
Australian
National
University
Collections
J13
Collections

The Collection Framework

- Interfaces
  - Implementation-agnostic interfaces for collections
- Implementations
  - Concrete implementations
- Algorithms
  - Searching, sorting, etc

Using the framework saves writing your own: better performance, fewer bugs, less work, etc.

The Collection Interface

- Basic operators
  - `size`, `isEmpty()`, `contains()`, `add()`, `remove()`
- Traversal
  - for-each, and iterators
- Bulk operators
  - `containsAll()`, `addAll()`, `removeAll()`, `retainAll()`, `clear()`
- Array operators
  - convert to and from arrays

Collection Types

- Primary collection types:
  - Set (no duplicates, mathematical set)
  - List (ordered elements)
  - Queue (shared work queues)
  - Map (<key, value> pairs)
- Each collection type is defined as an interface
  - You need to choose a concrete collection
  - Your choice will depend on your needs

Concrete Collection Types

	Implemented Using				
Interfaces	**Hash table**	**Resizable array**	**Tree**	**Linked list**	**Hash table + linked list**
Set	`HashSet`		`TreeSet`		`LinkedHashSet`
List		`ArrayList`		`LinkedList`	
Queue				`LinkedList`	`LinkedHashMap`
Map	`HashMap`		`TreeMap`		

Based on table from http://docs.oracle.com/javase/tutorial/collections/implementations/index.html

Four Commonly Used Collection Types

- `HashSet` implements a **set** as a hash table
  - Makes no ordering guarantees
- `ArrayList` implements a **list** using an array
  - Very fast access
- `HashMap` implements a **map** using a hash table
  - Makes no ordering guarantees
- `LinkedList` implements a **queue** using a linked list
  - First-in-first-out (FIFO) queue ordering