**Maps**

Integer 🡪 String

Integer 🡪 String

Integer 🡪 String

Integer 🡪 String

The *map* data structure stores many pairs of objects, called "keys" and "values." If you know the key, you can get to the value. (In arrays, the "key" is the index of the value.). Keys must be unique. You should be able to add pairs, remove keys, and test for keys in any map. Look at the API cheat sheet. How do you iterate over a map? (There are 2 ways.) Write code that prints the values in myMap, which maps Integers to Strings.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

HashMap<K,V> and TreeMap<K,V> are two concrete classes that implement the Map<K,V> interface.

A HashMap stores objects according to its hash function, which means the actual storage location is impossible to predict in advance. A TreeMap stores objects in BST order. Therefore, if the order of the objects is not important, use HashMaps. If the order of the objects is important, use TreeMaps.

A HashMap provides O(1) run times for the get and put operations. A TreeMap provides O(log n) run times for the get and put operations. That's because Java implements TreeMaps as self-balancing binary search trees. Nice.

**Example code for Maps**

6 Map<String, String> h = new HashMap<String, String>();  
 7 h.put("Othello", "green");  
 8 h.put("MacBeth", "XXX");  
 9 h.put("MacBeth", "red"); //what happens if the key is the same?  
10 h.put("Hamlet", "blue");   
11 if(!h.containsKey("Lear"))  
12 h.put("Lear", "black");  
13 System.out.println( h.containsKey("Othello") );   
14 System.out.println( h.keySet() ); //print the \_\_\_\_\_\_\_\_\_\_  
15   
16 Iterator<String> it = h.keySet().iterator(); //use an iterator   
17 while(it.hasNext())  
18 System.out.print( h.get(it.next()) ); //print the \_\_\_\_\_\_\_\_   
19 System.out.println();  
20   
21 Map<String, String> t = new TreeMap<String, String>(h); //from

//HashMap to TreeMap  
22 for( String str : t.keySet() ) //use a for-each  
23 System.out.print( t.get( str ) ); //print the \_\_\_\_\_\_\_\_\_   
24 System.out.println();  
25 System.out.println(t); //print any Collection—wow!

Output: line 13: \_\_\_\_\_\_\_\_\_

line 14: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

lines 16-18:  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

lines 22-23:  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

line 25:  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**