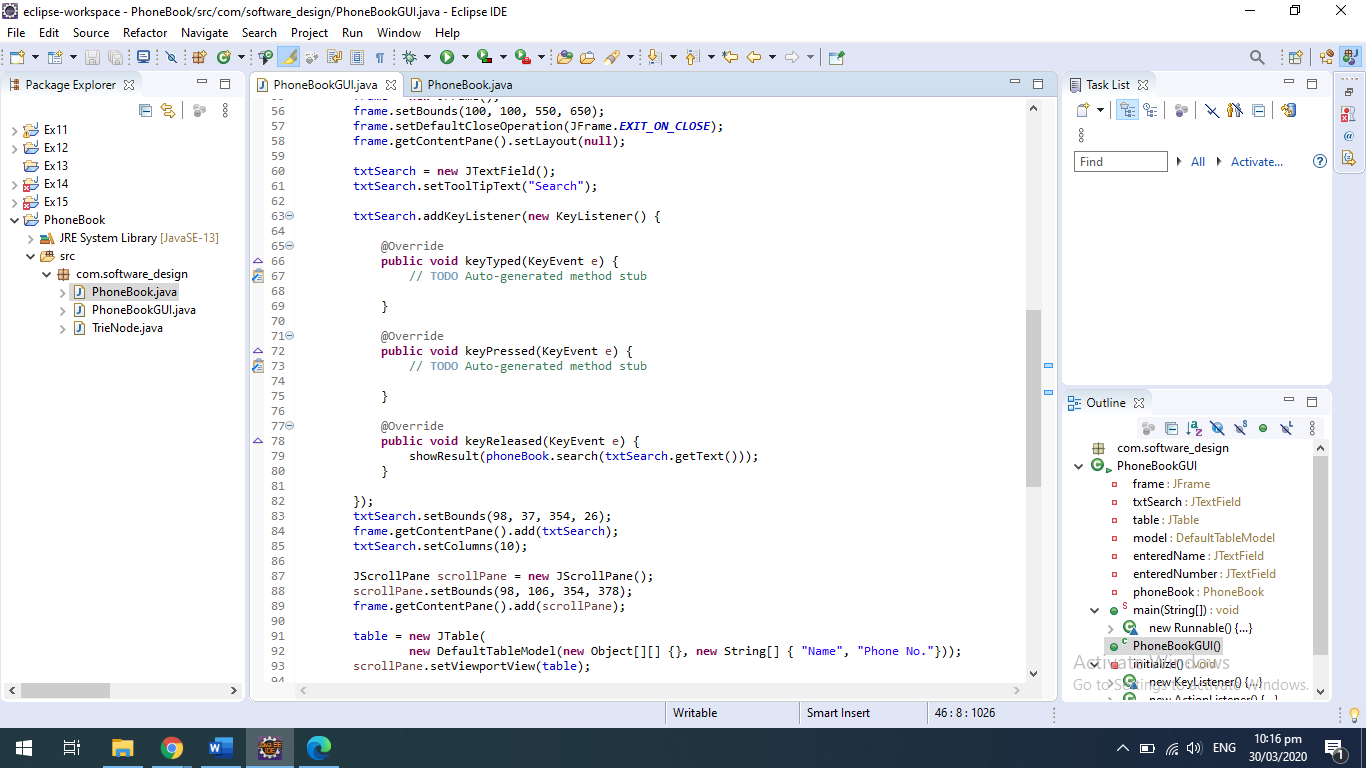
HashMap lets us store and work with data as key value pairs. The same property has been used to store data in the table in the phonebook UI.

