Hash tables in different programming languages

Collection classes are specialized classes for data storage and retrieval. Depending on your application's need, selecting an appropriate data structure can significantly impact your approach to coding and the efficacy of the application. A hashtable will store key-value pairs and offers a quick lookup. Some languages include out-of-the-box implementations, while others require another collection type to act like a hashtable.

What is a hashtable?

A hashtable offers very quick lookups for an application. This is achieved by creating a hashing function that will create an alpha-numeric (letters and numbers) output from a given input. This hash is then used to determine where in memory to store something. This means that when you want to know if an element is in the data structure, instead of looking through every item and making a comparison, you only need to apply the hashing function and see if that item has been hashed to memory. When you consider that a data source might have millions of entries, not having to check every single one is a great time saver.

None of the languages covered in this course have a built-in hashtable implementation. So, in order to implement an instance, it is necessary to alter an existing data structure to perform the operations of a hashtable.

Implementing hashtables in Kotlin

While Kotlin does not have a built-in implementation of a hashtable, a very similar data structure called a hashmap is supported. Hashmaps are very similar to hashtables as they also store key-value pairs and use a hash to determine where in memory to find the key. There are some distinct differences that should be kept note of: a hashmap will allow the use of nulls for keys and values, and it's not thread-safe.

What is thread-safe, and how does it apply to hashtables?

Threads are processes that a computer can run. Typically, when you turn on your computer, a number of processes will begin. These processes include things like starting up a Word document, Excel, a Java application and so on. These processes are often run at the same time. To do this, the compiler will create many threads that can execute the code. So, a thread is a small executable piece of code that can run a process. So how can you say that code is thread-safe? Thread-safe means that if you are to write a program that accesses a data structure, you can duplicate this application and access the same data structure via different threads without causing an error. Having five different threads working on the same data structure might make your code run five times faster, but it is only useful if the information that is being changed is done so correctly.

How does thread-safe relate to the Kotlin implementation of hashtables?

One feature of hashtables is that they can be synchronized. This means that if five different processes are using and changing the same information in a table, the information is always correct. Kotlin does not provide an implementation of hashtables; however, there is a very relatable data structure that can be adapted for this purpose called hashmaps. Hashmaps have the same key, value and hash lookup implementation, but they are not thread-safe. So, in implementing a hashtable in Kotlin, one can take a hashmap and add some code that ensures synchronization so multiple threads can access it at the same time. Additionally, it is important to ensure that the hashtable will not allow nulls to be added as keys or values. This can cause an issue with comparing values within the table.

Hashtable implementation in Python

As with Kotlin, Python does not have a native implementation of a hashtable. In the previous section, it was

demonstrated that a table could be mimicked using an existing structure called a hashmap. This can be done in Python as well, though the underlying structure used is a dictionary. A dictionary is an appropriate data structure to use, as it works on the same principle as a hashtable. Namely, it stores key-value pairs. The keys are hashable and used as an indicator of where in the memory to store the value. This means that it has very fast search and insertion methods. In addition, dictionaries are already thread-safe, so they don't require changing if you require operations to perform concurrently.

Conclusion

In this reading, hashtables were discussed. The implementation of a hashtable is dependent on the language. Here it was demonstrated how a hashmap in Kotlin and a dictionary in Python are sufficiently similar structures from which one can create a hashtable.