# Java 核心技术

## 第十章 Java数据结构

## 第五节 映射Map

华东师范大学 陈良育

# 映射(1)

- Map映射
  - 数学定义：两个集合之间的元素对应关系。
  - 一个输入对应到一个输出
  - {1，张三}，{2，李四}，{Key，Value}，键值对，K-V对
- Java中Map
  - Hashtable（同步，慢，数据量小）
  - HashMap（不支持同步，快，数据量大）
  - Properties (同步，文件形式，数据量小)

# 映射(2)

- Hashtable
  - K-V对，K和V都不允许为null
  - 同步，多线程安全
  - 无序的
  - 适合小数据量
  - 主要方法： clear, contains/containsValue, containsKey, get, put,remove, size
  - 通过HashtableTest.java 了解其基本用法

# 映射(3)

- HashMap
  - K-V对，K和V都允许为null
  - 不同步，多线程不安全
    - Map m = Collections.synchronizedMap(new HashMap(...));
  - 无序的
  - 主要方法： clear, containsValue, containsKey, get, put,remove, size
  - 通过HashMapTest.java 了解其基本用法

# 映射(4)

- LinkedHashMap
  - 基于双向链表的维持插入顺序的HashMap
- TreeMap
  - 基于红黑树的Map，可以根据key的自然排序或者compareTo方法进行排序输出
- 查看LinkedHashMapTest.java以了解其用法
- 查看TreeMapTest.java以了解其用法

# 映射(5)

- Properties
  - 继承于Hashtable
  - 可以将K-V对保存在文件中
  - 适用于数据量少的配置文件
  - 继承自Hashtable的方法： clear, contains/containsValue, containsKey, get, put,remove, size
  - 从文件加载的load方法， 写入到文件中的store方法
  - 获取属性 getProperty ， 设置属性setProperty
  - 查看PropertiesTest了解其用法

# 映射(6)

- 总结
  - HashMap是最常用的映射结构
  - 如需要排序，考虑LinkedHashMap和TreeMap
  - 如需要将K-V存储为文件，可采用Properties类

# 代码(1) HashtableTest.java


```
public class HashtableTest {

    public static void main(String[] args) {
        Hashtable<Integer,String> ht =new  Hashtable<Integer,String>();
        //ht.put(1, null); 编译不报错 运行报错
        //ht.put(null,1);  编译报错
        ht.put(1000, "aaa");
        ht.put(2, "bbb");
        ht.put(30000, "ccc");
        System.out.println(ht.contains("aaa"));
        System.out.println(ht.containsValue("aaa"));
        System.out.println(ht.containsKey(30000));
        System.out.println(ht.get(30000));

        ht.put(30000, "ddd");  //更新覆盖ccc
        System.out.println(ht.get(30000));

        ht.remove(2);
        System.out.println("size: " + ht.size());

        ht.clear();
        System.out.println("size: " + ht.size());
```

# 代码(2) HashtableTest.java


```
        Hashtable<Integer,String> ht2 =new  Hashtable<Integer,String>();
        for(int i=0;i<100000;i++)
        {
            ht2.put(i, "aaa");
        }
        traverseByEntry(ht2);
        traverseByKeySet(ht2);
        traverseByKeyEnumeration(ht2);
    }

    public static void traverseByEntry(Hashtable<Integer,String> ht)
    {
        long startTime = System.nanoTime();
        System.out.println("============Entry迭代器遍历==============");
        Integer key;
        String value;
        Iterator<Entry<Integer, String>> iter = ht.entrySet().iterator();
        while(iter.hasNext()) {
            Map.Entry<Integer, String> entry = iter.next();
            // 获取key
            key = entry.getKey();
            // 获取value
            value = entry.getValue();
            //System.out.println("Key:" + key + ", Value:" + value);
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
```

# 代码(3) HashtableTest.java


```
public static void traverseByKeySet(Hashtable<Integer,String> ht)
{
    long startTime = System.nanoTime();
    System.out.println("=============KeySet迭代器遍历==============");
    Integer key;
    String value;
    Iterator<Integer> iter = ht.keySet().iterator();
    while(iter.hasNext()) {
        key = iter.next();
        // 获取value
        value = ht.get(key);
        //System.out.println("Key:" + key + ", Value:" + value);
    }
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    System.out.println(duration + "纳秒");
}
```

