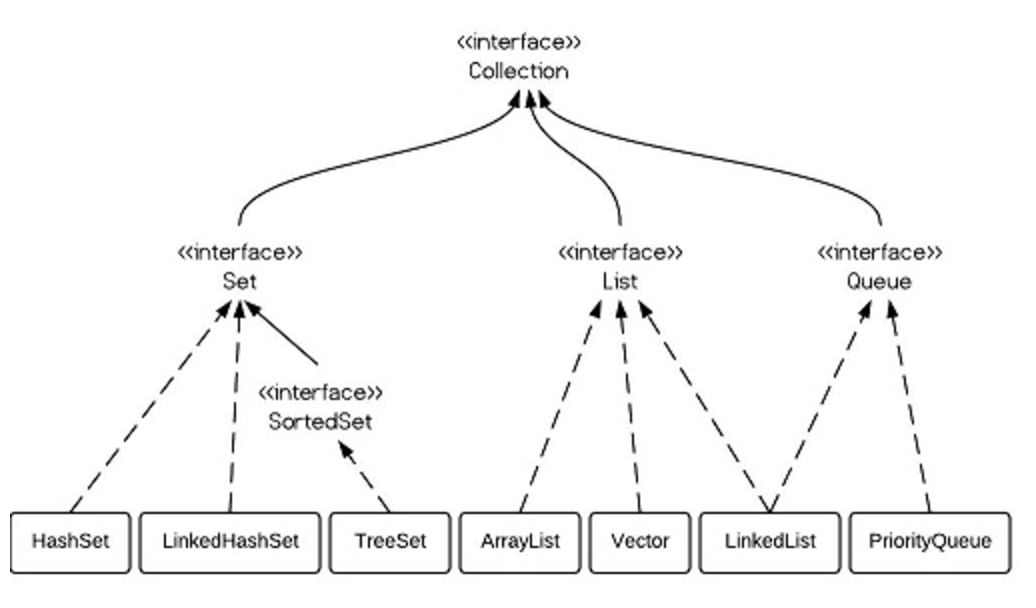
Array



They all implement List interface and their methods and results are almost identical.

ArrayList

Arraylist use array as its base data structure which makes it faster for searching, O(1) get using index. But add and remove will cost O(n) in the worst time since array need to shift elements to fill space created by removed element. It consumes less space than linkedlist due to it just maintain indexes and data.

Extends AbstractList, implements List, RandomAccess, Cloneable and java.io.Serializable interfaces.

Integer value = null;

Iterator iter = list.iterator();

While(iter.hasNext()){

Value = (Integer) iter.next();

}

Integer value = null;

int size = list.size();

for (int i=0; i<size; i++) {

value = (Integer)list.get(i);

}

Integer value = null;

for (Integer integ:list) {

value = integ;

}

Object[] toArray()

<T> T[] toArray(T[] contents)

public static Integer[] vectorToArray2(ArrayList<Integer> v) {

Integer[] newText = (Integer[])v.toArray(new Integer[0]);

return newText;

}

LinkedList

Linkedlist use doubly linked list as its as data structure which makes it slower for searching, that in the worst case need to traversal all elements to find the target. But add and remove will cost O(1) in the worst time since linkedlist just need to change the two pointers. Besides, it will cost more spaces than arraylist because it maintains element data and two pointers for neighbor nodes.

Vector

Vector is almost the same as arraylist. The difference is vector is thread-safe. Implement randomaccess.