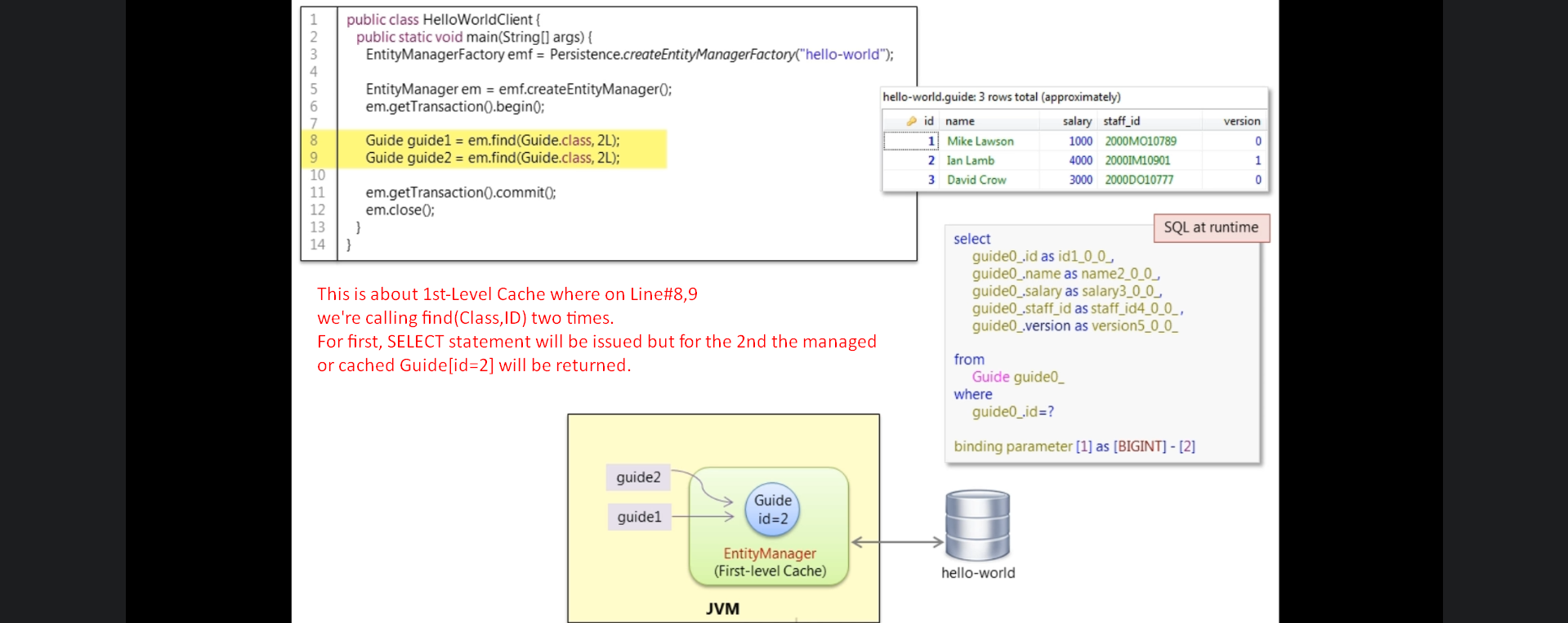
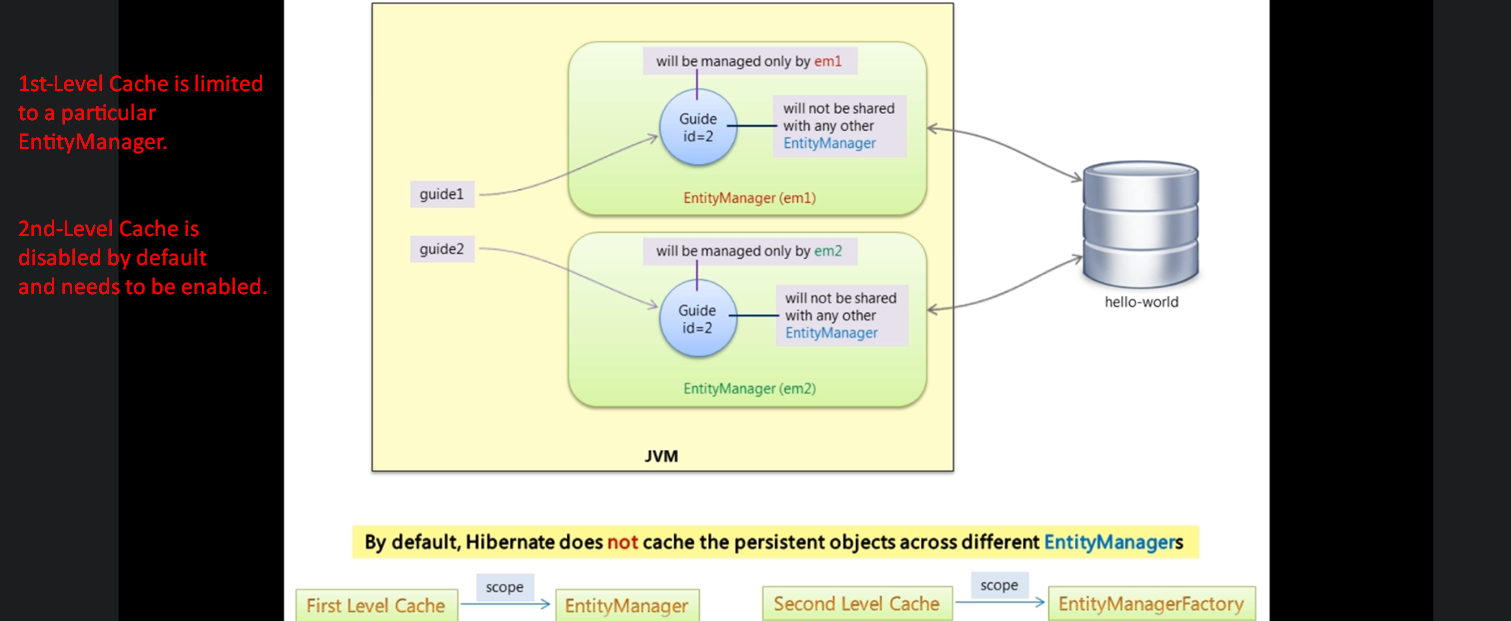
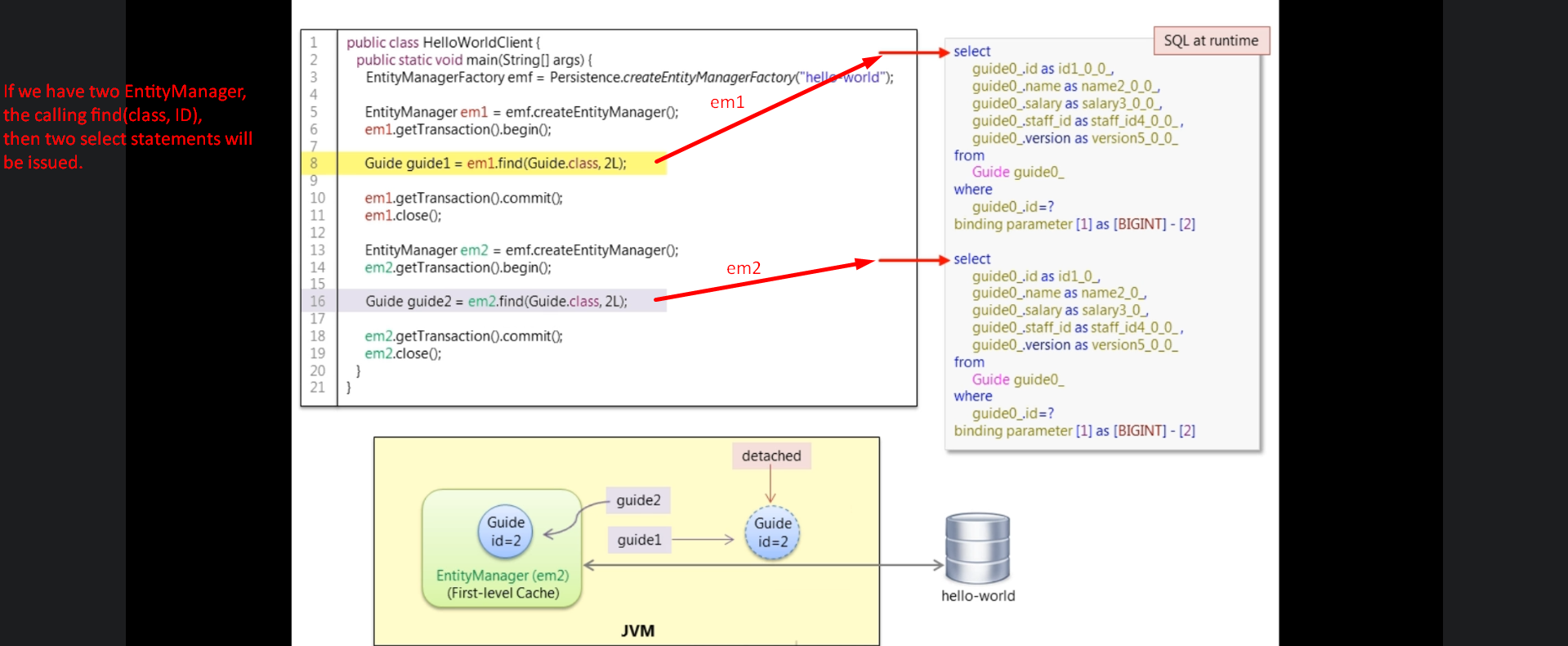
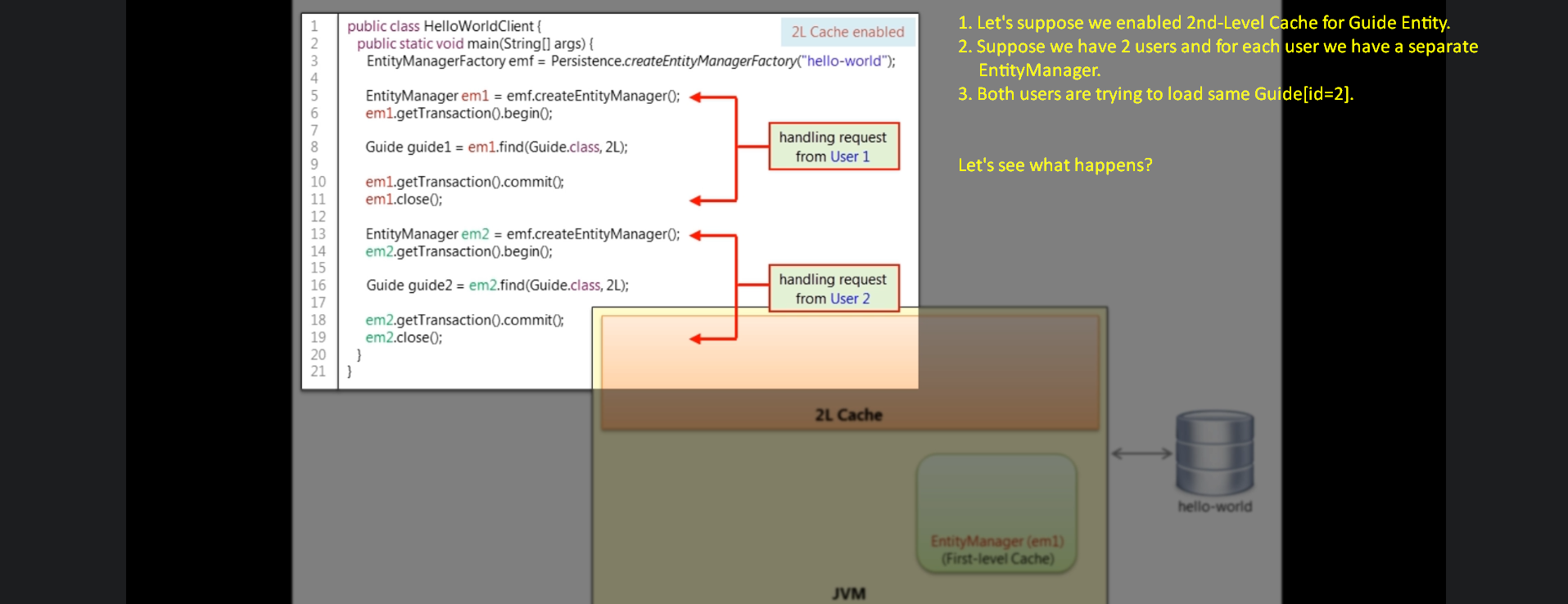
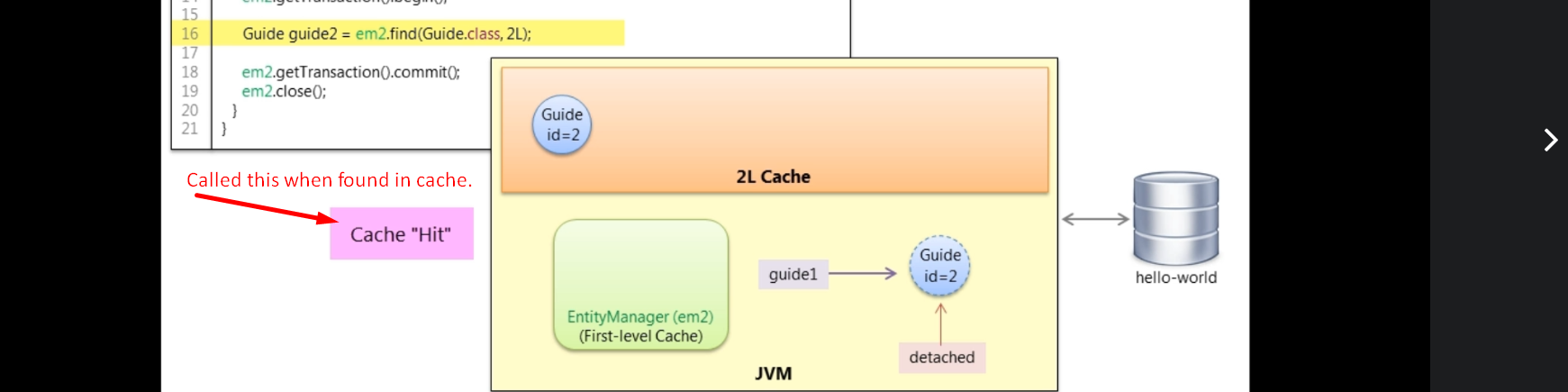
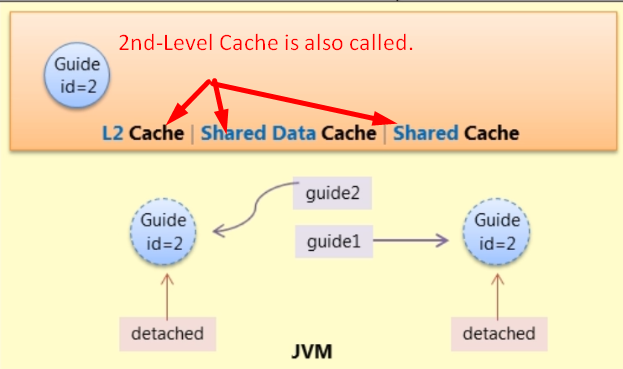
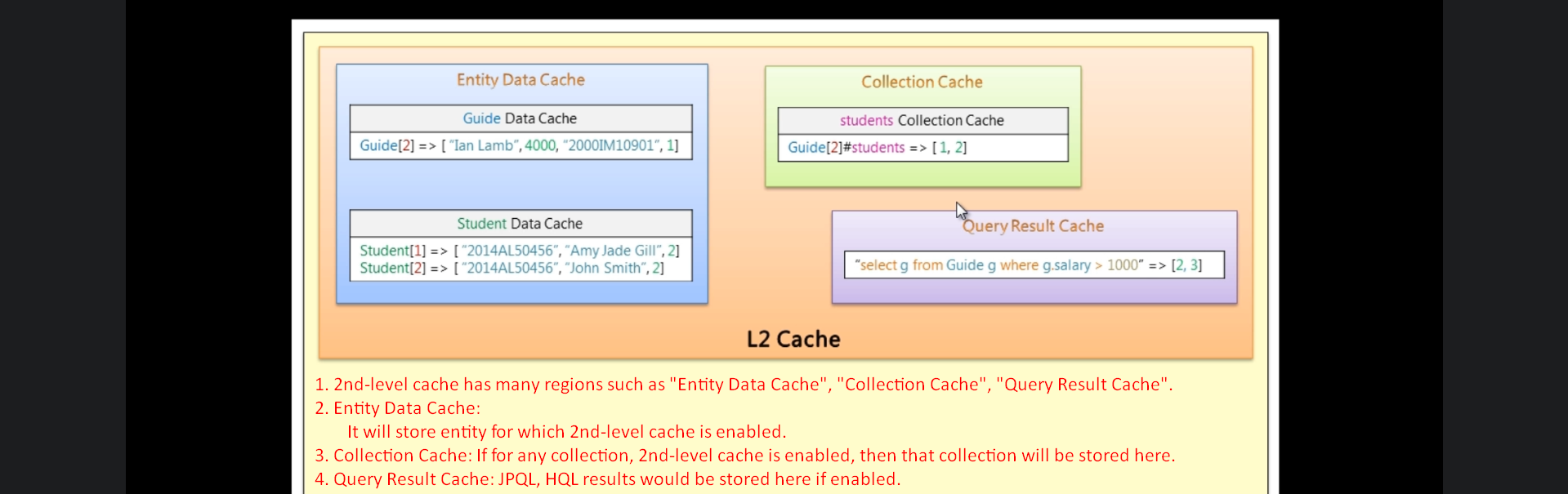
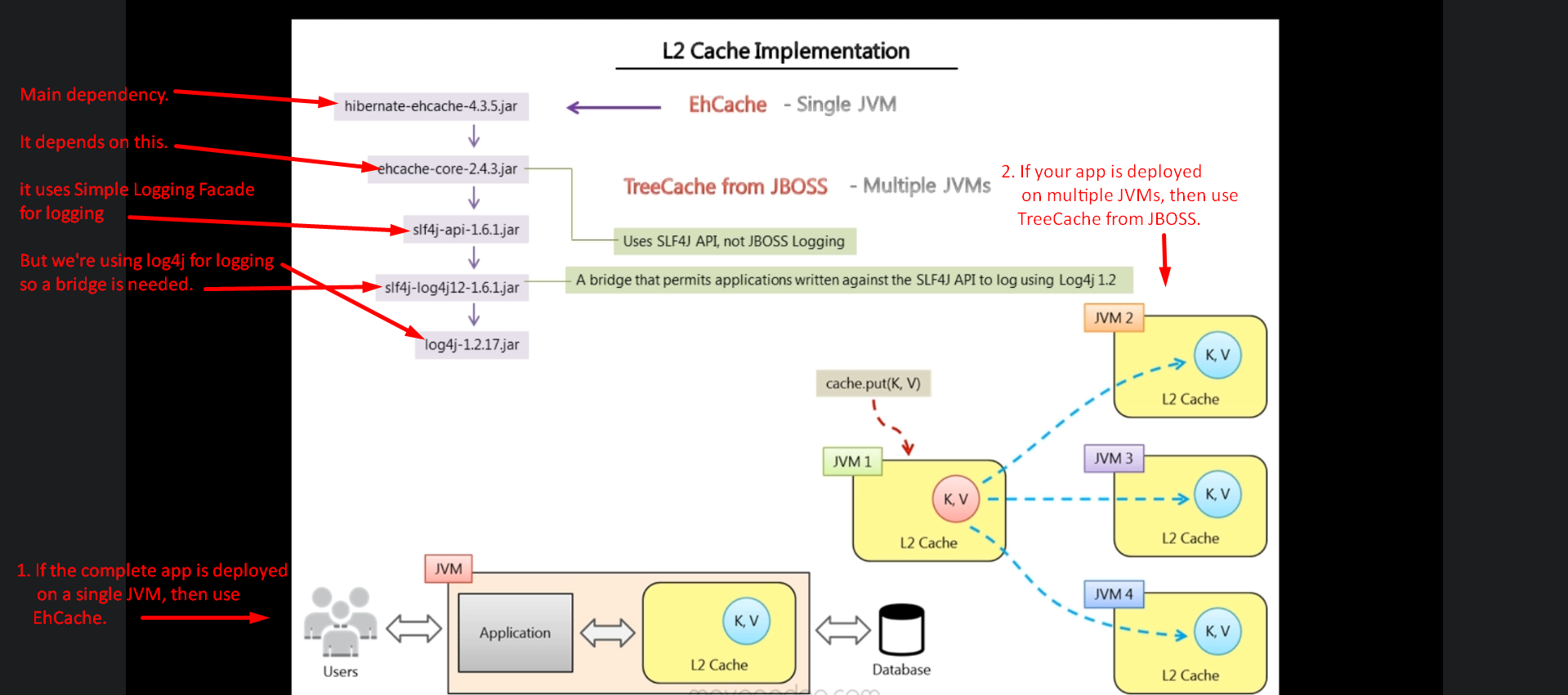
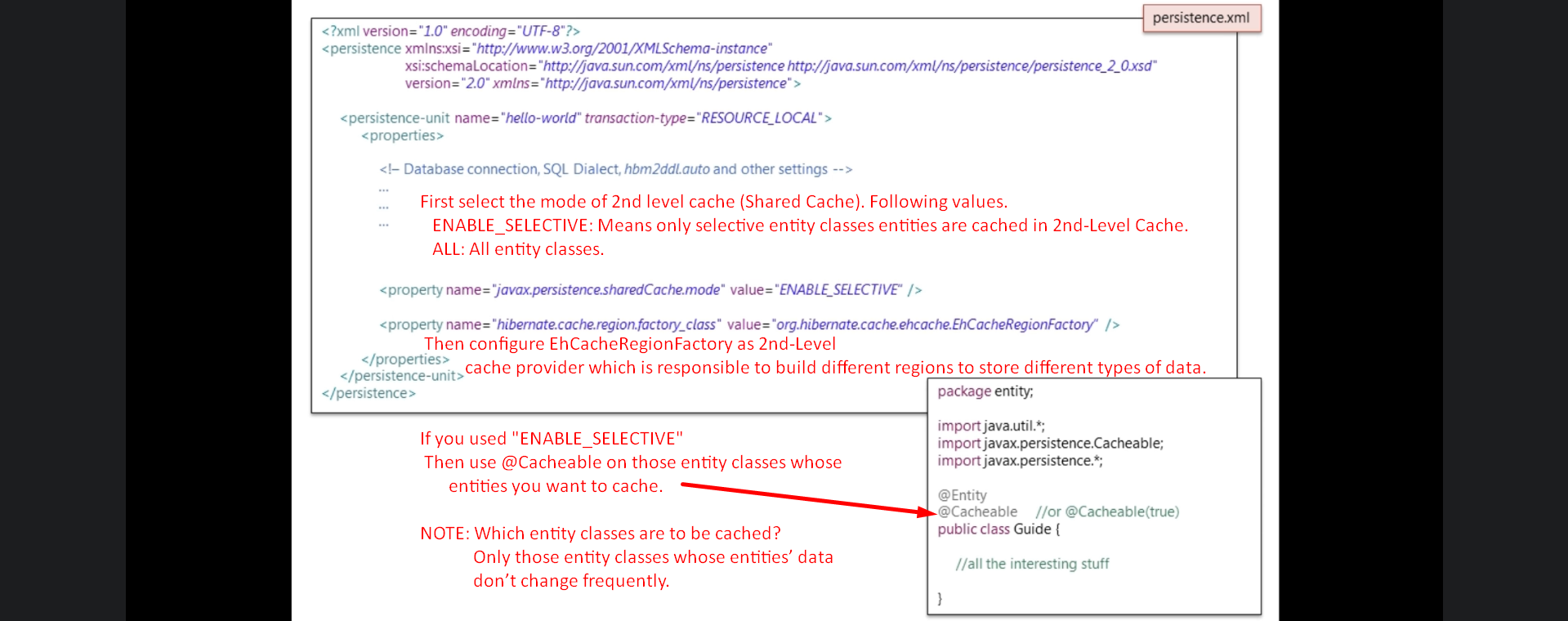
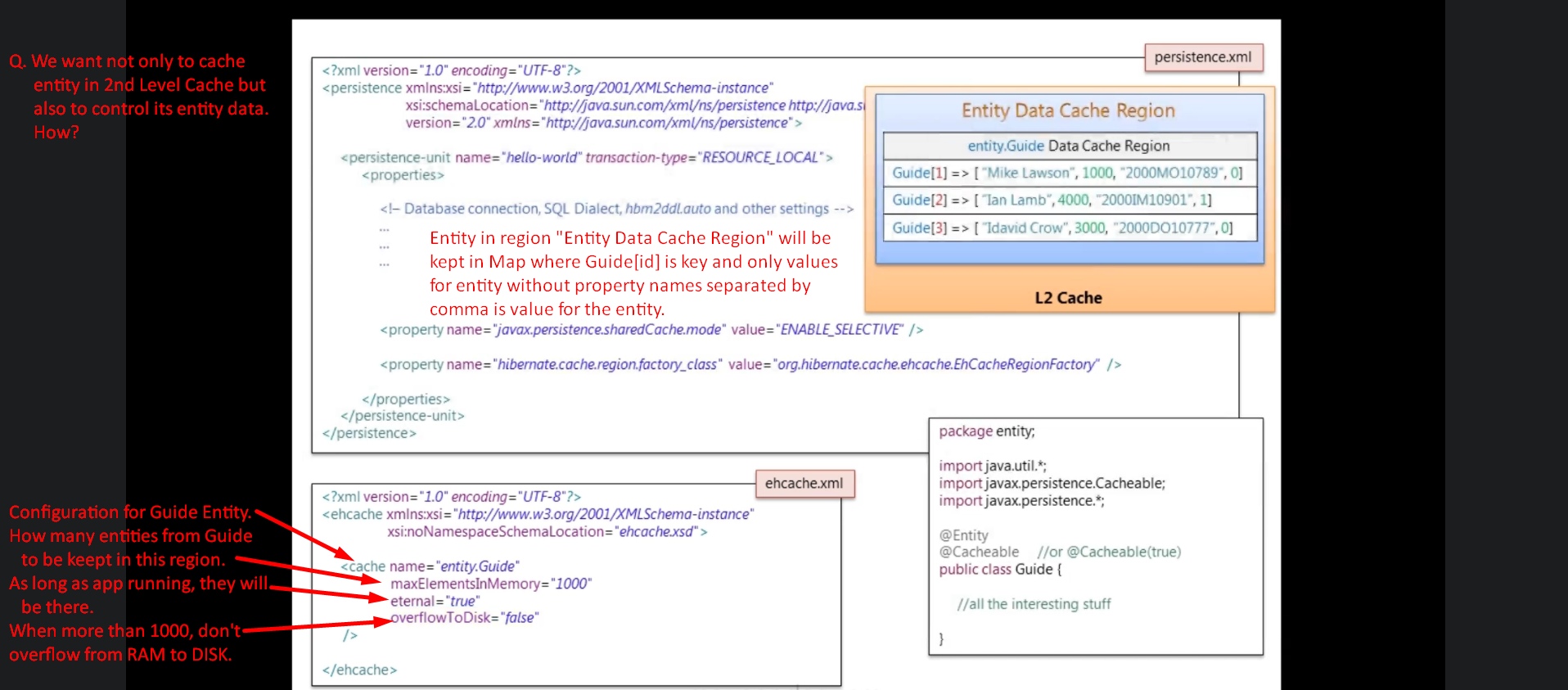
