Maps

- Maps - (Generic type) This is an interface that describes a key and a value pair. Each key has to be unique and is hashed, this hash value is used as the index of item. This is known as an associated array, as there are basically two arrays one that stores the value and the other stores the key and they are linked together. This can't be ordered, so it is not linear.

Maps are very efficient for lookups, insertions, and deletions. The time complexity for these operations is O(1) on average. However, in the worst case, the time complexity can be O(n), where n is the number of elements in the map.

This hashes the key and uses that to find the index of the value. This is not a continues piece of memory. If there are multiple elements at the same index (known as collisions), there are two ways to handle it:

- Separate chaining: In this approach, each element at the same index is stored in a linked list. This is the approach used by the HashMap class.
- Open addressing: In this approach, the elements are stored in the same array. If there is a collision, the algorithm looks for the next available slot in the array. This is the approach used by the Hashtable class.

Here's an example of using a HashMap in Java:

```java
import java.util.HashMap;

public class HashMapExample {
    public static void main(String[] args) {
        // Create a new HashMap
        HashMap<String, Integer> map = new HashMap<>();

        // Add key-value pairs to the HashMap
        map.put("apple", 10);
        map.put("banana", 5);
        map.put("orange", 7);

        // Retrieve a value based on a key
        int count = map.get("apple");
        System.out.println("Count of apples: " + count);

        // Check if a key exists
        boolean containsKey = map.containsKey("banana");
        System.out.println("Contains banana: " + containsKey);

        // Remove a key-value pair
        map.remove("orange");

        // Iterate over the keys and values in the HashMap
        for (String key : map.keySet()) {
            int value = map.get(key);
```

```java
                System.out.println(key + ": " + value);
            }
        }
    }
```

There are two main types:

HashMap

This is an implementation of map, this works by hashing the key and is used as the index that is stored in the array. This is not synchronized so multiple threads can use it at the same time and can have nulls.

~~~
```java
public class LetsHashMap{
    Map <Integer, String> map = new HashMap<>();
}
```
~~~

HashTable

This is an implementation of map, and is similar to hashmap but it is synchronized so only one threads can use it at a time and can't have nulls.

~~~
```java
public class LetsHashTable{
    Map <Integer, String> map = new HashTable<>();

    map.put(5, "Minas");
    map.get(5)
}
```
~~~

Example of when you would use a map

- You can use a map to store the number of times a word appears in a string. You can use the word as the key and the number of times it appears as the value.
- When you want to store a list of students and their grades, you can use the student's id number as the key and the grades as the value.