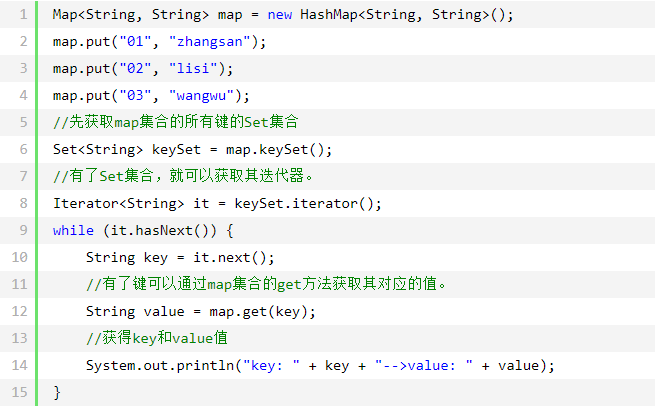
1、Map对象中keyset()和entryset()区别

Set<K>keyset() //返回值是个只存放key值的Set集合（集合中无序存放的）

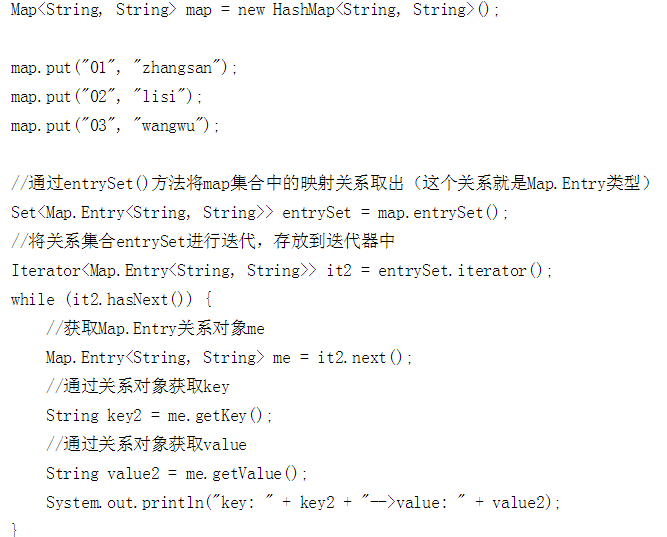
Set<Map.Entry<K.V>>entrySet() //返回映射所包含的映射关系的Set集合（一个关系就是一个键-值对），就是把（key-value）作为一个整体一对一对的存放到Set集合当中。

例子：

1. Keyset()方式



1. entrySet()方式：



虽然使用keyset及entryset来进行遍历能取得相同的结果，但两者的遍历速度是有差别的。

keySet()：迭代后只能通过get(key)取值

entrySet()：迭代后可以e.getKey(),e.getValue取key和value。返回的是Entry接口。

说明：keySet()的速度比entrySet()慢了很多，也就是keySet方式遍历的性能不如entrySet性能好。

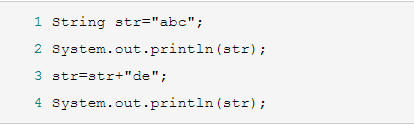
2、Java中的String,StringBuilder,StringBuffer

这三个类之间的区别主要是在两个方面，即运行速度和线程安全这两方面。

1. 首先说运行速度，或者说是执行速度，在这方面运行速度快慢为：StringBuilder > StringBuffer > String

String最慢的原因：

String为字符串常量，而StringBuilder和StringBuffer均为字符串变量，即String对象一旦创建之后该对象是不可更改的，但后两者的对象是变量，是可以更改的。以下面一段代码为例：



如果运行这段代码会发现先输出”abc”，然后又输出”abcde”,好像是str这个对象被更改了，其实，这只是一种假象罢了。JVM对于这几行代码是这样处理的，首先创建一个String对象str，并把”abc”赋值给str，然后在第三行中，其实JVM又创建了一个新的对象也名为str，然后再把原来的str的值和”de”加起来再赋值给新的str，而原来的str就会被JVM的垃圾回收机制(GC)给回收掉了，所以,str实际上并没有被更改，也就是前面说的String对象一旦创建之后就不可更改了。所以，Java中对String对象进行的操作实际上是一个不断创建新的对象并且将旧的对象回收的一个过程，所以执行速度很慢。

而StringBuilder和StringBuffer的对象是变量，对变量进行的操作就是直接对该对象进行更改，而不进行创建和回收的操作，所以速度要比String快很多。

1. 再来说线程安全

在线程安全上，StringBuilder是线程不安全的，而StringBuffer是线程安全的

如果一个StringBuffer对象在字符串缓冲区被多个线程使用时，StringBuffer中很多方法可以带有synchronized关键字，所以可以保证线程安全的，但StringBuilder的方法则没有该关键字，所以不能保证线程安全，有可能会出现一些错误的操作。所以如果要进行的操作是多线程的，那么就要使用StringBuffer,但是单线程的情况下，还是建议使用速度比较快的StringBuilder.

③ 总结一下

String : 适用于少量的字符串操作的情况

StringBuilder : 适用于单线程下在字符缓冲区进行大量操作的情况

StringBuffer : 适用于多线程下在字符缓冲区进行大量操作的情况

3、hashcode()和equals()的作用、区别、联系

在java中任何一个对象都具备equals(Object obj)和hashCode()这两个方法,因为他们是在Obejct类中定义的.

在java应用程序执行期间,在同一对象上多次调用hashcode方法时,必须一致地返回相同的整数,前提是对象上equals比较中所用的信息没有被修改.从某一应用程序的一次执行到同一应用程序的另一次执行,该整数无需保持一致.

小的归纳：

若重写了equals(Object obj)方法,则有必要重写hashCode()方法.

若两个对象equals(Object obj)返回true,则hashCode()也返回相同的int数.

若两个对象equals(Object obj)返回false,则hashCode()不一定返回不同的int数.

若两个对象hashCode()返回相同的int数,则equals(Object obj)不一定返回true.

若两个对象hashCode()返回不同的int数,则equals(Object obj)一定返回false.

同一对象在执行期间若已经存储在集合中,则不能修改影响hahCode值的相关信息,否则会导致内存泄漏.

<https://blog.csdn.net/qq_21688757/article/details/53067814>

<https://www.cnblogs.com/keyi/p/7119825.html>

<https://blog.csdn.net/haobaworenle/article/details/53819838>