**There are 4 way to traverse or loop Map, HashMap or TreeMap in Java**

1. Using Java5 foreach loop and using KeySet of map for getting keys. this will iterate through all values of Map and display key and value together.

HashMap<String, String> loans = new HashMap<String, String>();

loans.put("home loan", "citibank");loans.put("personal loan", "Wells Fargo");

for (String key : loans.keySet()) {

   System.out.println("key: " + key + " value: " + loans.get(key));}

1. Using KeySet Iterator we have used [Java Iterator](http://javarevisited.blogspot.com/2011/10/java-iterator-tutorial-example-list.html) instead of for loop

Set<String> keySet = loans.keySet();

Iterator<String> keySetIterator = keySet.iterator();

while (keySetIterator.hasNext()) {

String key = keySetIterator.next();

   System.out.println("key: " + key + " value: " + loans.get(key));}

1. **Looping HashMap in Java using EntrySet and Java 5 for loop**

Set<Map.Entry<String, String>> entrySet = loans.entrySet();

for (Entry entry : entrySet) {   System.out.println("---------------------");

   System.out.println("key: " + entry.getKey() + " value: " + entry.getValue());

}

1. **Using EntrySet and Java iterator** Combination of Iterator and EntrySet.

Set<Map.Entry<String, String>> entrySet1 = loans.entrySet();

Iterator<Entry<String, String>> entrySetIterator = entrySet1.iterator();

while (entrySetIterator.hasNext()) {

  Entry entry = entrySetIterator.next();

   System.out.println("key: " + entry.getKey()+" value:"+ entry.getValue());}

**Label :** In Java, any [statement](https://www.cis.upenn.edu/~matuszek/General/JavaSyntax/statements.html) can be given a label (a name followed by a colon). However, it is only useful to label [loops](https://www.cis.upenn.edu/~matuszek/General/JavaSyntax/loops.html) and [switch statements](https://www.cis.upenn.edu/~matuszek/General/JavaSyntax/statements.html).

A [**break**](https://www.cis.upenn.edu/~matuszek/General/JavaSyntax/break.html) statement typically takes control out of the innermost loop or **switch** statement. However, it can also specify the label of *any* enclosing labeled loop or **switch** statement, to cause that statement to exit. For example:

boolean containsZero = false;

zeroCheck:

for (int i = 0; i < myArray.length; i++) {

for (int j = 0; j < myArray[i].length; j++) {

if (myArray[i][j] == 0) {

containsZero = true;

break zeroCheck;

}

} }

**STYLE** Use labels only as a last resort, after exploring other possible means of writing the code. For example, the above code could be better written as a call to the following method:

boolean containsZero(int[][] myArray) {

for (int i = 0; i < myArray.length; i++) {

for (int j = 0; j < myArray[i].length; j++) {

if (myArray[i][j] == 0) return true;

}

}

return false;

}