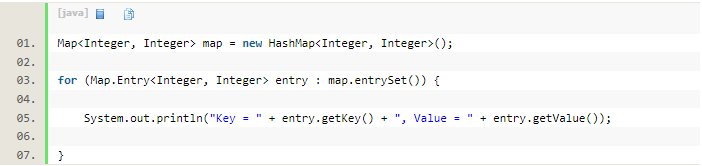
既然java中的所有map都实现了Map接口，以下方法适用于任何map实现（HashMap, TreeMap, LinkedHashMap, Hashtable, 等等）

**方法一 在for-each循环中使用entries来遍历**

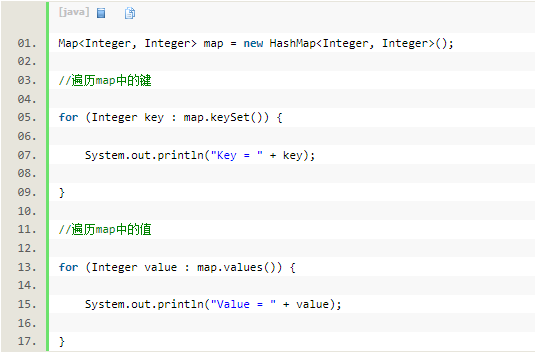
这是最常见的并且在大多数情况下也是最可取的遍历方式。在键值都需要时使用。



注意：for-each循环在java 5中被引入所以该方法只能应用于java 5或更高的版本中。如果你遍历的是一个空的map对象，for-each循环将抛出NullPointerException，因此在遍历前你总是应该检查空引用。

**方法二 在for-each循环中遍历keys或values。**

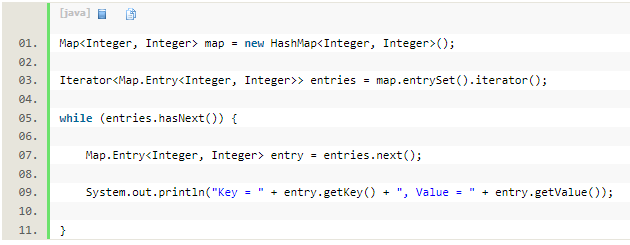
如果只需要map中的键或者值，你可以通过keySet或values来实现遍历，而不是用entrySet。



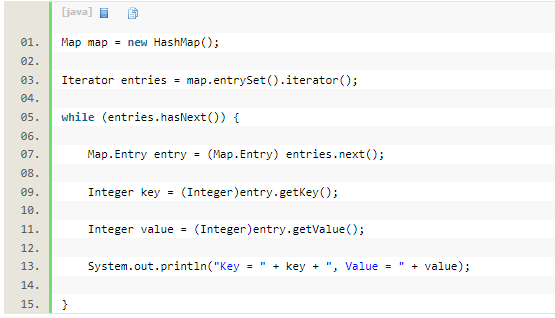
该方法比entrySet遍历在性能上稍好（快了10%），而且代码更加干净。

**方法三使用Iterator遍历**

使用泛型：



不使用泛型：

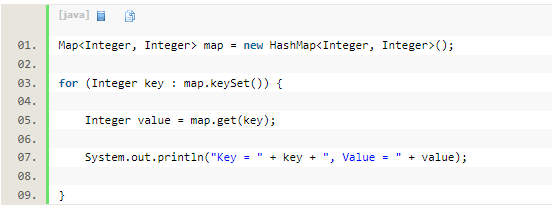


你也可以在keySet和values上应用同样的方法。

该种方式看起来冗余却有其优点所在。首先，在老版本java中这是惟一遍历map的方式。另一个好处是，你可以在遍历时调用iterator.remove()来删除entries，另两个方法则不能。根据javadoc的说明，如果在for-each遍历中尝试使用此方法，结果是不可预测的。

从性能方面看，该方法类同于for-each遍历（即方法二）的性能。

**方法四、通过键找值遍历（效率低）**



作为方法一的替代，这个代码看上去更加干净；但实际上它相当慢且无效率。因为从键取值是耗时的操作（与方法一相比，在不同的Map实现中该方法慢了20%~200%）。如果你安装了FindBugs，它会做出检查并警告你关于哪些是低效率的遍历。所以尽量避免使用。

**总结**

如果仅需要键(keys)或值(values)使用方法二。如果你使用的语言版本低于java 5，或是打算在遍历时删除entries，必须使用方法三。否则使用方法一(键值都要)。