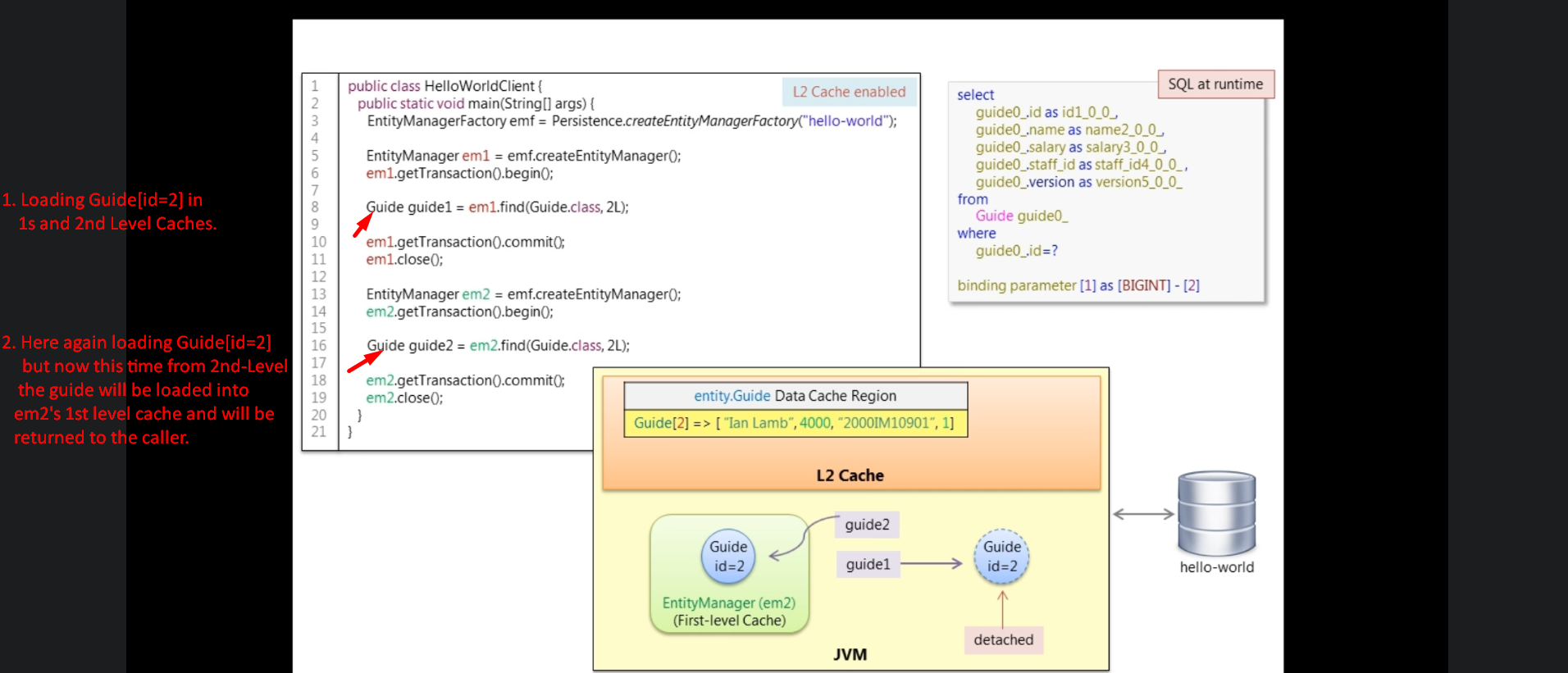
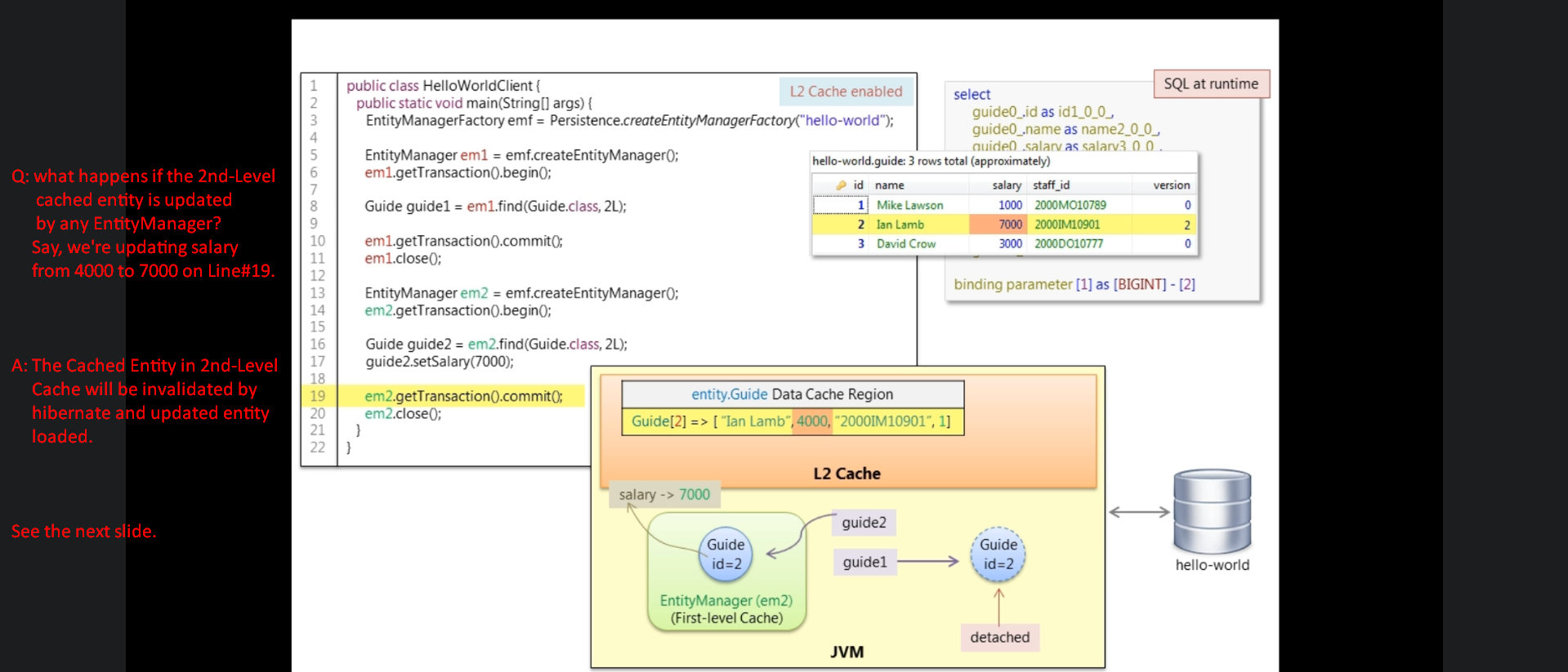
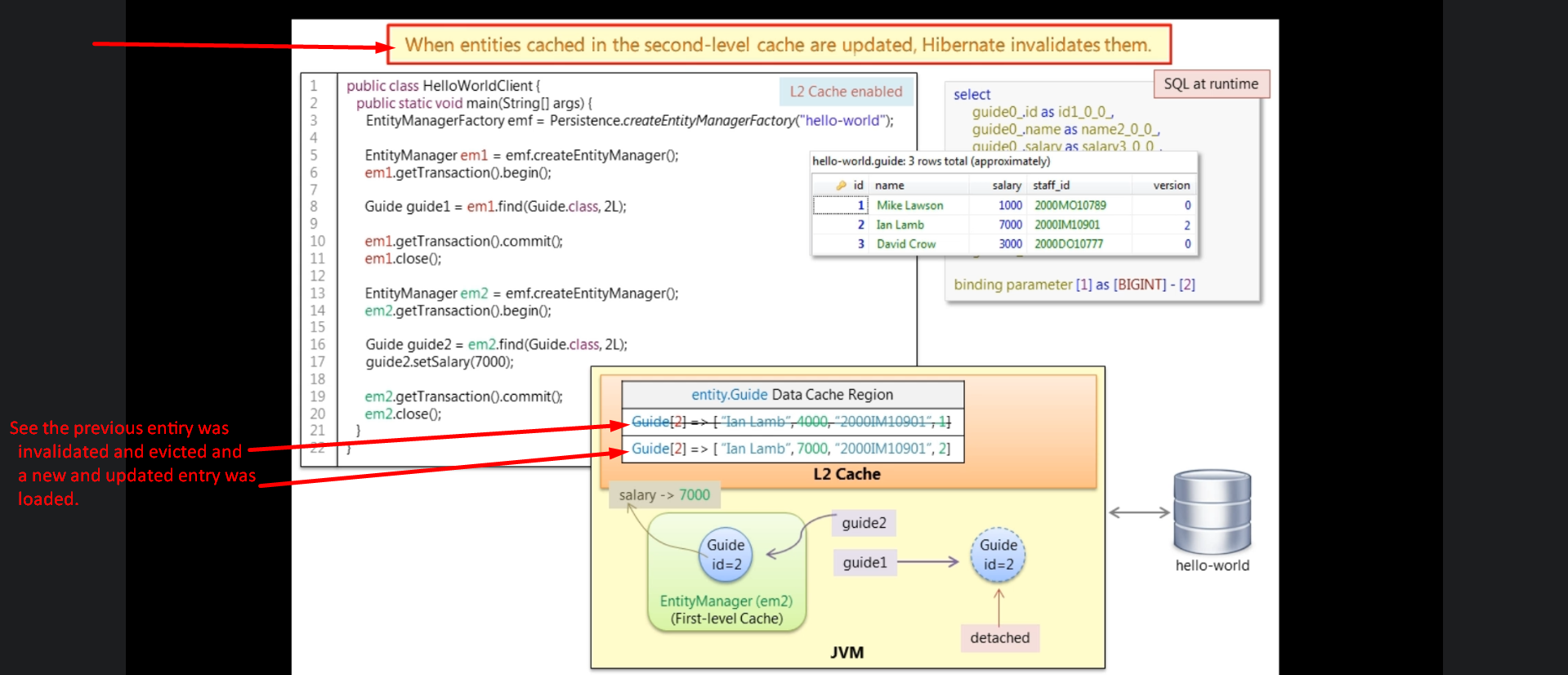
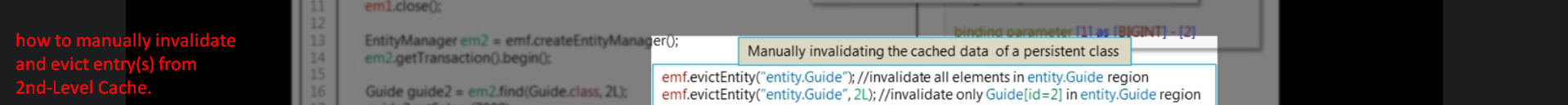
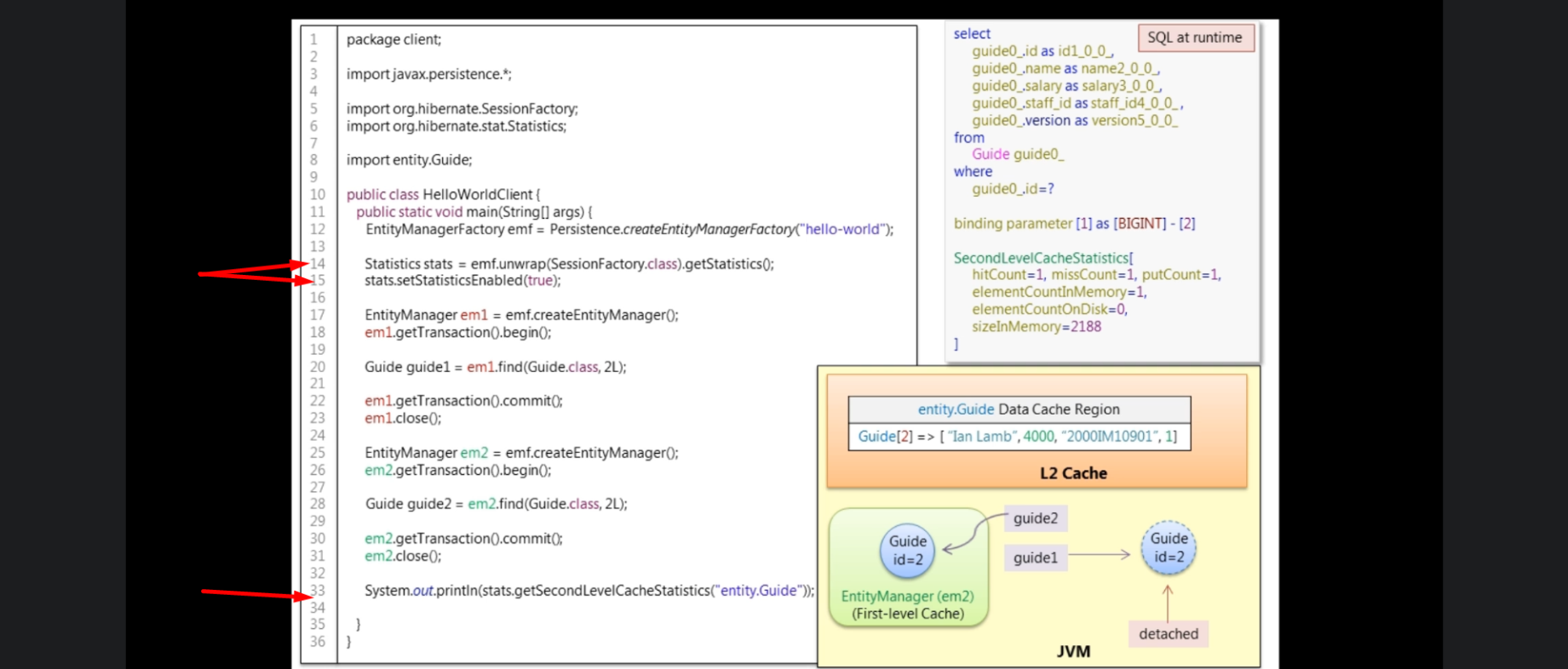
1. 
2. 
3. 
4. 
   1. First of all first level cache is looked into. If entity not found, then searching into 2nd-Level Cache.
   2. If not found even in 2nd-Level Cache, then SELECT is issued and entity is loaded in both caches.
   3. Cache Hit.
