- A stack is a collection of elements with “last-in, first-out” retrieval.
- A queue is a collection of elements with “first-in, first-out” retrieval
- When removing an element from a priority queue, the element with the most urgent priority is retrieved.

An Overview of the Collections Framework

- A collection groups together elements and allows them to be retrieved later.
- Java collections framework: a hierarchy of interface types and classes for collecting objects.
  - Each interface type is implemented by one or more classes

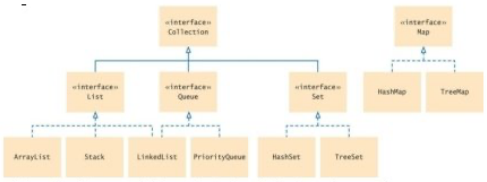

Figure 1 Interfaces and Classes in the Java Collections Framework

- The Collectioninterface is at the root
  - All Collectionclass implement this interface
  - So all have a common set of methods

- A stack can be used to check whether parentheses in an expression are balanced
- Use a stack to evaluate expressions in reverse Polish notation.
- Using two stacks, you can evaluate expressions in standard algebraic notation.
- Use a stack to remember choices you haven’t yet made so that you can backtrack to them.
- The HashSet and TreeSet classes both implement the Set interface.
- Set implementations arrange the elements so that they can locate them quickly.
- You can form hash sets holding objects of type String, Integer, Double, Point, Rectangle, or Color.
- You can form tree sets for any class that implements the Comparable interface, such as String or Integer.
-
- Sets don’t have duplicates. Adding a duplicate of an element that is already present is ignored.
- A set iterator visits the elements in the order in which the set implementation keeps them.
- You cannot add an element to a set at an iterator position..
-

Working with Maps

Table 5 Working with Maps

`Map<String, Integer> scores;`	Keys are strings, values are Integer wrappers. Use the interface type for variable declarations.
`scores = new TreeMap<>();`	Use a `HashMap` if you don't need to visit the keys in sorted order.
`scores.put("Harry", 90);` `scores.put("Sally", 95);`	Adds keys and values to the map.
`scores.put("Sally", 100);`	Modifies the value of an existing key.
`int s = scores.get("Sally");` `Integer n2 = scores.get("Diane");`	Gets the value associated with a key, or `null` if the key is not present. `s` is 100, `n2` is `null`.
`System.out.println(scores);`	Prints `scores.toString()`, a string of the form `{Harry=90, Sally=100}`

`for (String key : scores.keySet())`
`{`
`  Integer value = scores.get(key);`
`  . . .`
`}`
	Iterates through all map keys and values.

`scores.remove("Sally");`	Removes the key and value.