

学习目标

- * 能够掌握Map子类LinkedHashMap
 - * 能够掌握Map子类TreeMap
 - * 能够理解Set接口概述
 - * 能够通过阅读HashSet源码理解它底层实现就是HashMap的key集合
 - * 能够通过阅读LinkedHashSet源码理解它底层实现就是LinkedHashMap的key集合
 - * 能够通过阅读TreeSet源码理解它底层实现就是TreeMap的key集合
-

* 能够掌握Map子类LinkedHashMap

* 概述：

- * LinkedHashMap是继承于HashMap，是基于HashMap和双向链表来实现的
- * HashMap无序而LinkedHashMap有序，可分为插入顺序和访问顺序两种
- * LinkedHashMap存取数据，还是跟HashMap一样使用的Entry[]的方式，双向链表只是为了保证顺序
- * LinkedHashMap是线程不安全的

```
1 * HashMap是存放无序而LinkedHashMap是存放有序的（插入顺序）
2 public static void main(String[] args) {
3     Map<String,String> map1=new HashMap<String,String>();
4     map1.put("name", "xiaohai");
5     map1.put("age", "18");
6     map1.put("sex", "男");
7     // 存放是无序
8     Set<Entry<String, String>> entries = map1.entrySet();
9     for(Entry<String, String> entry:entries) {
10         System.out.printf("%s %s ",entry.getKey(),entry.getValue());
11     }
12     System.out.println();
```

```

13     Map<String,String> map2=new LinkedHashMap<String,String>();
14     map2.put("name", "xiaohei");
15     map2.put("age", "18");
16     map2.put("sex", "男");
17     // 存放是有序
18     entries = map2.entrySet();
19     for(Entry<String, String> entry:entries) {
20         System.out.printf("%s %s ",entry.getKey(),entry.getValue());
21     }
22 }
23 代码结果:
24 sex 男 name xiaohei age 18
25 name xiaohei age 18 sex 男
26
27 * 访问顺序
28     public static void main(String[] args) {
29         Map<String,String> map=new LinkedHashMap<String,String>(16, 0.75f, true);
30         for (int i = 0; i < 10; i++) {
31             map.put(String.valueOf(i), String.valueOf(i));
32         }
33         // 访问置顶
34         map.get("6");
35         Set<Entry<String, String>> entries = map.entrySet();
36         for(Entry<String, String> entry:entries) {
37             System.out.printf("%s ",entry.getValue());
38         }
39     }
40 结果:
41     0 1 2 3 4 5 7 8 9 6
42 备注: 通过例子可以看出, 最近经常使用的元素就放在后面, 最近最少使用的就排在了链表的前
43 可断点调试: 查看结果
44
45 * 通过LinkedHashMap (顺序访问的特点)
46 * 编写简单一个LRU (Least Recently Used) 最近最少使用缓存
47 import java.util.LinkedHashMap;
48 public class LruCache<K,V> extends LinkedHashMap<K, V> {
49     private static final long serialVersionUID = 14353245324L;
50     private int maxSize;// 最大可以缓存多少个
51     public LruCache(int maxSize){
52         super(16, 0.75f, true);

```

```

53         this.maxSize=maxSize;
54     }
55     //重写删除最老元素的方法
56     @Override
57     protected boolean removeEldestEntry(java.util.Map.Entry<K, V> eldest) {
58         // 当集合的大小超过缓存最大的值，删除最老的元素
59         return size()>maxSize;
60     }
61 }
62
63 public static void main(String[] args) {
64     LruCache<String, String> lruCache=new LruCache<String,String>(8);
65     for (int i = 0; i < 10; i++) {
66         lruCache.put(String.valueOf(i), String.valueOf(i));
67     }
68     lruCache.get("5");
69     Set<Entry<String, String>> entries = lruCache.entrySet();
70     for(Entry<String, String> entry:entries) {
71         System.out.printf("%s ",entry.getValue());
72     }
73
74 }

```

75 结果:

```

76     2 3 4 6 7 8 9 5

```

77 断点调试

78 * 分析removeEldestEntry源码调用过程

```

79 * void afterNodeInsertion(boolean evict) { // possibly remove eldest
80     LinkedHashMap.Entry<K,V> first;
81     if (evict && (first = head) != null && removeEldestEntry(first)) {
82         K key = first.key;
83         removeNode(hash(key), key, null, false, true);
84     }
85 }

```

86 * afterNodeInsertion 这个方法是put的的调用到

87

88 * LruCache案例的编写

89 * 提前感受一下IO流和网络编程

```

90 public static void main(String[] args) throws Exception {
91     // https://www.jd.com:e1d84f4301e444a3db82c908f29947b1
92     // https://www.taobao.com:1b263ad2261782bddac1d35c0528d40e

```

```

93 // https://www.baidu.com:f9751de431104b125f48dd79cc55822a
94 LruCache<String, String> cache=new LruCache<String,String>(2);
95 Scanner input=new Scanner(System.in);
96 String path="";
97 System.out.println("请输入url的地址(推出请输入n): ");
98 path=input.next();
99 while(!("n".equals(path))) {
100     String key=MD5Utils.md5(path);
101     String content=null;
102     if(cache.containsKey(key)) {
103         // 缓存中已经存在, 直接从缓存中
104         content = cache.get(key);
105         System.out.println(content);
106         System.out.println("从缓存中获取");
107     }else {
108         //从网络获取内容
109         content = getNetContent(path);
110         cache.put(key, content);
111         System.out.println(content);
112         System.out.println("从网络中获取");
113     }
114     Set<Entry<String, String>> entries = cache.entrySet();
115     for(Entry entry:entries) {
116         System.out.printf("key=%s ",entry.getKey());
117     }
118     System.out.println();
119     System.out.println("请输入url的地址(推出请输入n): ");
120     path=input.next();
121 }
122 System.out.println("程序结束...");
123 }
124
125 private static String getNetContent(String path) throws MalformedURLException {
126     // 构建链接
127     URL url=new URL(path);
128     HttpURLConnection connection = (HttpURLConnection) url.openConnection();
129     // 链接起来
130     connection.connect();
131     // 获得链接的输入流
132     InputStream is = connection.getInputStream();

