**ANSWER THESE QUESTIONS**

**1. Why to use java.util.WeakHashMap map which is so inconsistent and unpredictable in behaviour?**

**Answer:**

Let's say we have huge application which consists of lots n lots of object and may run short of memory at any time, we will like [garbage collector](http://www.javamadesoeasy.com/2015/09/how-garbage-collection-works-internally.html) to quickly discard less used key value pair to free up some memory. As, behavior of the WeakHashMap class depends upon garbage collector.

I believe discarding less used key-value is always going to a better option than running out of memory.

**2. Let’s say you have to build dictionary and multiple users can add data in that dictionary? And you can use 2 Collection classes? Which Collection classes you will prefer and WHY?**

**Answer:**

**3. How ConcurrentHashMap works? Can 2 threads on same ConcurrentHashMap object access it concurrently?**

**Answer:**

**4. How can you sort given HashMap on basis of keys?**

**Answer:**

**5. What are different ways of iterating over keys, values and entry in Map?**

**Answer:**

**6.What are differences between HashMap vs IdentityHashMap in java?**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Property | ***java.util.HashMap*** | ***java.util.IdentityHashMap*** |
| 1 | **Keys comparison***object-equality  vs reference-equality* | HashMap uses object equality to compare the key and values. (equals() method) | IdentityHashMap uses reference equality to compare the key and values.  Operator “==” |
| 2 | Initial size | 16 | 21 |
| 3 | Introduced in which java version | Jdk 1.2 | Jdk 1.4 |
| 4 | *Program* |  |  |
| 5 | overridden equals() and hashCode() method call? | overridden equals()  uses hashCode() | does not use hashCode() method instead it uses System.identityHashCode() |
| 6 | Application - can maintain *proxy object* |  |  |

**7. What are differences between HashMap and ConcurrentHashMap in java?**

|  |  |  |
| --- | --- | --- |
| Property | *java.util.****HashMap*** | *java.util.concurrent.* ***ConcurrentHashMap*** |
| synchronization | non-synchronized | synchronized |
| 2 threads on same Map object can access it at concurrently? |  |  |
| Performance | HashMap performance is relatively high because it is non-synchronized in nature and any number of threads can perform simultaneously | ConcurrentHashMap performance is low sometimes because sometimes Threads are required to wait on ConcurrentHashMap. |
| Null keys and values | null values are allowed for key and values | null value is not allowed for key and value, otherwise we will get Run-time exception saying**NullPointerException.** |
| Iterators | HashMap iterator is fail-fast and ArrayList throws ConcurrentModificationException if concurrent modification happens during iteration. | ConcurrentHashMap is fail-safe and it will never throw ConcurrentModificationException during iteration. |
| putIfAbsent |  |  |
| Introduced  in which java version | Java 2 (JDK 1.2) | Java 5 (JDK 1.5). |
| Implements which interface |  |  |
| Package |  |  |

**8. When to use HashMap vs Hashtable vs LinkedHashMap ?**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Property** | ***HashMap*** | ***Hashtable*** | ***LinkedHashMap*** |
| 1 | Insertion order |  |  |  |
| 2 | Performance |  |  |  |
| 3 | Null keys and values |  |  |  |
| 4 | Implements which interface |  |  |  |
| 5 | Implementation uses? |  |  |  |
| 6 | Complexity of put, get and remove methods |  |  |  |
| 7 | Extends java.util.**Dictionary** (Abstract class, which is obsolete) |  |  |  |
| 8 | Introduced in which java version? |  |  |  |