### Java集合类架构层次关系

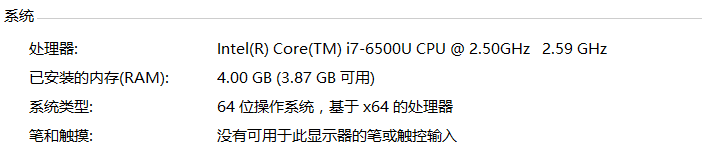
### 262238192165666

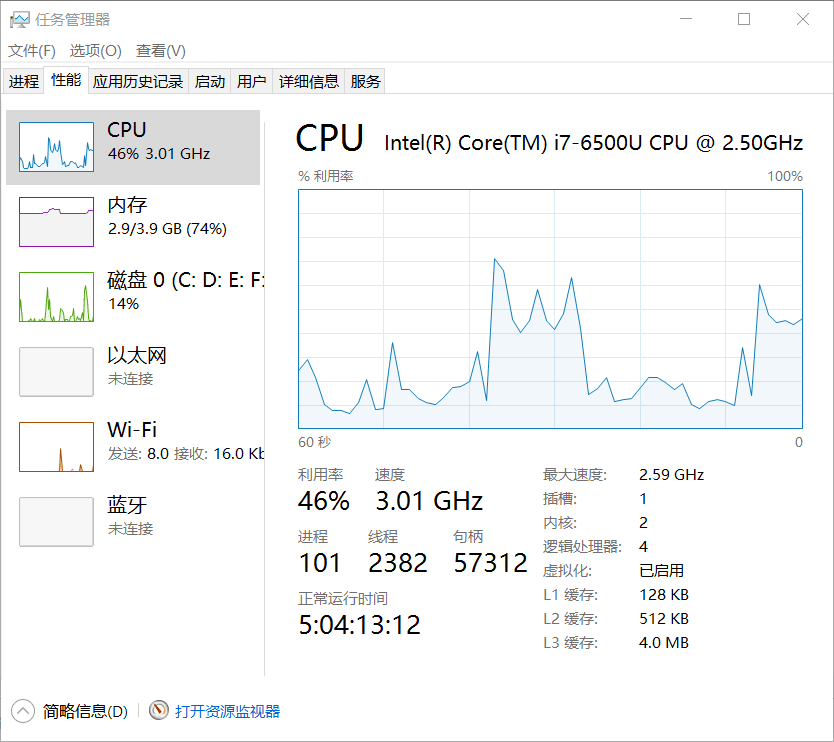
### 二、集合类型描述

|  |  |  |
| --- | --- | --- |
| **集合类** | **描述** | **使用说明** |
| ArrayList | 可以动态增长和缩减的索引序列 | 长于随机访问元素；但插入、删除元素较慢（数组特性） |
| LinkedList | 可以在任何位置进行高效的插入和删除操作的有序序列 | 插入、删除元素较快，但随机访问较慢（链表特性） |
| Vector | 一种老的动态数组，线程同步，效率很低 |  |
| HashSet | 没有重复元素的无序集合 | 底层HashMap:数组+链表  使用散列，最快的获取元素方法 |
| TreeSet | 可以确保集合元素处于排序状态的集合 | 底层红黑树实现  将元素存储在红-黑树数据结构中，默认为升序 |
| HashMap | 存储键/值关联的数据结构 | 数组+链表的结合体—哈希表  使用散列，提供最快的查找技术 |
| TreeMap | 键值有序排列的映射表 | 底层数据结构是红黑树  默认按照比较结果的升序保存键 |

### 三、性能测试

**配置**





1. **添加性能测试**

1)数据（x轴表示数据量，y轴时间为平均时间）

注：最大只能进行到10000，否则统计时间过长，作出的图表不理想（下同）。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 10 | 50 | 100 | 500 | 1000 | 2000 | 4000 | 5000 | 6000 | 8000 | 10000 |
| ArrayList | 82 | 21 | 11 | 9 | 6 | 10 | 10 | 8 | 9 | 8 | 7 |
| Linkedlist | 65 | 24 | 8 | 10 | 18 | 17 | 16 | 12 | 12 | 14 | 18 |
| Vector | 67 | 7 | 8 | 10 | 10 | 8 | 7 | 12 | 7 | 8 | 10 |
| TreeSet | 333 | 50 | 59 | 65 | 52 | 60 | 59 | 71 | 64 | 63 | 78 |
| HashSet | 200 | 18 | 13 | 15 | 13 | 14 | 14 | 16 | 14 | 14 | 16 |
| TreeMap(put) | 269 | 44 | 43 | 64 | 56 | 64 | 66 | 72 | 64 | 67 | 79 |
| HashMap(put) | 175 | 15 | 15 | 33 | 19 | 17 | 17 | 22 | 19 | 19 | 21 |

2)数据->散点图

1. **查找性能测试**

1)数据（x轴表示数据量，y轴时间为平均时间）

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 10 | 50 | 100 | 500 | 1000 | 2000 | 4000 | 5000 | 6000 | 8000 | 10000 |  |
| ArrayList | 16 | 15 | 15 | 14 | 13 | 13 | 14 | 13 | 13 | 18 | 13 |  |
| Linkedlist | 31 | 35 | 41 | 166 | 342 | 748 | 1615 | 1990 | 2542 | 3271 | 4197 |  |
| Vector | 67 | 7 | 8 | 10 | 10 | 14 | 15 | 12 | 15 | 15 | 10 |  |
| TreeSet(contains) | 165 | 30 | 35 | 41 | 47 | 53 | 60 | 67 | 60 | 63 | 67 |  |
| HashSet(contains) | 76 | 4 | 4 | 7 | 5 | 5 | 6 | 6 | 6 | 6 | 6 |  |
| TreeMap | 111 | 20 | 28 | 37 | 47 | 55 | 57 | 65 | 59 | 60 | 67 |  |
| HashMap | 88 | 3 | 3 | 6 | 6 | 5 | 6 | 7 | 5 | 6 | 6 |  |

2)数据->散点图

1. **LinkedList与ArrayList插入删除性能测试**

1)数据（x轴表示数据量，y轴时间为平均时间）

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 10 | 50 | 100 | 500 | 1000 | 2000 | 4000 | 5000 | 6000 | 8000 | 10000 |
| ArrayList（insert） | 222 | 211 | 215 | 250 | 218 | 440 | 477 | 438 | 477 | 930 | 934 |
| Linkedlist（insert） | 152 | 58 | 36 | 43 | 44 | 74 | 51 | 45 | 53 | 40 | 55 |
| ArrayList（remove） | 139 | 40 | 22 | 34 | 55 | 95 | 169 | 208 | 255 | 336 | 429 |
| Linkedlist（remove） | 116 | 25 | 17 | 18 | 16 | 26 | 18 | 23 | 27 | 14 | 17 |

2)数据->散点图

**总结：**

（1）LinkedList的底层实现是双向链表，查找性能随着链表容量的增大而下降。

（2）比较来看，如果要进行大量的随机访问，使用ArrayList；如果经常进行插入与删除操作，则使用LinkedList。

（3）HashSet提供最快的查询速度，而TreeSet保持元素处于排序状态。总体情况上来看，HashSet的性能要优于TreeSet。因为TreeSet有排序的功能，它可以维持集合内元素的排序状态。

（4）TreeMap与HashMap相比，前者通常较慢。因为TreeMap需要保证元素的顺序，所以HashMap是日常开发的首选。

（5）程序中不应该使用过时的Vector\Hashtable\Stack。