**Initializing a Java HashMap with Reasonable Values:**

**How a Java HashMap is implemented?**

**Example:** HashMap<String, String > map = **new** HashMap<String, String > ();

Above example is in principal valid Java code, but the hash map is not initialized with a reasonable values. In this case the Java implementation has to resize the number of buckets with increasing number of data in the hash map. In Java the default load factor is 0.75 and the default initial capacity of a hash map is 16.

This means that the threshold for the next resize equates like:

**DEFAULT\_INITIAL\_CAPACITY = 16;**

**DEFAULT\_LOAD\_FACTOR = 0.75;**

**THRESHOLD = DEFAULT\_INITIAL\_CAPACITY \* DEFAULT\_LOAD\_FACTOR;**

The hash map will automatically rehash the contents of the hash map into a new array with a larger capacity. This happens when the number of data in the hash map exceeds the THRESHOLD. Each resize leads to a doubled capacity 16\*0.75, 32\*0.75, 64\*0.75, etc. The following diagram depicts this:

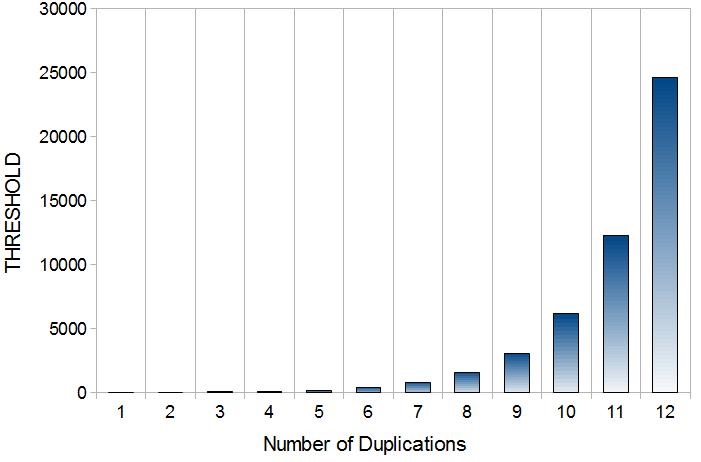
[[](http://www.sw-engineering-candies.com/blog-1/howtoinitializeajavahashmapwithreasonablevalues/Threshold-HashMap.jpg?attredirects=0)](http://www.sw-engineering-candies.com/blog-1/howtoinitializeajavahashmapwithreasonablevalues/Threshold-HashMap.jpg?attredirects=0)

Figure 1: Growth of HashMap

Setting the initial capacity to just a high value is no solution, because an iteration over collection requires time proportional to the capacity of the hash map. It is important not to set the initial capacity too high if you iterate over the values.

**How to initialize a Java HashMap with reasonable values?**

The minimal initial capacity would be **(number of data)/0.75+1**.

***int initialCapacity = (int) ((expected\_maximal\_number\_of\_data)/0.75+1);***

***HashMap<String, String> map = new HashMap<String, String > (initialCapacity);***

For small hash maps there is no big difference, but for situations with really large hash maps (e.g. 100 MB size) it can seriously increase the needed memory and the rehashing causes a slowdown of the application.