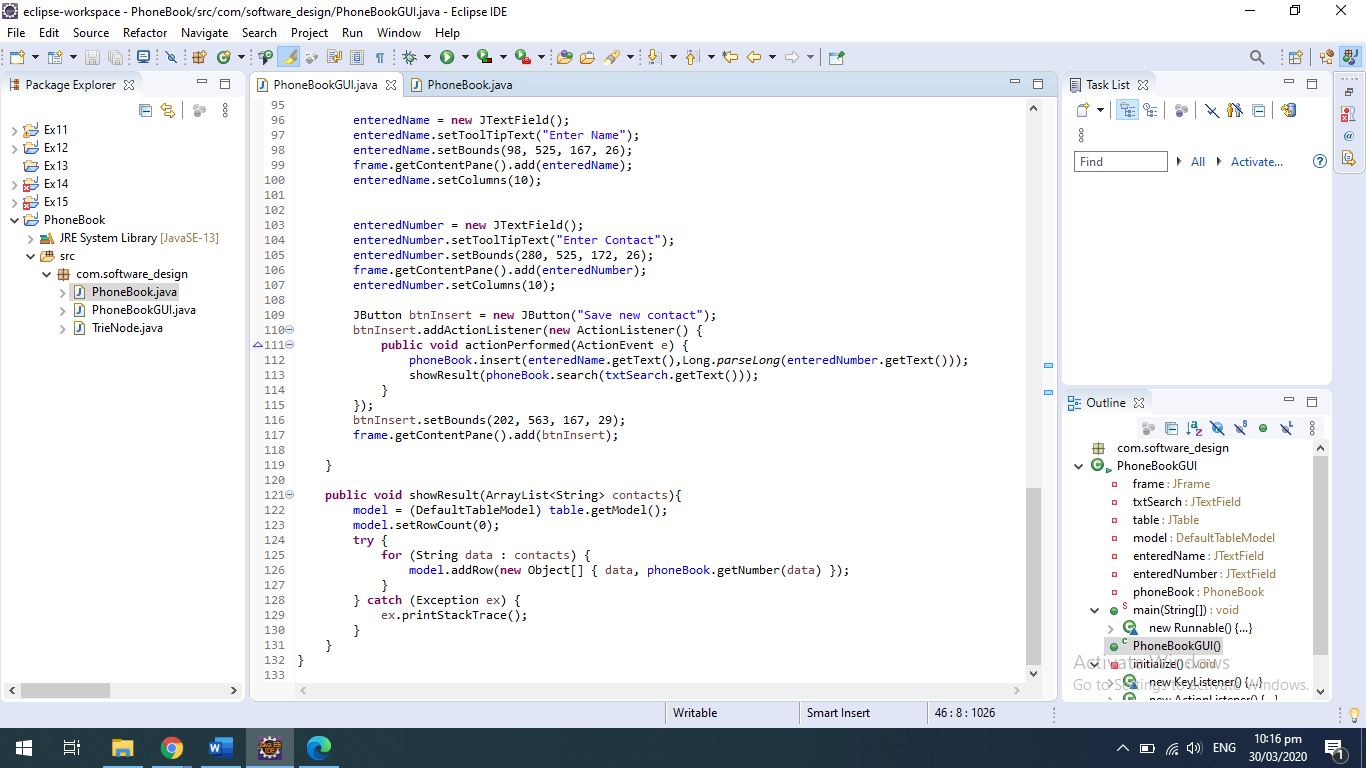
**Key Classes:**



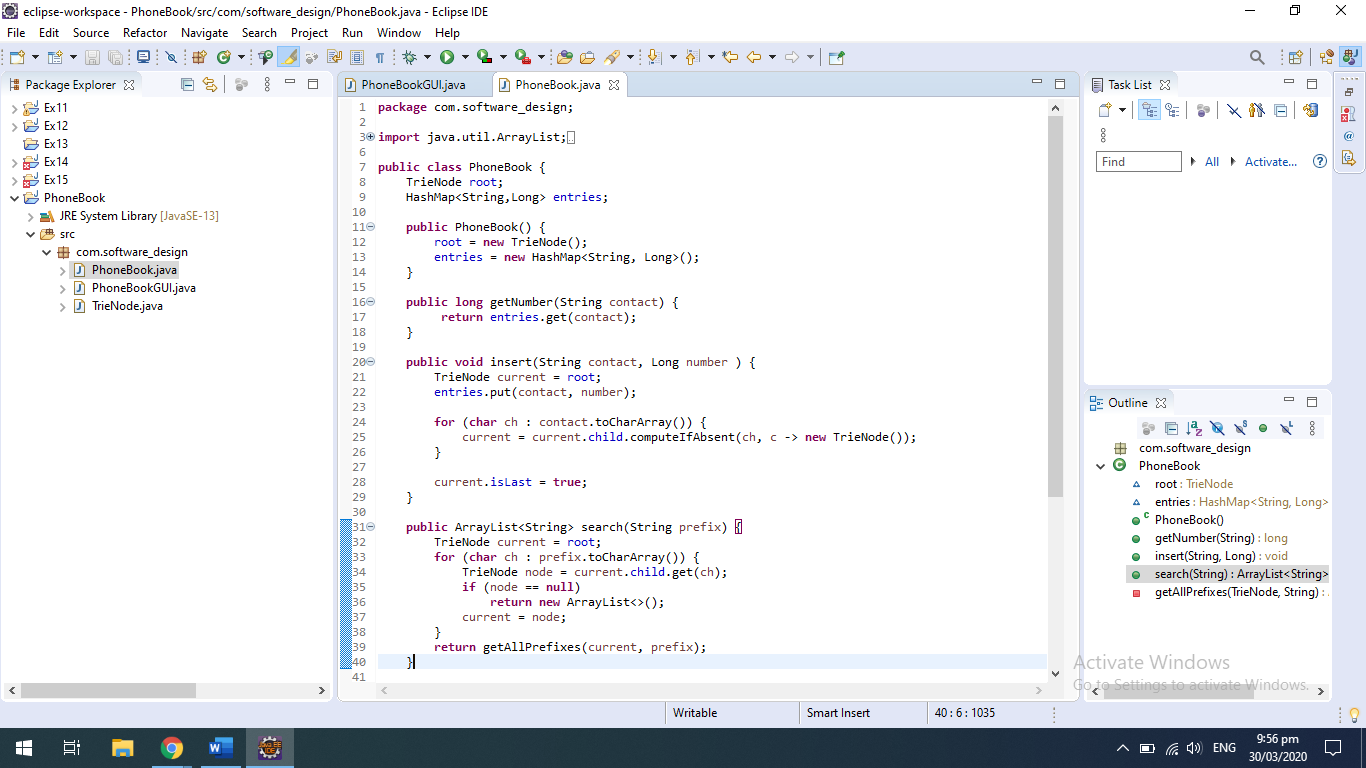
Here are the classes that have been used:

