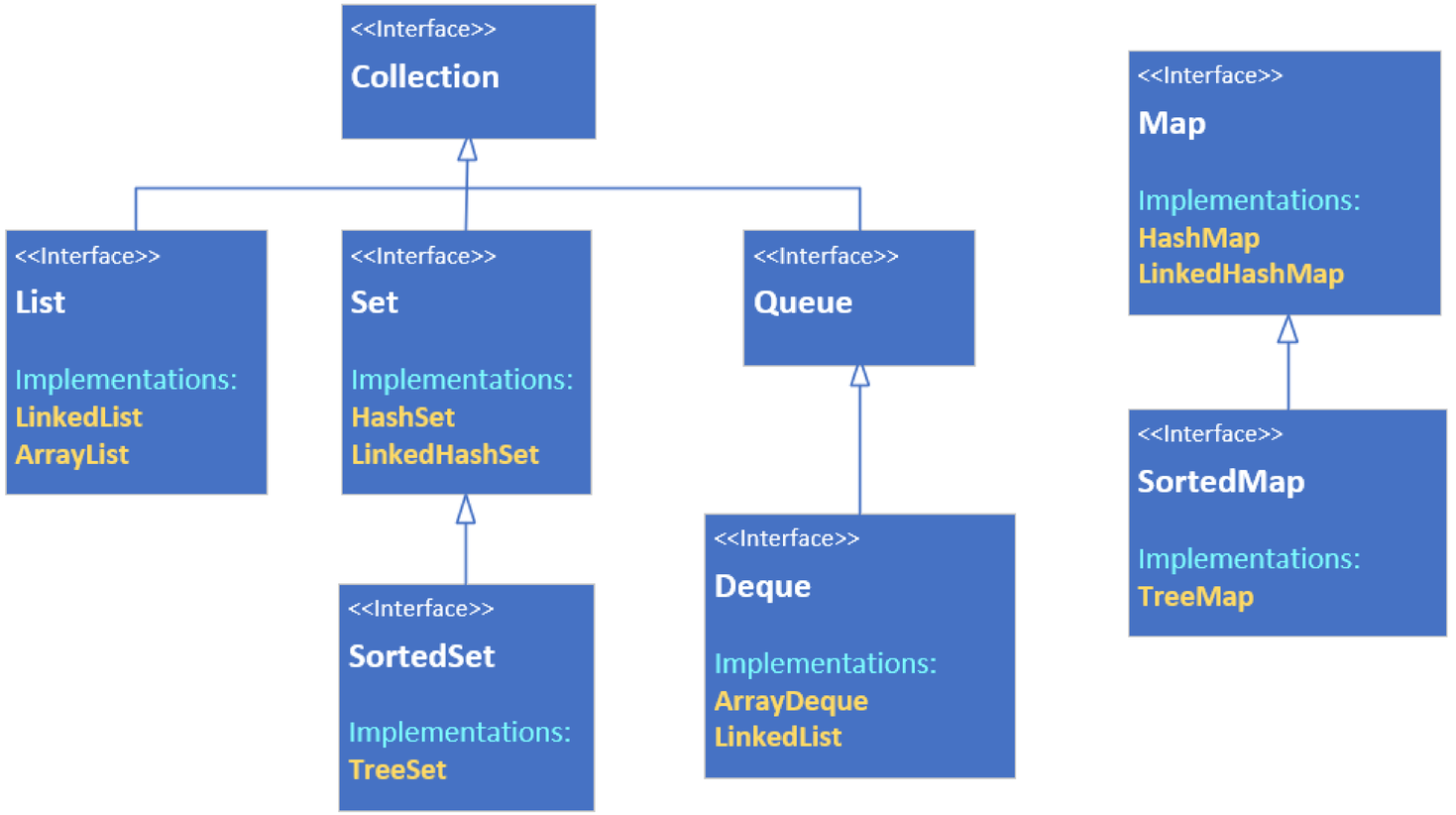
The Map Interface, why is it different?



From this diagram on the slide, you can obviously see the Map is out here on its own.

The Map Interface, why is it different?

A map in the collections framework is another data structure.

Although it's still a grouping of elements, it's different, because elements are stored with keyed references.

This means a Map requires two type arguments, as you can see on this slide, where I'm showing the root interface, Collection, compared to the Map interface.

Collection Interface	Map Interface
`interface Collection<E> extends Iterable<E>`	`interface Map<K, V>`

The Map has K for it's key type, and V for the value type.

As with any generic classes, the only restriction on these types is, they must be reference types, and not primitives.

Map characteristics

A Java Map can't contain duplicate keys.

Each key can only map to a single value.

Map Implementations (the classes that implement Map)

In the next few lectures, I'll be looking at 3 of the Java classes that implement the map interface, the **HashMap**, the **LinkedHashMap**, and the **TreeMap**.

The HashMap is unordered, the LinkedHashMap is ordered by insertion order, and the TreeMap is a sorted map.