# 代码(4) HashtableTest.java


```
    public static void traverseByKeyEnumeration(Hashtable<Integer,String> ht)
    {
        long startTime = System.nanoTime();
        System.out.println("============KeyEnumeration迭代器遍历==============");
        Integer key;
        String value;
        Enumeration<Integer> keys = ht.keys();
        while(keys.hasMoreElements()) {
            key = keys.nextElement();
            // 获取value
            value = ht.get(key);
            //System.out.println("Key:" + key + ", Value:" + value);
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
}
```

# 代码(5) HashMapTest.java


```
public class HashMapTest {

    public static void main(String[] args) {
        HashMap<Integer,String> hm =new  HashMap<Integer,String>();
        hm.put(1, null);
        hm.put(null, "abc");
        hm.put(1000, "aaa");
        hm.put(2, "bbb");
        hm.put(30000, "ccc");
        System.out.println(hm.containsValue("aaa"));
        System.out.println(hm.containsKey(30000));
        System.out.println(hm.get(30000));

        hm.put(30000, "ddd");  //更新覆盖ccc
        System.out.println(hm.get(30000));

        hm.remove(2);
        System.out.println("size: " + hm.size());

        hm.clear();
        System.out.println("size: " + hm.size());
```

# 代码(6) HashMapTest.java


```
        HashMap<Integer,String> hm2 =new  HashMap<Integer,String>();
        for(int i=0;i<100000;i++)
        {
            hm2.put(i, "aaa");
        }
        traverseByEntry(hm2);
        traverseByKeySet(hm2);
    }

    public static void traverseByEntry(HashMap<Integer,String> ht)
    {
        long startTime = System.nanoTime();
        System.out.println("=============Entry迭代器遍历==============");
        Integer key;
        String value;
        Iterator<Entry<Integer, String>> iter = ht.entrySet().iterator();
        while(iter.hasNext()) {
            Map.Entry<Integer, String> entry = iter.next();
            // 获取key
            key = entry.getKey();
            // 获取value
            value = entry.getValue();
            //System.out.println("Key:" + key + ", Value:" + value);
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
```

# 代码(7) HashMapTest.java


```
    public static void traverseByKeySet(HashMap<Integer,String> ht)
    {
        long startTime = System.nanoTime();
        System.out.println("=============KeySet迭代器遍历==============");
        Integer key;
        String value;
        Iterator<Integer> iter = ht.keySet().iterator();
        while(iter.hasNext()) {
            key = iter.next();
            // 获取value
            value = ht.get(key);
            //System.out.println("Key:" + key + ", Value:" + value);
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
}
```

# 代码(8) LinkedHashMapTest.java


```
public class LinkedHashMapTest {

    public static void main(String[] args) {
        LinkedHashMap<Integer,String> hm =new  LinkedHashMap<Integer,String>();
        hm.put(1, null);
        hm.put(null, "abc");
        hm.put(1000, "aaa");
        hm.put(2, "bbb");
        hm.put(30000, "ccc");
        System.out.println(hm.containsValue("aaa"));
        System.out.println(hm.containsKey(30000));
        System.out.println(hm.get(30000));

        hm.put(30000, "ddd");  //更新覆盖ccc
        System.out.println(hm.get(30000));

        hm.remove(2);
        System.out.println("size: " + hm.size());

        //hm.clear();
        //System.out.println("size: " + hm.size());
```

# 代码(9) LinkedHashMapTest.java


```
        System.out.println("遍历开始==================");

        Integer key;
        String value;
        Iterator<Entry<Integer, String>> iter = hm.entrySet().iterator();
        while(iter.hasNext()) {
            Map.Entry<Integer, String> entry = iter.next();
            // 获取key
            key = entry.getKey();
            // 获取value
            value = entry.getValue();
            System.out.println("Key:" + key + ", Value:" + value);
        }
        System.out.println("遍历结束==================");

        LinkedHashMap<Integer,String> hm2 =new  LinkedHashMap<Integer,String>();
        for(int i=0;i<100000;i++)
        {
            hm2.put(i, "aaa");
        }
        traverseByEntry(hm2);
        traverseByKeySet(hm2);
    }
```