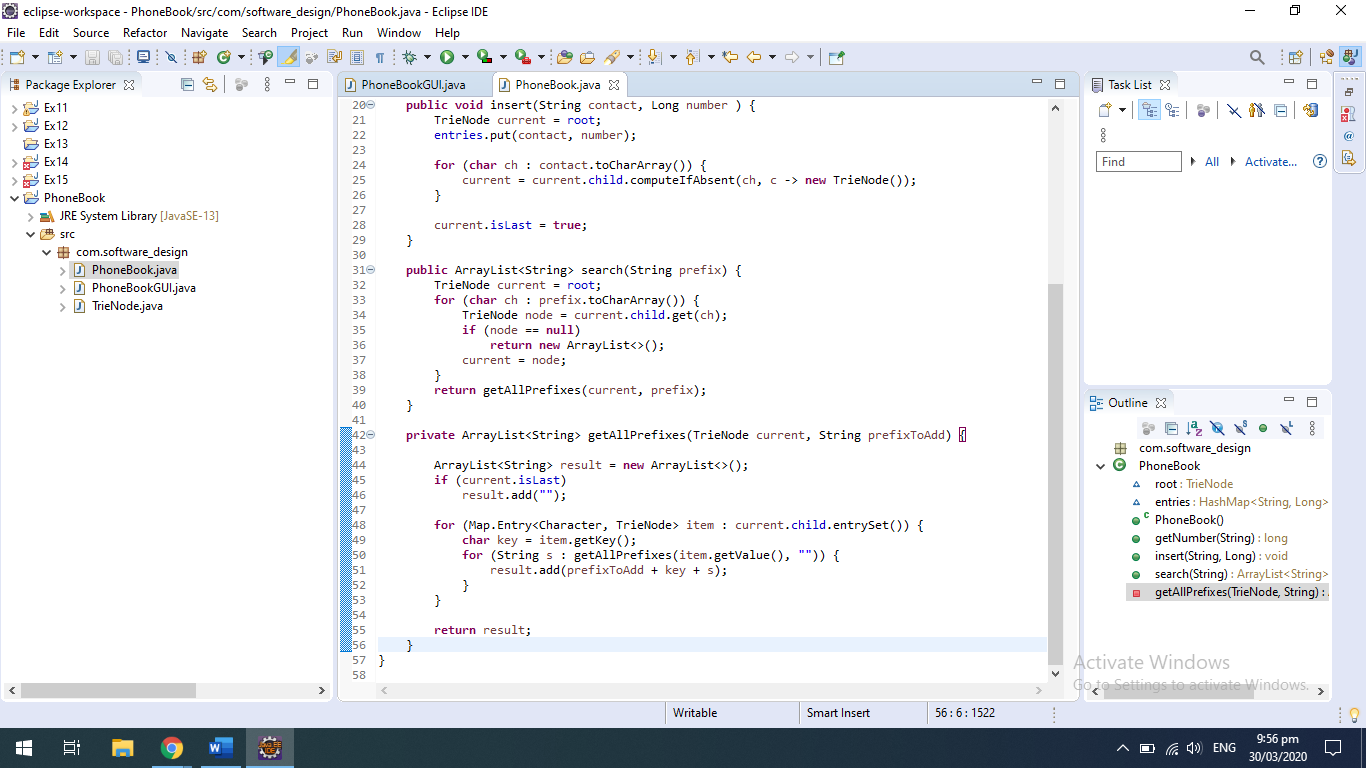
1. PhoneBookGUI is the Swing class used to create the GUI for the directory.