   4. 2nd-Level cache is managed by EntityManager Factory and that EntityManagerFactory will share this 2nd-Level Cache with its all EntityManager.   
      **NOTE**: One app can have EntityManagerFactory more than one. In that case 2nd-Level Cache by one EntityManagerFactory will not be shared with other EntityManagerFactory.
   5. 2nd-Level Cache is maintained by EntityManagerFactory and is shared with all EntityManagers created by that EntityManagerFactory.
   6. 
5. 
6. Let’s configure 2nd-Level Cache:
   1. First select the 2nd-Level cache provider means implementation.
   2. When using Hibernate as provider for JPA, it doesn’t provide any default 2nd-Level cache implementation as in JPA specification 2nd-level cache is optional optimization feature.
   3. You have to choose one implementation and then enable it.
   4. Many implementations in market.
   5. Most common implementation is 🡺 EhCache.
   6. **EhCache**: It will work only if your entire app is deployed on a single JVM.
   7. **TreeCache**: Use it from JBoss if your app is deployed on multiple virtual machines.
      1. It is a replicated cache where the 2nd-level cached is replicated/copied to each JVM where your app is running.
   8. To use EhCache, we need integration hibernate-ehcache-4.35.jar jar file in classpath.
      1. It depends on ehcache-core-2.4.3 jar.
         1. It uses slf4j (Simple Logging Façade for java) as the logging API not JBOS Logging
   9. 
7. Let’s configure 2nd-Level Cache;
   1. First select the mode of 2nd level cache (Shared Cache). Following values.
      1. **ENABLE\_SELECTIVE**: Means only selective entity classes entities are cached in 2nd-Level Cache.
         1. Then use @Cacheable on those entity classes whose entities you want to cache.  
            
      2. ALL: All entity classes.  
         
   2. **NOTE**: Which entity classes are to be cached?  
       Only those entity classes whose entities’ data don’t change frequently.
   3. Then configure EhCacheRegionFactory as 2nd-Level cache provider which is responsible to build different regions to store different types of data. 
8. When enabling 2nd-Level Cache for an entity, we want to control how data for that entity is to be cached.
9. 
10. 
11. 
12.  
13. Hibernate Statistics API:
    1. Can be used for analysis for 2nd-Level cache in Development Mode.
    2. It is not part of JPA specification.
    3. It comes from Hibernate Native API.
    4. 
14. How to handle concurrency when someone accessing entity in 2nd-Level Cache and other one modifying it?