```

```

133 // 把输入流转换成字符成
134 byte[] buffer=new byte[1024];
135 int len=0;
136 StringBuilder sb=new StringBuilder();
137 while((len=is.read(buffer))!=-1) {
138     String value=new String(buffer,0,len,Charset.forName("UTF-8"));
139     sb.append(value);
140 }
141 String content=sb.toString();
142 is.close();
143 return content;
144 }
145

```

* 能够掌握Map子类TreeMap

* TreeMap概述

* 底层用黑红树实现的

* TreeMap中的元素默认按照keys的自然排序排列

* 使用TreeMap前提

* 要么Key的类实现了Comparable接口

* 假如Key的类没有实现Comparable接口，就需要使用Comparator比较器

```

1 * Integer: 数字的升序，已经实现了Comparable接口
2 * String: 按照字母表排序，已经实现了Comparable接口
3 public static void main(String[] args) {
4     // Integer
5     Map<Integer,String> treeMap=new TreeMap<Integer,String>();
6     treeMap.put(2, "关羽");
7     treeMap.put(1, "刘备");
8     treeMap.put(3, "张飞");
9     Set<Entry<Integer, String>> entries = treeMap.entrySet();
10    for(Entry<Integer, String> entry:entries) {
11        System.out.printf("%s %s ",entry.getKey(),entry.getValue());
12    }

```

```

13     System.out.println();
14     // String
15     Map<String,String> treeMap1=new TreeMap<String,String>();
16     treeMap1.put("B", "关羽");
17     treeMap1.put("A", "刘备");
18     treeMap1.put("C", "张飞");
19     Set<Entry<String, String>> entries1 = treeMap1.entrySet();
20     for(Entry<String, String> entry:entries1) {
21         System.out.printf("%s %s ",entry.getKey(),entry.getValue());
22     }
23 }

```

24 结果:

25 1 刘备 2 关羽 3 张飞

26 A 刘备 B 关羽 C 张飞

27

28 * 根据User的Id排序购物车

```

29 public class User {
30     private int id;
31     private String name;
32     private String password;
33     ...
34 }
35 public class Goods {
36     private int id;
37     private String name;
38     private double price;
39     ...
40 }
41 public class Cart {
42     private int userId;
43     private List<Goods> cartList;
44     ...
45 }

```

46 * HashMap写法

```

47 public static void main(String[] args) {
48     Map<User, Cart> carts=new HashMap<User, Cart>();
49     carts.put(new User(1001, "刘备", "123"), new Cart());
50     carts.put(new User(1002, "关羽", "123"), new Cart());
51     carts.put(new User(1003, "张飞", "123"), new Cart());
52     Set<Entry<User, Cart>> entries = carts.entrySet();

```

```

53         for(Entry<User, Cart> entry:entries) {
54             System.out.println(entry.getKey());
55         }
56     }
57     结果(无序):
58     User [id=1002, name=关羽, password=123]
59     User [id=1001, name=刘备, password=123]
60     User [id=1003, name=张飞, password=123]
61     * TreeMap: 要么User实现Comparable,要么使用Comparator构造器
62     * 假如两种都不使用会报异常: xx cannot be cast to java.lang.Comparable
63     * 第一种实现Comparable接口
64     public class User implements Comparable<User>
65     @Override
66     public int compareTo(User o) {
67         return id-o.id;
68     }
69     public static void main(String[] args) {
70         Map<User, Cart> carts=new TreeMap<User, Cart>();
71         carts.put(new User(1001, "刘备", "123"), new Cart());
72         carts.put(new User(1002, "关羽", "123"), new Cart());
73         carts.put(new User(1003, "张飞", "123"), new Cart());
74         Set<Entry<User, Cart>> entries = carts.entrySet();
75         for(Entry<User, Cart> entry:entries) {
76             System.out.println(entry.getKey());
77         }
78     }
79     结果:
80     User [id=1001, name=刘备, password=123]
81     User [id=1002, name=关羽, password=123]
82     User [id=1003, name=张飞, password=123]
83     * 第二种使用Comparator
84     public static void main(String[] args) {
85         Map<User, Cart> carts=new TreeMap<User, Cart>(new Comparator<User>() {
86
87             @Override
88             public int compare(User o1, User o2) {
89                 return o1.getId()-o2.getId();
90             }
91         });
92         carts.put(new User(1001, "刘备", "123"), new Cart());

```

```
93     carts.put(new User(1002,"关羽","123"), new Cart());
94     carts.put(new User(1003,"张飞","123"), new Cart());
95     Set<Entry<User, Cart>> entries = carts.entrySet();
96     for(Entry<User, Cart> entry:entries) {
97         System.out.println(entry.getKey());
98     }
99 }
100 * 结果:
101 User [id=1001, name=刘备, password=123]
102 User [id=1002, name=关羽, password=123]
103 User [id=1003, name=张飞, password=123]
104 * 假如既实现Comparable接口, 又使用了Comparator, Comparator的优先级比较高
```

- * 能够理解Set接口概述
- * 能够通过阅读HashSet源码理解它底层实现就是HashMap的key集合
- * 能够通过阅读LinkedHashSet源码理解它底层实现就是LinkedHashMap的key集合
- * 能够通过阅读TreeSet源码理解它底层实现就是TreeMap的key集合