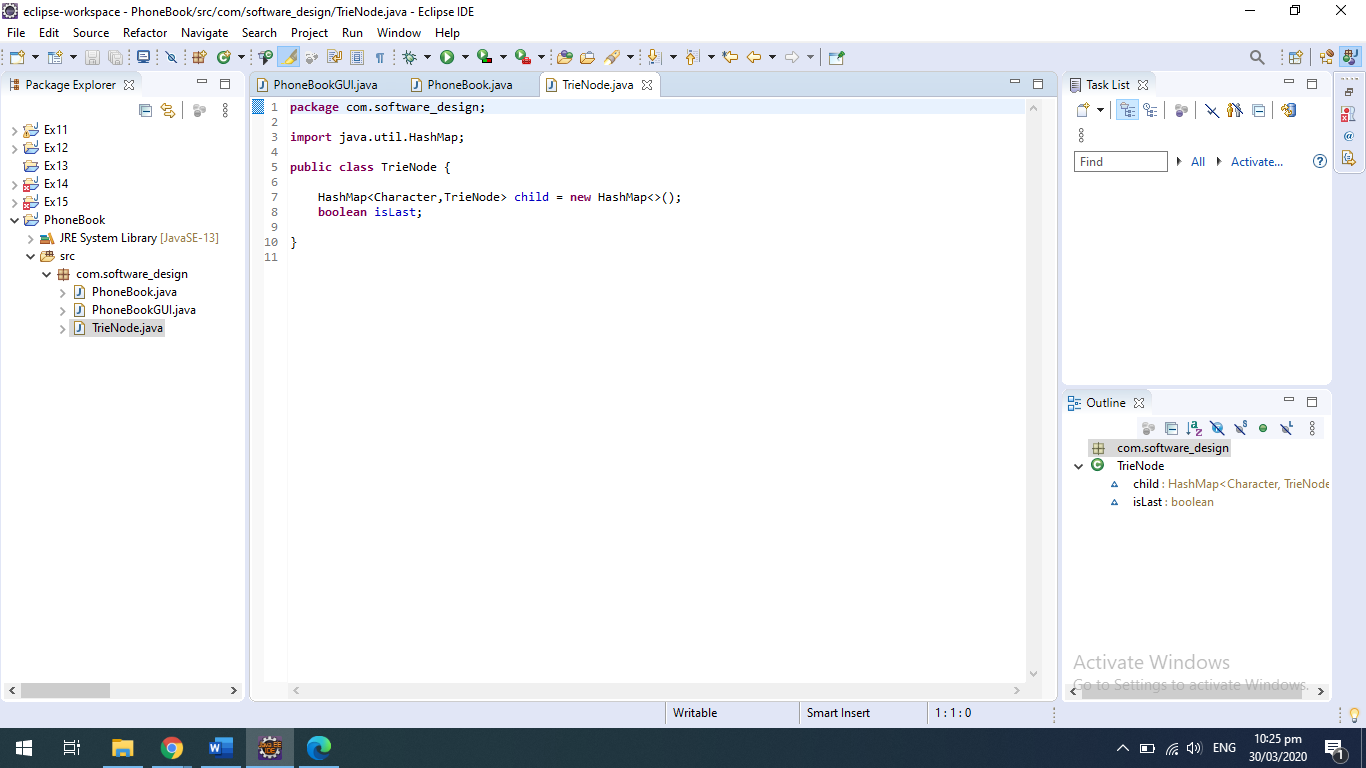


1. Phone Book is where the TRIE and HashMap are initialised