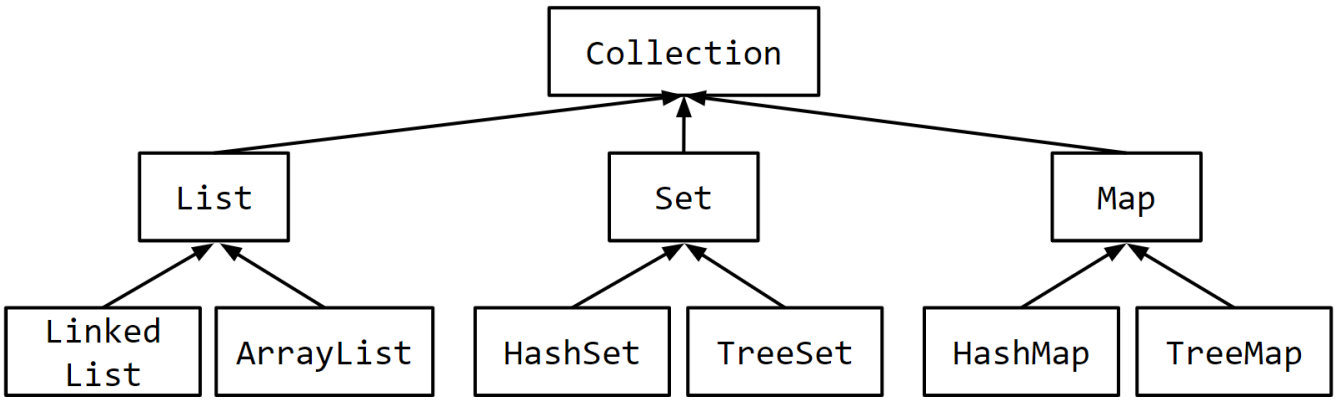
Map, Set, List Reference Sheet, Page 1

```
public interface Map<K, V> ... { ...
   public boolean containsKey(K key)
   public V get(K key)
   public V getOrDefault(K key, V value)
   public void put(K key, V value)
   public Set<K> keySet()
   public Iterator<K> iterator()
   public int size()
}

public interface Set<K> ... { ...
   public boolean contains(K key)
   public void add(K key)
   public Set<K> keySet()
   public Iterator<K> iterator()
   public int size()
}

public interface List<T> ... { ...
   public boolean contains(T item)
   public void add(T item)
   public void add(int index, T item)
   public T get(int i)
   public T set(int i, T item)
   public int indexOf(Object o)
   public boolean remove(Object o)
   public Iterator<T> iterator()
   public int size()
}
```

Implementations:



Other handy data types:

```
public class Stack<T> ... { ...
   public T pop()
   public void push(T item)
   public Iterator<T> iterator()
   public int size()
}

public class Queue<T> ... { ...
   public T dequeue()
   public void enqueue(T item)
   public Iterator<T> iterator()
   public int size()
}

public class MinPQ<T> ... { ...
   public T MinPQ()
   public T MinPQ(Comparator<T> c)
   public T removeSmallest()
   public T smallest()
   public void add(T item)
   public Iterator<T> iterator()
   public int size()
}
```

Iterator, Iterable, Comparator, Comparable:

```
public interface Iterator<T> {
    boolean hasNext();
    T next();
}

public interface Iterable<T> {
    Iterator<T> iterator();
}

public interface Comparator<T> {
     int compare(T o1, T o2);
}

public interface Comparable<T> {
     int compareTo(T obj);
}
```