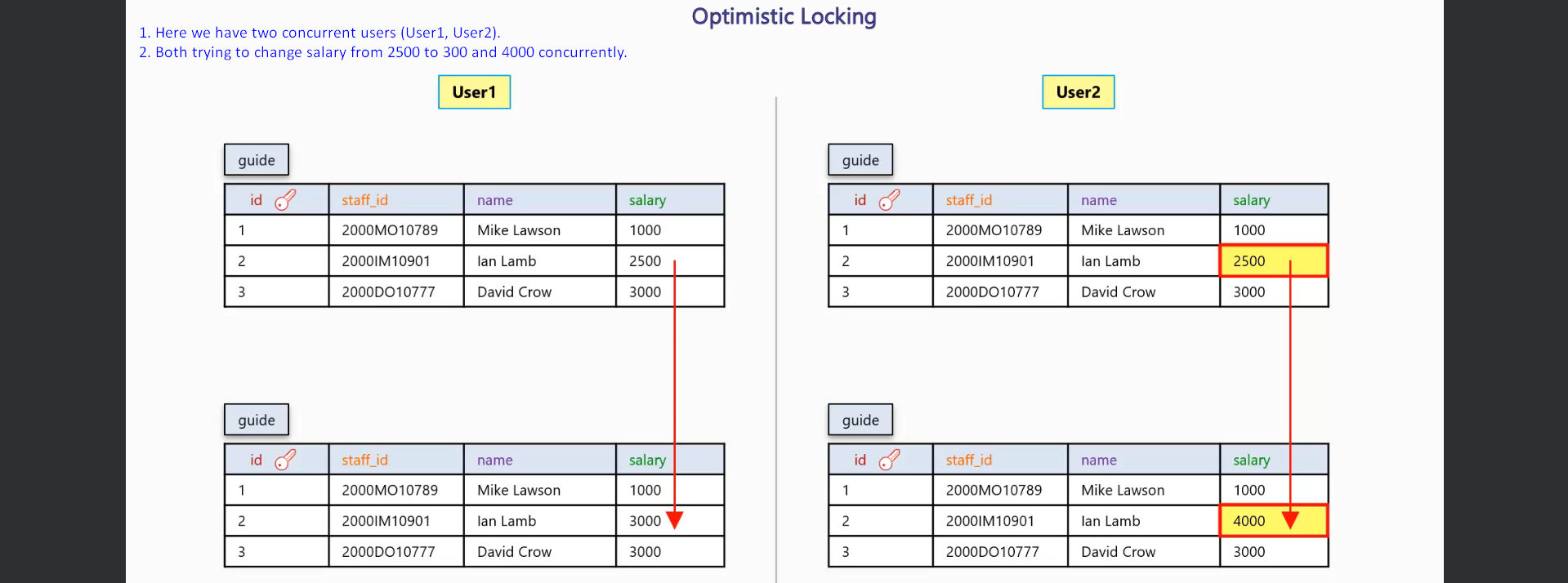
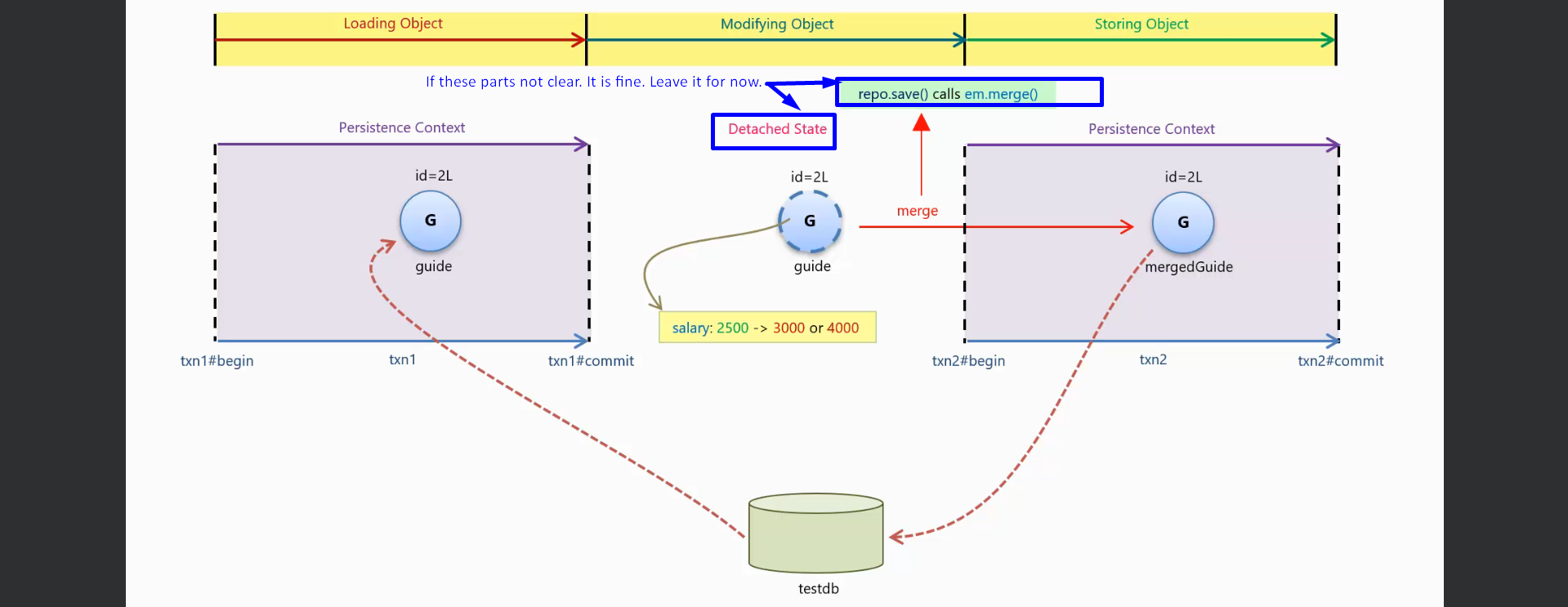
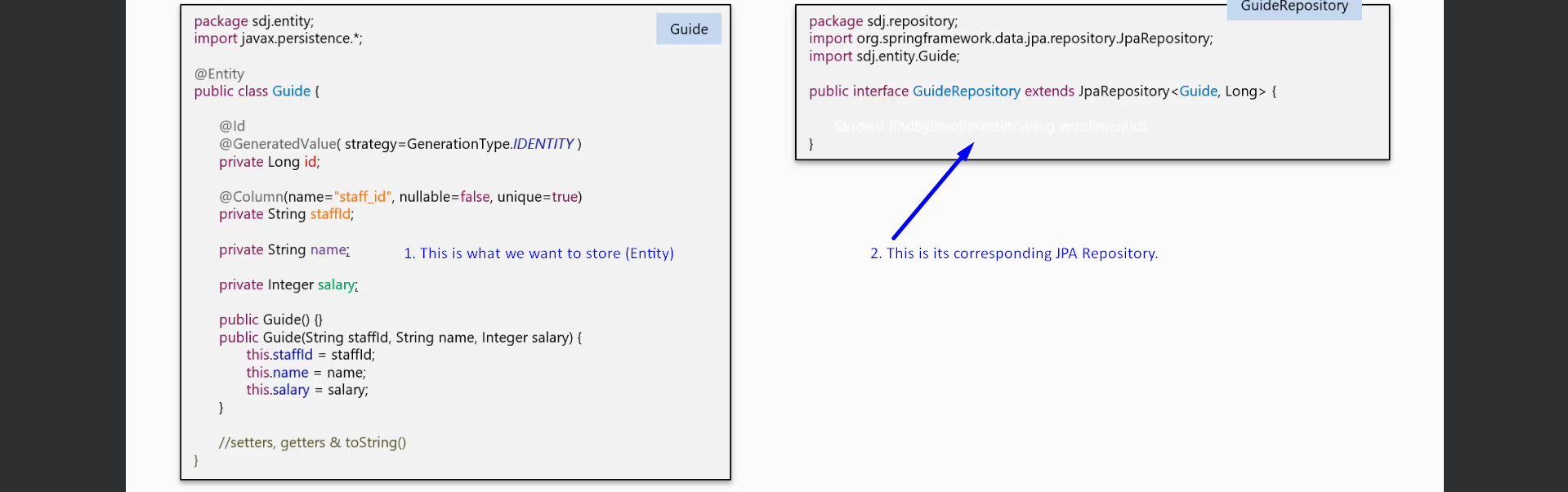
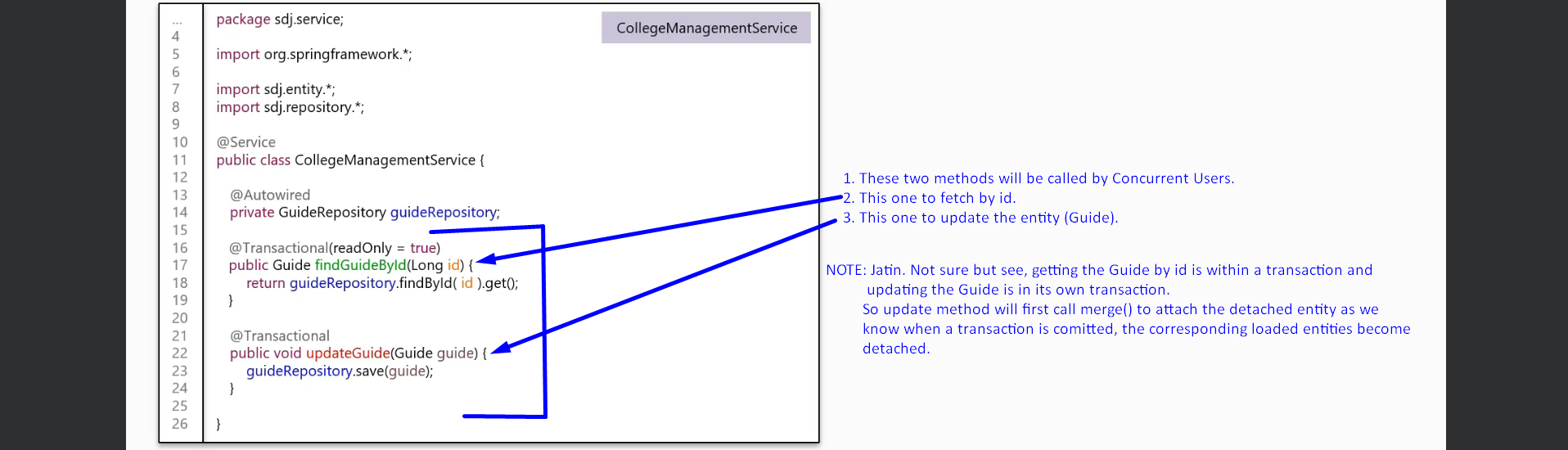
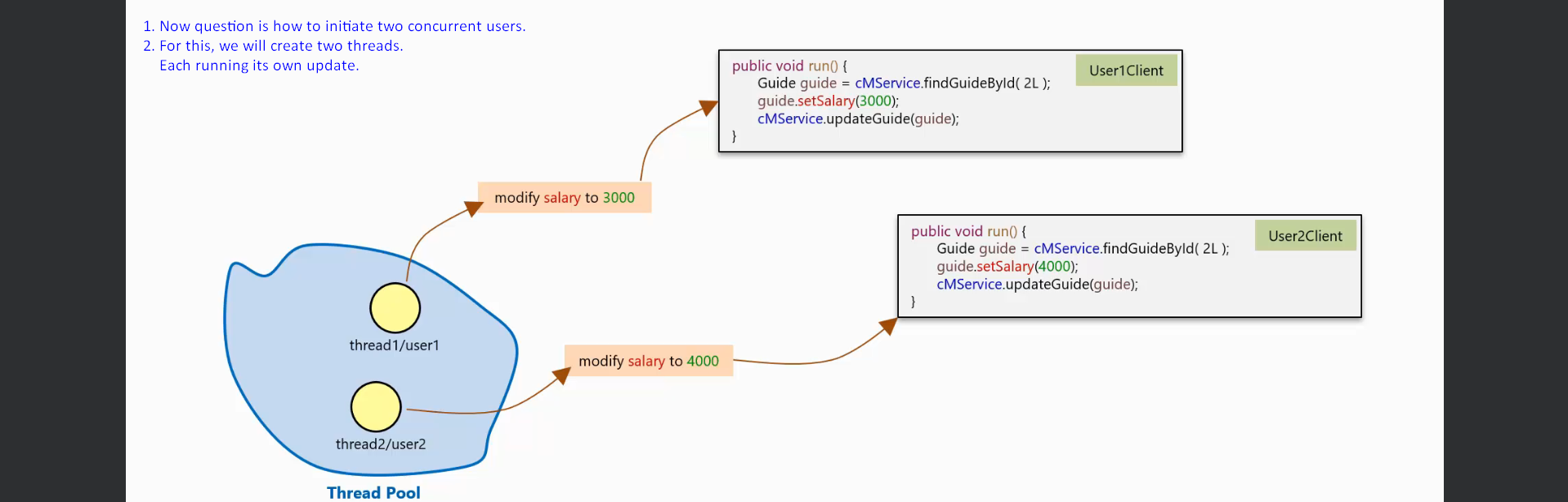
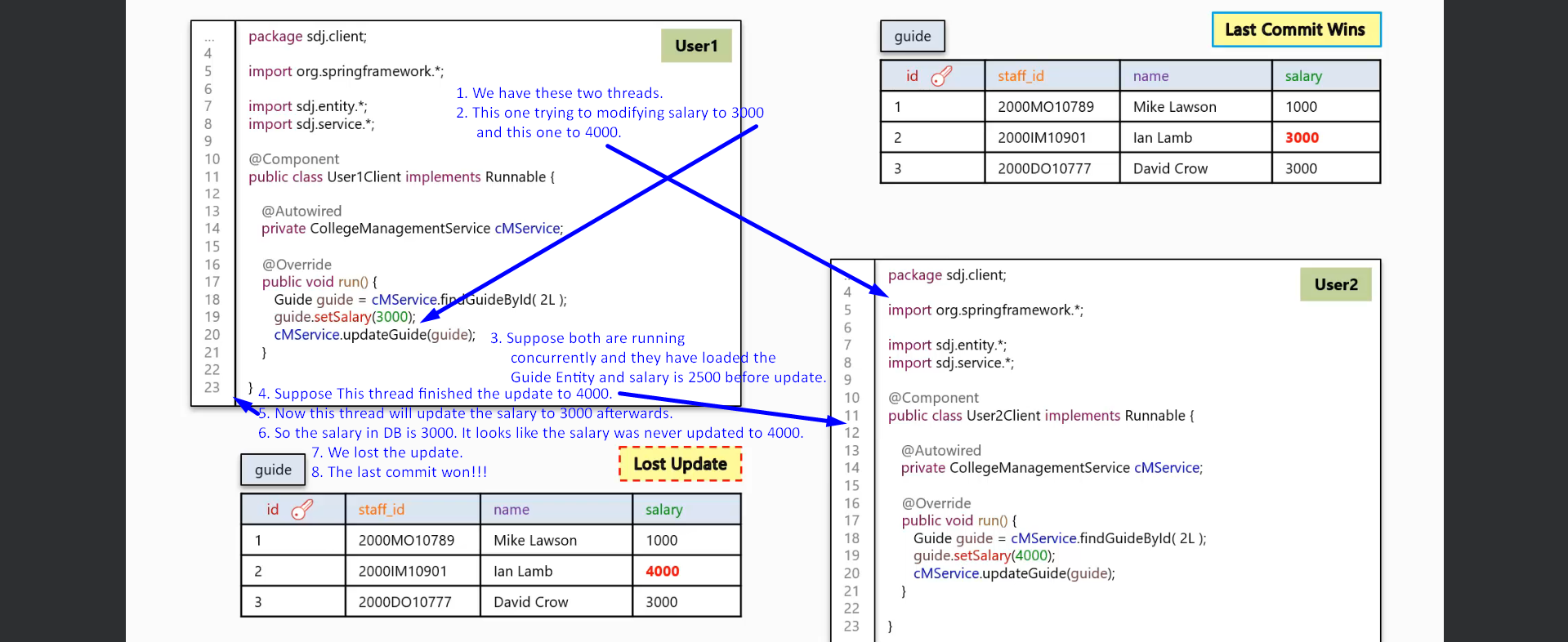