and search and find prefix logic is written.



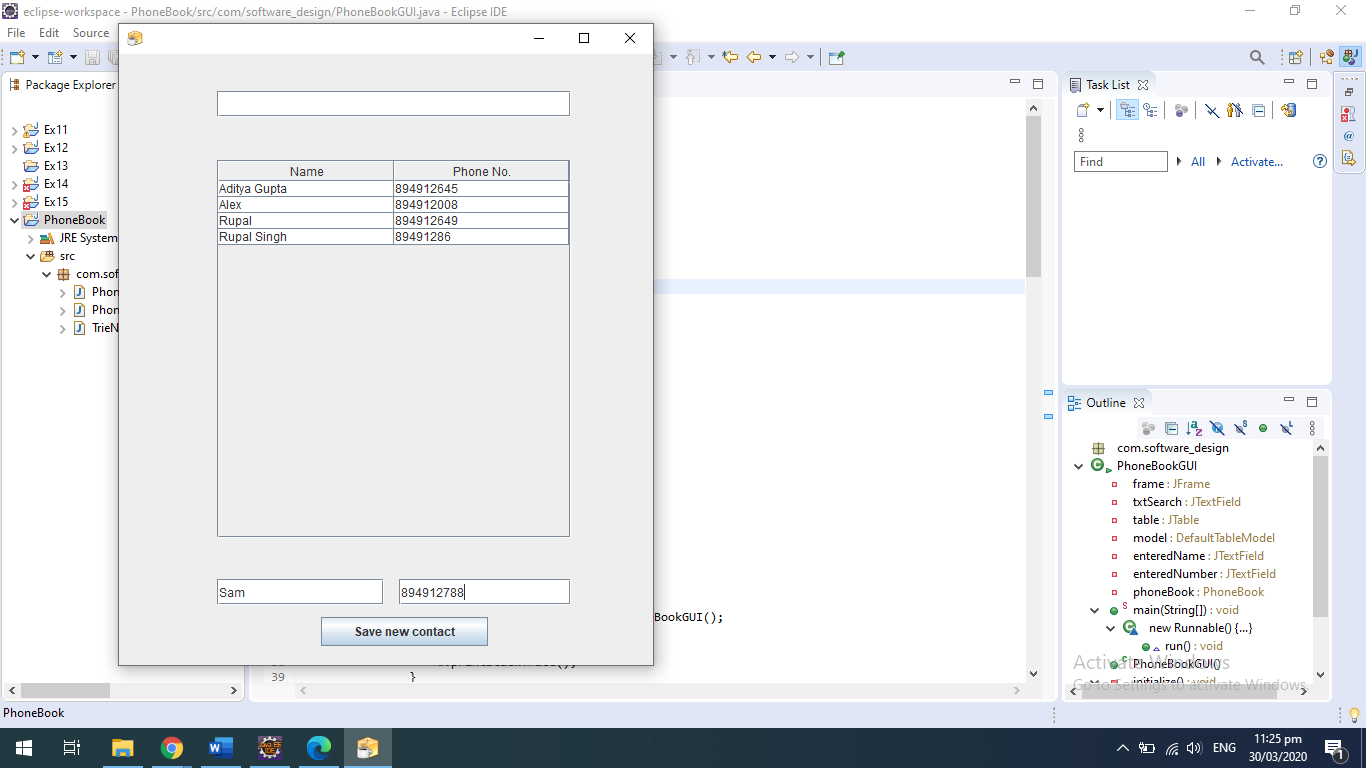


