**HashSet, TreeSet Classes**

The Collections Framework provides two general-purpose implementations of the Set interface: HashSet and TreeSet. More often than not, you will use a HashSet for storing your duplicate-free collection. For efficiency, objects added to a HashSet need to implement the hashCode() method in a manner that properly distributes the hash codes. While most system classes override the default [hashCode()](http://java.sun.com/products/jdk/1.2/docs/api/java/lang/Object.html#hashCode%28%29) implementation in Object, when creating your own classes to add to a HashSet remember to override hashCode(). The TreeSet implementation is useful when you need to extract elements from a collection in a sorted manner.

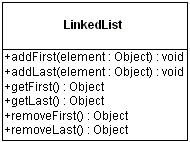
##### List Interface

The List interface extends the Collection interface to define an ordered collection, permitting duplicates. The interface adds position-oriented operations, as well as the ability to work with just a part of the list.

###### ArrayList, LinkedList Classes

There are two general-purpose List implementations in the Collections Framework: ArrayList and LinkedList. Which of the two List implementations you use depends on your specific needs. If you need to support random access, without inserting or removing elements from any place other than the end, than ArrayList offers the optimal collection. If, however, you need to frequently add and remove elements from the middle of the list and only access the list elements sequentially then LinkedList offers the better implementation.

Both ArrayList and LinkedList implement the Cloneable interface. In addition, LinkedList adds several methods for working with the elements at the ends of the list (only the new methods are shown in the following diagram):



import java.util.\*;

public class ListExample {

public static void main(String args[]) {

List list = new ArrayList();

list.add("Bernadine");

list.add("Elizabeth");

list.add("Gene");

list.add("Elizabeth");

list.add("Clara");

System.out.println(list);

System.out.println("2: " + list.get(2));

System.out.println("0: " + list.get(0));

LinkedList queue = new LinkedList();

queue.addFirst("Bernadine");

queue.addFirst("Elizabeth");

queue.addFirst("Gene");

queue.addFirst("Elizabeth");

queue.addFirst("Clara");

System.out.println(queue);

queue.removeLast();

queue.removeLast();

System.out.println(queue);

}

}

###### HashMap, TreeMap Classes

The Collections Framework provides two general-purpose Map implementations: HashMap and TreeMap . As with all the concrete implementations, which implementation you use depends on your specific needs. For inserting, deleting, and locating elements in a Map, the HashMap offers the best alternative. If, however, you need to traverse the keys in a sorted order, then TreeMap is your better alternative.