# 代码(10) LinkedHashMapTest.java


```
public static void traverseByEntry(LinkedHashMap<Integer,String> ht)
{
    long startTime = System.nanoTime();
    System.out.println("============Entry迭代器遍历==============");
    Integer key;
    String value;
    Iterator<Entry<Integer, String>> iter = ht.entrySet().iterator();
    while(iter.hasNext()) {
        Map.Entry<Integer, String> entry = iter.next();
        // 获取key
        key = entry.getKey();
        // 获取value
        value = entry.getValue();
        //System.out.println("Key:" + key + ", Value:" + value);
    }
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    System.out.println(duration + "纳秒");
}
```

# 代码(11) LinkedHashMapTest.java


```
    public static void traverseByKeySet(LinkedHashMap<Integer,String> ht)
    {
        long startTime = System.nanoTime();
        System.out.println("============KeySet迭代器遍历==============");
        Integer key;
        String value;
        Iterator<Integer> iter = ht.keySet().iterator();
        while(iter.hasNext()) {
            key = iter.next();
            // 获取value
            value = ht.get(key);
            //System.out.println("Key:" + key + ", Value:" + value);
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
}
```

# 代码(12) TreeMapTest.java


```
public class TreeMapTest {

    public static void main(String[] args) {
        TreeMap<Integer,String> hm =new  TreeMap<Integer,String>();
        hm.put(1, null);
        //hm.put(null, "abc");   编译没错，运行报空指针异常
        hm.put(1000, "aaa");
        hm.put(2, "bbb");
        hm.put(30000, "ccc");
        System.out.println(hm.containsValue("aaa"));
        System.out.println(hm.containsKey(30000));
        System.out.println(hm.get(30000));

        hm.put(30000, "ddd");  //更新覆盖ccc
        System.out.println(hm.get(30000));

        //hm.remove(2);
        System.out.println("size: " + hm.size());

        //hm.clear();
        //System.out.println("size: " + hm.size());
```

# 代码(13) TreeMapTest.java


```
        System.out.println("遍历开始==================");

        Integer key;
        String value;
        Iterator<Entry<Integer, String>> iter = hm.entrySet().iterator();
        while(iter.hasNext()) {
            Map.Entry<Integer, String> entry = iter.next();
            // 获取key
            key = entry.getKey();
            // 获取value
            value = entry.getValue();
            System.out.println("Key:" + key + ", Value:" + value);
        }
        System.out.println("遍历结束==================");

        TreeMap<Integer,String> hm2 =new  TreeMap<Integer,String>();
        for(int i=0;i<100000;i++)
        {
            hm2.put(i, "aaa");
        }
        traverseByEntry(hm2);
        traverseByKeySet(hm2);
    }
```

# 代码(14) TreeMapTest.java


```java
public static void traverseByEntry(TreeMap<Integer,String> ht)
{
    long startTime = System.nanoTime();
    System.out.println("============Entry迭代器遍历==============");
    Integer key;
    String value;
    Iterator<Entry<Integer, String>> iter = ht.entrySet().iterator();
    while(iter.hasNext()) {
        Map.Entry<Integer, String> entry = iter.next();
        // 获取key
        key = entry.getKey();
        // 获取value
        value = entry.getValue();
        //System.out.println("Key:" + key + ", Value:" + value);
    }
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    System.out.println(duration + "纳秒");
}
```

# 代码(15) TreeMapTest.java


```
    public static void traverseByKeySet(TreeMap<Integer,String> ht)
    {
        long startTime = System.nanoTime();
        System.out.println("============KeySet迭代器遍历==============");
        Integer key;
        String value;
        Iterator<Integer> iter = ht.keySet().iterator();
        while(iter.hasNext()) {
            key = iter.next();
            // 获取value
            value = ht.get(key);
            //System.out.println("Key:" + key + ", Value:" + value);
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
}
```