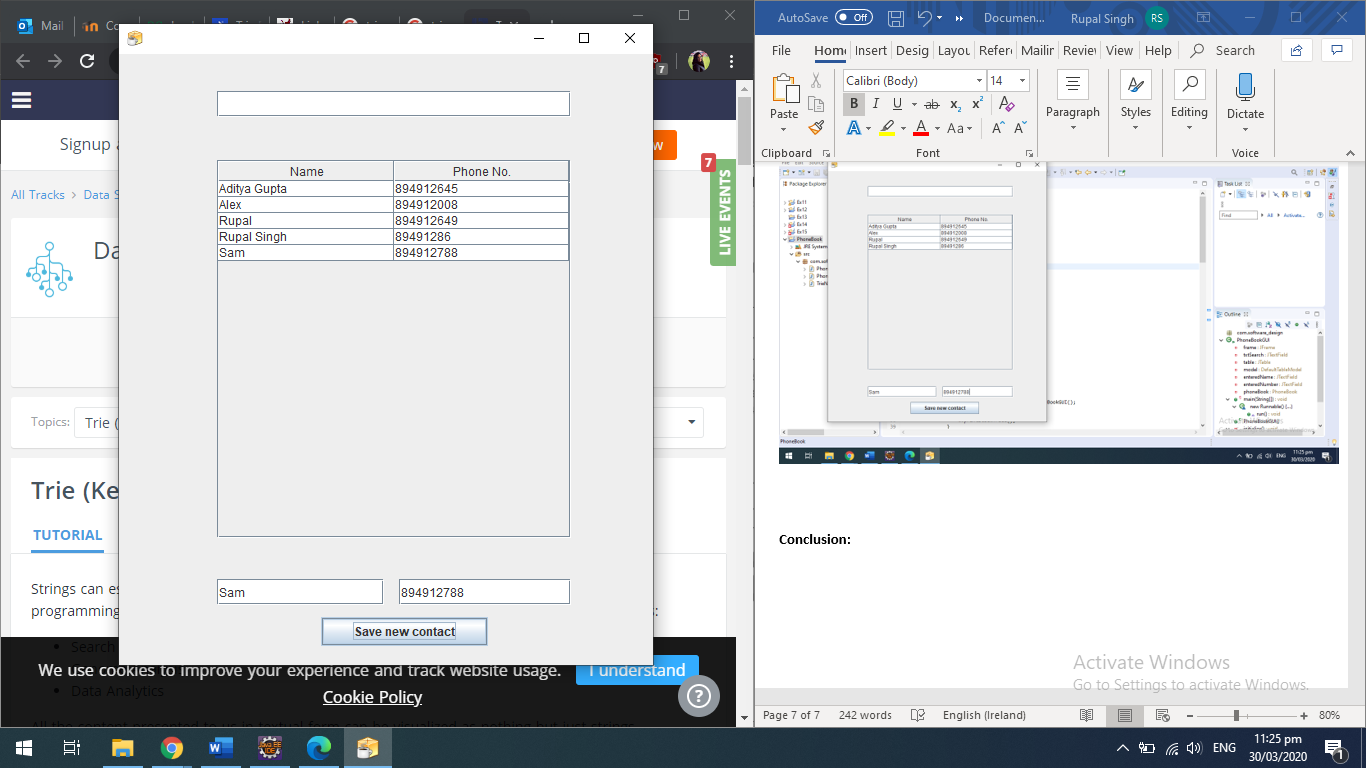
1. Trie Node is the node that has been implemented beforehand.

