Data Structures:

Hash Table:

* Unordered data structure
* O(1) insertion and retrieval.
* Great when order doesn’t matter and you need to quickly retrieve

Binary Search Tree

* Ordered Data Structure
* O(log n) insertion and retrieval
* Good for when sorting and order is important (i.e. address book, sorted lookup)

Trie

* O(m) lookup where m=length of the string. Can be faster than an imperfect hash table with many collissions. For short keys, they are just as fast or faster than hash tables.
* Can provide alphabetical ordering by keys.
* Great for prefixing into data and are faster than BST.
* Tries can be more space efficient than BST with short keys, since key nodes can be shared.

Java:

Difference between HashMap and Hashtable in Java

* Both are implementation of the Map interface
* HashMap is roughly equivalent to Hashtable, except that it is non synchronized and permits nulls as key values, whereas Hashtable doesn’t allow nulls as keys.
* HashMap is non-synchronized whereas Hashtable is synchronized, which means that Hashtable is thread-safe and can be shared between multiple threads, whereas HashMap must be manually synchronized. Java 5 introduces ConcurrentHashMap for this reason, and provodies better scalability than Hashtable.
* They differ in Iterators. HashMap is a fail-fast iterator, while the enumerator for HashTable is not and throw ConcurrentModificationException if any other thread modifies the map structure by adding or removing any element except Iterator’s own remove() method.
* Because of the threadsafety, Hashtable is much slower than HashMap if used in Single threaded environments. So if you don’t need synchronization, use HashMap.
* HashMap does not guarantee that the order of the map will remain constant over time.