* **Map实现类之一：HashMap**
* HashMap是Map接口使用频率最高的实现类。
* **key是用Set来存放**的，不可重复。**value是用collection来存放**的，可重复。
* 一个key-value对，是一个Entry。所有的**Entry是用Set存放**的，也是不可重复的。
* **允许使用null键和null值**，与HashSet一样，不保证映射的顺序。
* HashMap判断两个key相等的标准是：**两个key通过equals()方法返回true，hashCode值也相等**。
* HashMap判断两个value相等的标准是：**两个value通过equals()方法返回true**。
* **向HashMap中添加元素时，会调用key所在类的equals()方法，判断两个key是否相同。若相同，则只能添加进后添加的哪个元素。**
* 遍历

@Test

public void test2() {  
 Map map = new HashMap(16);  
 map.put("AA", 213);  
 map.put("BB", 45);  
 map.put(123, "CC");  
 map.put(null, null);  
 map.put(new Person("DD", 23), 89);  
  
 //1）遍历key集  
 Set set = map.keySet();  
 for (Object obj : set) {  
 System.***out***.println(obj);  
 }  
 //2）遍历value集  
 Collection values = map.values();  
 Iterator iterator = values.iterator();  
 while (iterator.hasNext()) {  
 System.***out***.println(iterator.next());  
 }  
 //3）如何遍历key-value对  
 // 方法一：  
 Set set1 = map.keySet();  
 for (Object obj : set1) {  
 System.***out***.println(obj + "-------->" + map.get(obj));  
 }  
 // 方法二：  
 Set set2 = map.entrySet();  
 for (Object obj : set2) {  
 Map.Entry entry = (Map.Entry) obj;  
 //System.out.println(entry.getKey() + "----->" + entry.getValue());  
 System.***out***.println(entry);  
 }  
}

* **Map实现类之二：LinkedHashMap**
* LinkedHashMap是HashMap的子类。
* **使用链表维护添加进Map中的顺序。故遍历Map时，是按添加的顺序遍历的**。
* 与LinkedHashSet类似，LinkedHashMap可以维护Map的迭代顺序：迭代顺序与Key-Value对的插入顺序一致。
* **Map实现类之三：TreeMap**
* TreeMap存储Key-Value对时，需要根据key-value对进行排序。**TreeMap可以保证所有的Key-Value对处于有序状态**。
* TreeMap的Key的排序：
  + **自然排序**：TreeMap的**所有的Key必须实现Comparable接口**，而且所有的Key应该是**同一个类的对象**，否则将会抛出ClasssCastException。
  + **定制排序**：创建TreeMap时，传入一个Comparator对象，该对象负责对TreeMap中的所有key进行排序。此时不需要Map的Key实现Comparable接口。
* **TreeMap判断两个key相等的标准：两个key通过compareTo()方法或者compare()方法返回0**。
* 若使用自定义类作为TreeMap的key，所属类需要重写equals()和hashCode()方法，且equals()方法返回true时，compareTo()方法应返回0。
* **Map实现类之四：Hashtable**
* Hashtable是个古老的Map实现类，线程安全。
* 与HashMap不同，**Hashtable不允许使用null作为key和value**。
* 与HashMap一样，Hashtable也不能保证其中Key-Value对的顺序。
* Hashtable判断两个key相等、两个value相等的标准，与HashMap一致。
* **Map实现类之五：Properties**
* Properties类是Hashtable的子类，该对象**用于处理属性文件**。
* 由于属性文件里的key、value都是字符串类型，所以**Properties里的key和value都是字符串类型**。
* 存取数据时，建议使用**setProperty(String key,String value)方法和getProperty(String key)方法**。

Properties pros=new Properties();

pros.load(new FileInputStream("jdbc.properties"));

String user=pros.getProperty("user");

System.out.println(user);

@Test  
public void test6() throws IOException {  
 Properties pros = new Properties();  
 pros.load(new FileInputStream(new File("jdbc.properties")));  
 String user = pros.getProperty("user");  
 System.***out***.println(user);  
 String password = pros.getProperty("password");  
 System.***out***.println(password);  
}