# 代码(16) MapCompareTest.java


```java
public class MapCompareTest {

    public static void main(String[] args) {
        int n = 1000000;
        System.out.println("============HashMap=============");
        HashMap<Integer,String> hm =new  HashMap<Integer,String>();
        for(int i=0;i<n;i++) {
            hm.put(i, "aaa");
        }
        traverseByEntry(hm);
        traverseByKeySet(hm);

        System.out.println("============Hashtable=============");
        Hashtable<Integer,String> ht2 =new  Hashtable<Integer,String>();
        for(int i=0;i<n;i++) {
            ht2.put(i, "aaa");
        }
        traverseByEntry(ht2);
        traverseByKeySet(ht2);
    }
```

# 代码(17) MapCompareTest.java


```
public static void traverseByEntry(Map<Integer,String> ht)
{
    long startTime = System.nanoTime();
    System.out.println("=============Entry迭代器遍历==============");
    Integer key;
    String value;
    Iterator<Entry<Integer, String>> iter = ht.entrySet().iterator();
    while(iter.hasNext()) {
        Map.Entry<Integer, String> entry = iter.next();
        // 获取key
        key = entry.getKey();
        // 获取value
        value = entry.getValue();
        //System.out.println("Key:" + key + ", Value:" + value);
    }
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    System.out.println(duration + "纳秒");
}
```

# 代码(18) MapCompareTest.java


```
    public static void traverseByKeySet(Map<Integer,String> ht)
    {
        long startTime = System.nanoTime();
        System.out.println("============KeySet迭代器遍历==============");
        Integer key;
        String value;
        Iterator<Integer> iter = ht.keySet().iterator();
        while(iter.hasNext()) {
            key = iter.next();
            // 获取value
            value = ht.get(key);
            //System.out.println("Key:" + key + ", Value:" + value);
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
}
```

# 代码(19) PropertiesTest


```
//关于Properties类常用的操作
public class PropertiesTest {
    //根据Key读取Value
    public static String GetValueByKey(String filePath, String key) {
        Properties pps = new Properties();
        try {
            InputStream in = new BufferedInputStream (new FileInputStream(filePath));
            pps.load(in); //所有的K-V对都加载了
            String value = pps.getProperty(key);
            //System.out.println(key + " = " + value);
            return value;

        }catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }
```

# 代码(20) PropertiesTest


```
//读取Properties的全部信息
public static void GetAllProperties(String filePath) throws IOException {
    Properties pps = new Properties();
    InputStream in = new BufferedInputStream(new FileInputStream(filePath));
    pps.load(in); //所有的K-V对都加载了
    Enumeration en = pps.propertyNames(); //得到配置文件的名字

    while(en.hasMoreElements()) {
        String strKey = (String) en.nextElement();
        String strValue = pps.getProperty(strKey);
        //System.out.println(strKey + "=" + strValue);
    }

}
```

# 代码(21) PropertiesTest


```
//写入Properties信息
public static void WriteProperties (String filePath, String pKey, String pValue) throws IOException {
    File file = new File(filePath);
    if(!file.exists())
    {
        file.createNewFile();
    }
    Properties pps = new Properties();

    InputStream in = new FileInputStream(filePath);
    //从输入流中读取属性列表（键和元素对）
    pps.load(in);
    //调用 Hashtable 的方法 put。使用 getProperty 方法提供并行性。
    //强制要求为属性的键和值使用字符串。返回值是 Hashtable 调用 put 的结果。
    OutputStream out = new FileOutputStream(filePath);
    pps.setProperty(pKey, pValue);
    //以适合使用 load 方法加载到 Properties 表中的格式，
    //将此 Properties 表中的属性列表（键和元素对）写入输出流
    pps.store(out, "Update " + pKey + " name");
    out.close();
}
```

# 代码(22) PropertiesTest


```
    public static void main(String [] args) throws IOException{
        System.out.println("写入Test.properties================");
        WriteProperties("Test.properties","name", "12345");

        System.out.println("加载Test.properties================");
        GetAllProperties("Test.properties");

        System.out.println("从Test.properties加载================");
        String value = GetValueByKey("Test.properties", "name");
        System.out.println("name is " + value);
    }
}
```

# 谢 谢！