Design a caching API in Java

Category: System Design

I usually use this on candidates with Java development experience to assess knowledge of the platform and OO design. It's quite open-ended so I'll note some common responses and things to look for. I find it a great measure of Java knowledge from newbie to total guru. Those who barely know Java can do very little but not be completely stumped; there's plenty of material to cover here though that'll let the gurus shine.

I ask the candidate to describe a simple, in-memory caching API for Java, of the sort that might, say, be used to cache relatively static data (e.g. user profile) by key (e.g. user ID). I ask for an interface and implementation.

1. Can they **write an interface declaration correctly**? Hopefully we get something like:

```
public interface Cache {
  Object get(Object key);
  void put(Object key, Object value);
  void clear();
  ...
}
```

```
public interface Cache<K, V> {
    V get(K key);
    void put(K key, V value);
    void clear();
}
```

Sometimes candidates suggest odd operations here. Follow up if you like. Sometimes they want the cache to return the key on put. OK.

2. Can they suggest a simple implementation?
Hopefully sometihng very simple based on "**HashMap**" comes up or something quite comparable. Ask them to code it if you like. These first two reveal candidates that can't really program in Java.

3. **Generics（泛型）**?
If not already used, prompt for use of generics. Should suggest a key and value generic type, a la Map.

4. **Synchronization**?
Ask about **thread-safety**. Looking for basic discussion of "**synchronized**", or maybe newer, smarter data structures like **ConcurrentHashMap**.

5. Memory usage?
Ask about **how to avoid triggering an OutOfMemoryException if cached data gets very large**. Really, I'm probing for advanced knowledge of the platform, and **WeakReference/SoftReference** discussion. Not a big deal if they can't come up with it.

6. **Limit cache size by memory usage**?
Ask how to limit the cache size by memory usage**. It can't really be done in Java since object size is not known for sure. Great candidates might talk about ways to estimate size.**

7. **Expire cache entries by age?**
Will probably suggest a **timestamp** somewhere. Where, and, when is it checked, passively on get()? **How to actively find and expire entrie**s**? do they talk about Threads or even a Timer**? locking parts of the data structure and not the whole thing?

8. **Limit cache size by number of entries**
Great point to hit -- see how well the candidate can come up with basic strategies like least-recently-used, least-frequently-used, random, etc. Probe for detail on how to implement these efficiently.

9. Cache synchronization
Ask for discussion of how to make sure outdated data isn't cached. No real right answer here.

As an aside, my personal favorite API would be something that does not have a set() operation. I personally think the cache should encapsulate knowledge about how to build an entry on demand and thus should be constructed with some kind of "Retriever" implementation that gets values from backing store. A few candidates have gone this route. Something to chew on if you find yourself looking for more to discuss with a great candidate -- push them down this path.