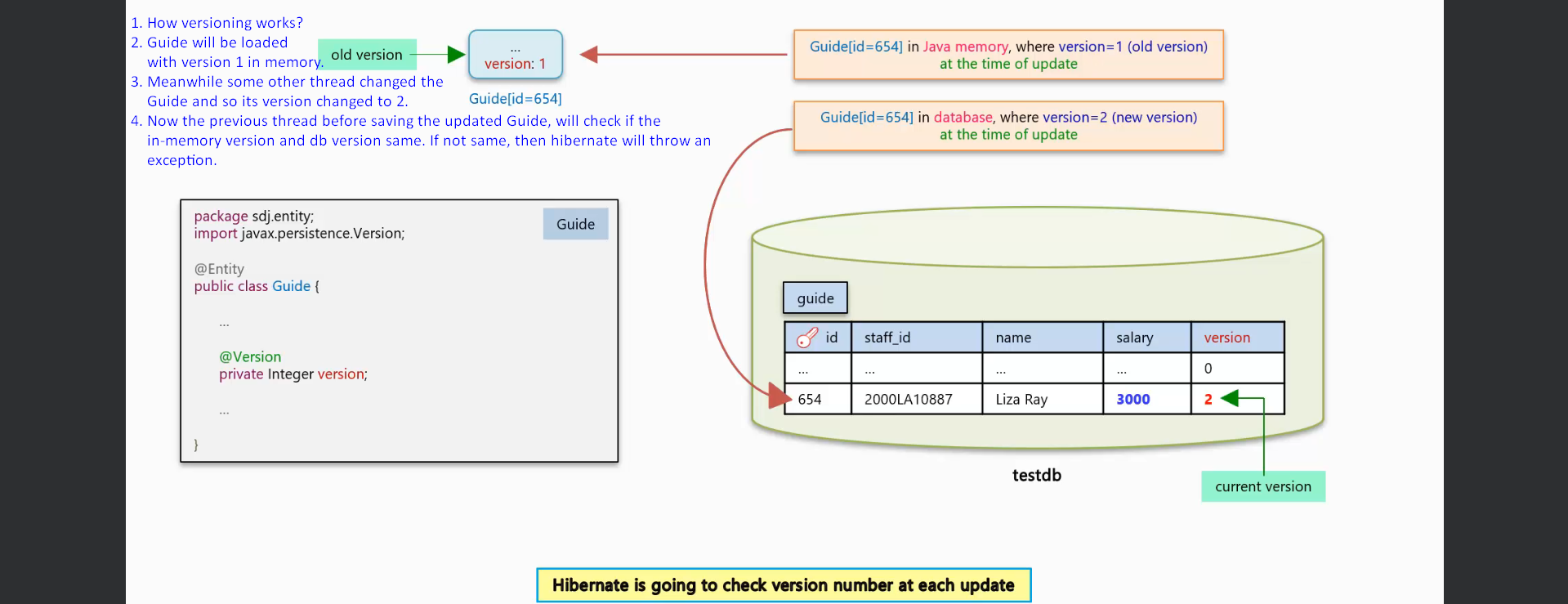
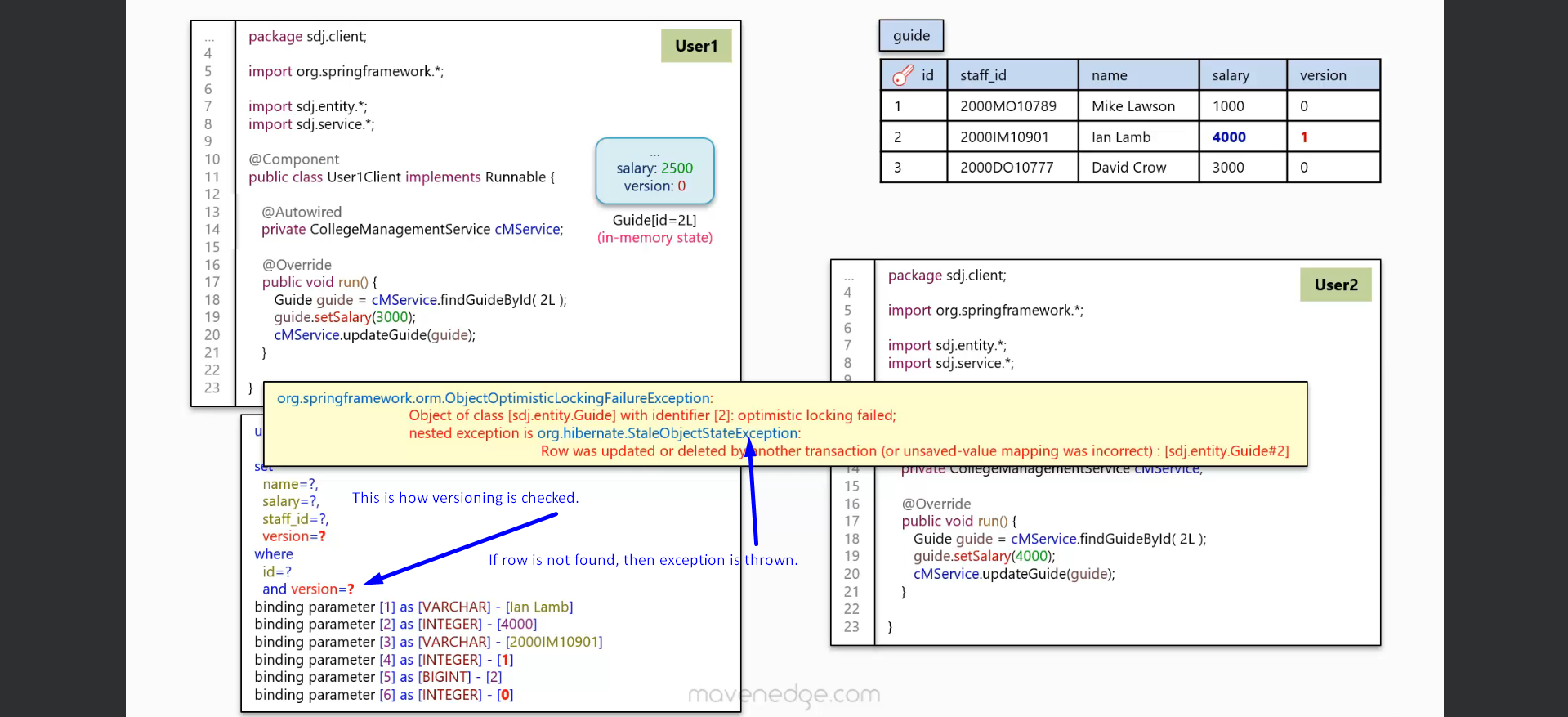
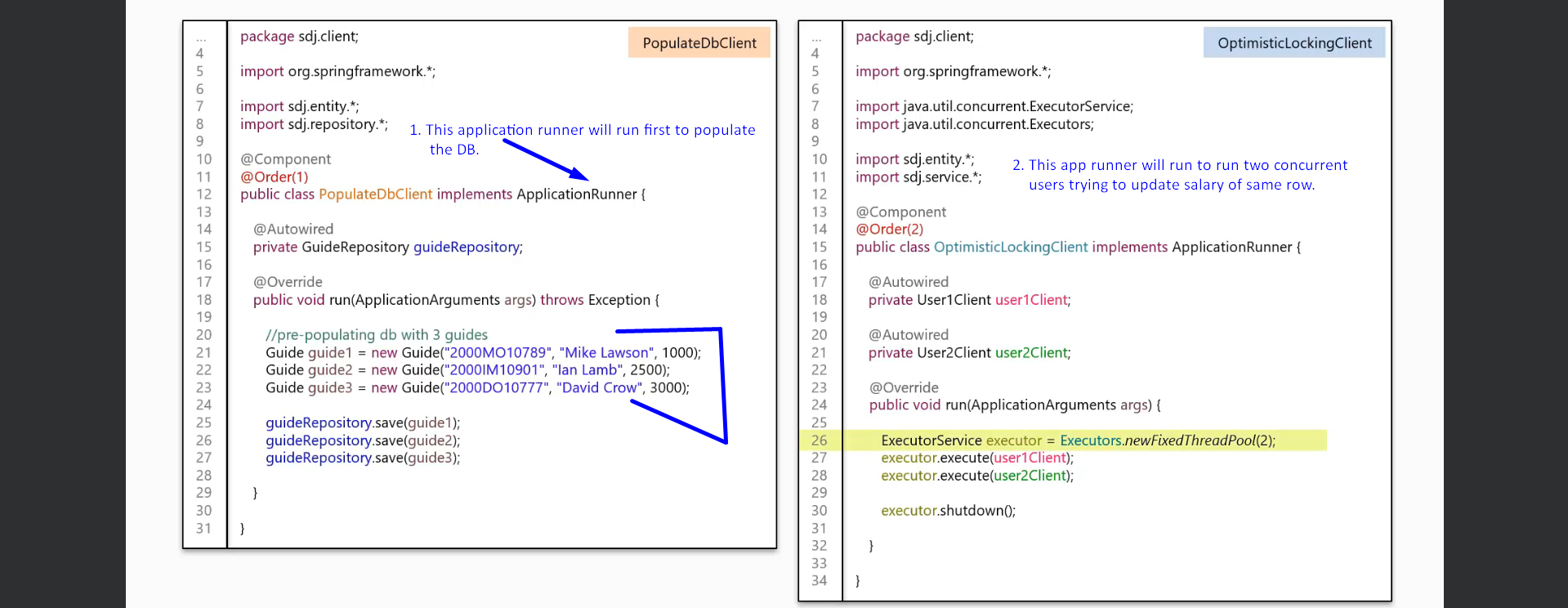
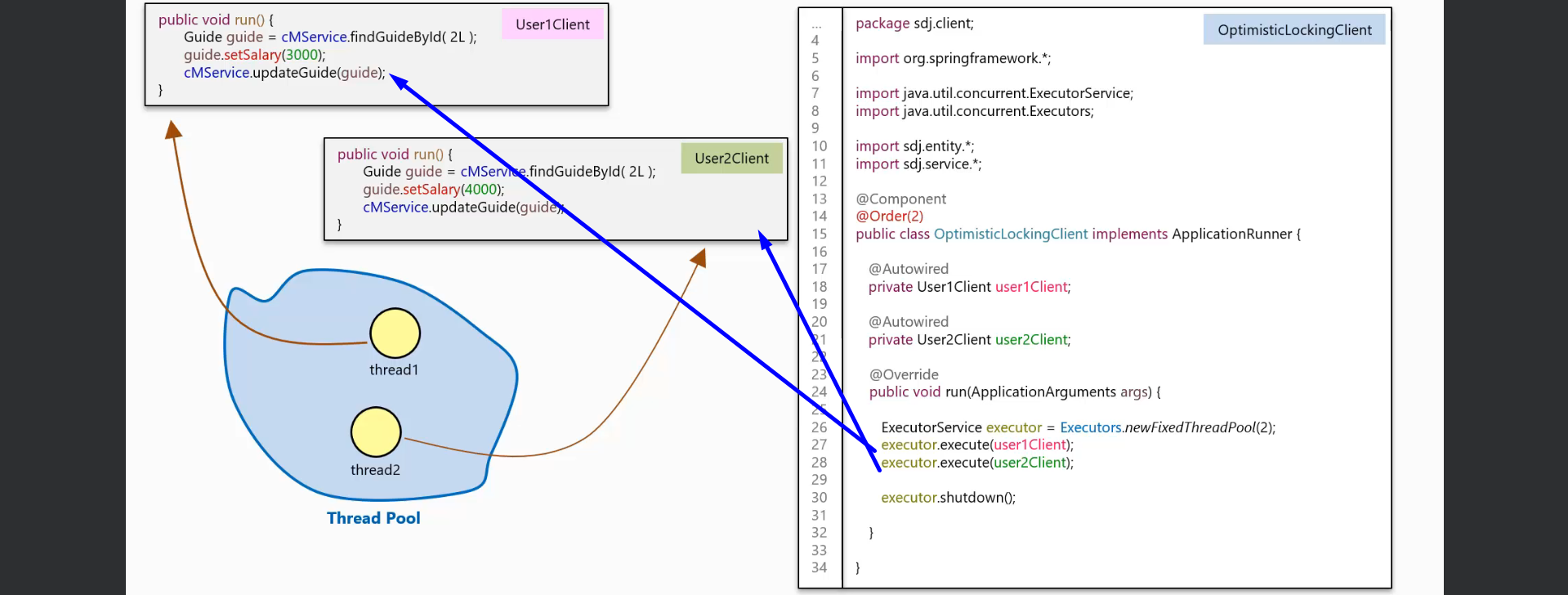
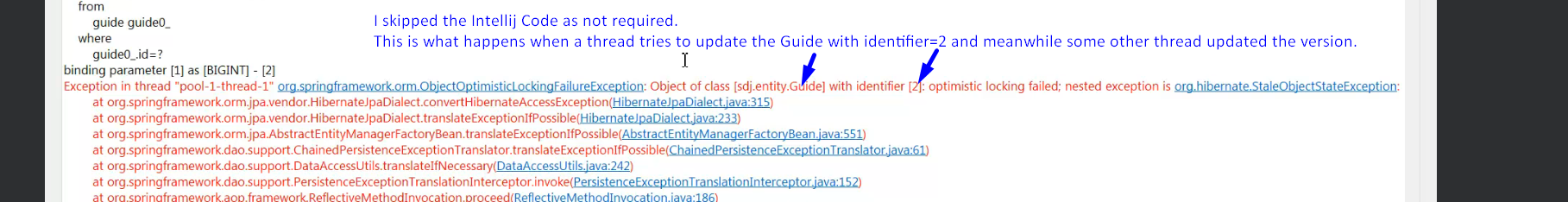
1. 
2. 
3. 
4. 
5. 
6. 
7. How do we prevent our app from having lost update?  
   Well, we tell user1 about the update made by user2 by displaying a message to him before user1 could persist its modifications.  
   We do that by the **Strategy called Versioning**.
8. How do we do versioning?
9. By introducing a new column named “version” that is updated every time, Guide entity is updated.  
   Hibernate automatically update the version number of the row whenever we update a row.
10. 
11. 
12. 
13. 
14. 
15. What we discussed about preventing an update loss is called **Optimistic Lock.**
16. Now the question is what is this **Optimistic Locking**?
17. **Answer**: Just official name of Versioning Strategy to prevent **Lost Updates**.  
    We call “**Optimistic”** because we are saying that we don’t expect a lost update to happen too often in our app.   
    In case such situation occurs, then we are going to handle that using the versioning.   
    Brief: We are very optimistic that we will not having any lost updates in our app and as we are very optimistic that we will not lose any updates, we are not using any DB locks to prevent such situations.  
    And by the term “Locking”, we give impression that with optimistic locking, we use DB locks. But actually, we don’t.  
    Optimistic Locking means no DB Locking.   
    It is the **pessimistic lock** where we actually use DB Locks.
18. 