# Project “Completely”

This is a system that returns autocomplete matches for query strings. The top 10 results that match the query string either as a direct prefix or as a part that immediately follows an underscore are returned. The system also includes functionality to build a dictionary from CSV input as well as write out and read in the prebuilt dictionary to and from JSON files.

Users are provided a command line interface where they can exercise functionality such as:
**add** a scored word
**load** a csv file with word, score pairs
**complete** a prefix
**clear** the dictionary
**write** dictionary to json file
**read** dictionary from json file

## Design

The high level design is a dictionary that maintains a couple of Trie structures; one for the direct matches and a second one for the underscore matches. Each node represents a character and holds its child nodes as well as a list of the top matches. Once the dictionary is built, it is fairly efficient to traverse the tries for a given prefix string and get the matches. For serialization and deserialization, the Jackson library is used to write out and read in JSON files.

## Classes

**CommandInput:** Command line user interface
**Completer:** Interface definition for Completer that defines the “complete,” “add,” and “loadCsv” methods.
**CompleterTrieDict:** The Trie based Completer implementation
**CompleterTrieNode:** Interface for CompleterTrieNode that defines general functionality
**HashTrieNode:** HashMap based implementation of CompleterTrieNode
**ScoredMatch:** Simple class for holding name, score pairs

## General Considerations

A design consideration when thinking about the implementation of CompleterTrieNode is what structure to use to hold the child nodes. A fixed array structure would give a direct access advantage, but it is likely going to be sparse for most nodes. With the HashTrieNode implementation, I opted for a HashMap based approach that is more conservative with memory use.

## Testing

All unit tests are in src/test/CompleterTest and test data in CSVs under resources.

## Libraries

In addition to the standard jdk1.8 libraries, the following libraries are also used:
Jackson 2.9.6
JUnit 5.2