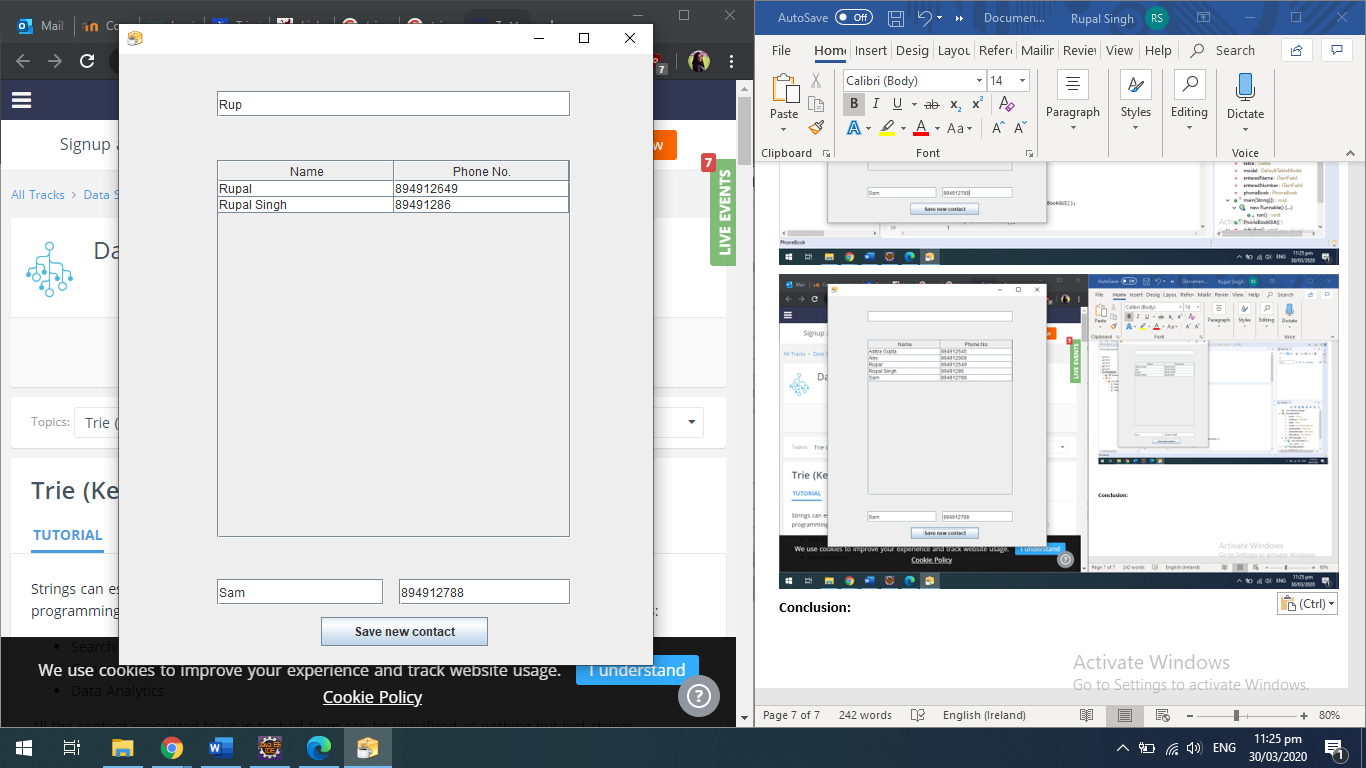


**Screen Shots:**

Following Screen shots show case the phone book UI in action. Data is being inserted and reflected in the table in real time using Trie and HashMap respectively. In order to manipulate recursive function operating on LinkedList has been used.







**Conclusion:**

This was indeed a great learning experience. Even though the sample code provided and the resources available online helped me a lot, I honestly wouldn’t have learnt about Trie Data structure, was it not for this project. It helped me understand complexity better and has piqued my interest enough to get me started to research and understand them better. Thanks to the way our curriculum was structured I didn’t face any issues implementing HashMap and LinkedList. For Tries, I did a lot off digging and later successfully implemented it by using reference from the code I found online.

**References:**

Geeks for geeks: <https://www.geeksforgeeks.org/trie-insert-and-search/>

TopTal: <https://www.toptal.com/java/the-trie-a-neglected-data-structure>

Stack Exchange: <https://cs.stackexchange.com/questions/91053/how-does-a-trie-work>

Wiki: <https://en.wikipedia.org/wiki/Trie>