List集合：有序，可重复

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **List** | **底层数据结构** | **查询** | **增删** | **效率** | **线程** |
| ArrayList | 动态数组 | 快 | 慢 | 高 | 不安全 |
| LinkedList | 链表 | 慢 | 快 | 高 | 不安全 |
| Vector | 数组 | 快 | 慢 | 低 | 安全 |

Set集合：无序，不可重复/唯一

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Set** | **底层数据结构** | **排序** |  | **效率** | **线程** |
| HashSet | 哈希表 | 无序 |  | 高 | 不安全 |
| LinkedHashSet | 链表和哈希表共同实现 | 有序 |  | 高 | 不安全 |
| TreeSet | 二叉树（红黑树） | 有序 |  | 低 | 安全 |

LinkedHashSet:底层数据结构采用链表和哈希表共同实现，链表保证了元素的顺序与存储顺序一致，哈希表保证了元素的唯一性。线程不安全，效率高。

TreeSet：底层数据结构采用二叉树来实现，元素唯一且已经排好序；唯一性同样需要重写hashCode和equals()方法，二叉树结构保证了元素的有序性

Set和List对比：   
Set：检索元素效率低下，删除和插入效率高，插入和删除不会引起元素位置改变。   
List：和数组类似，List可以动态增长，查找元素效率高，插入删除元素效率低，因为会引起其他元素位置改变。

Map用于保存具有映射关系的数据，Map里保存着两组数据：key和value，它们都可以使任何引用类型的数据，但key不能重复。所以通过指定的key就可以取出对应的value。

无序的Map

HashMap：底层数据结构采用哈希表来实现，是线程不安全的，效率高。最多一个Key为null。Value可以多个位Null

Hashtable：底层数据结构也采用哈希表来实现，是线程安全的，效率低。不能放入null。

有序的Map

TreeMap