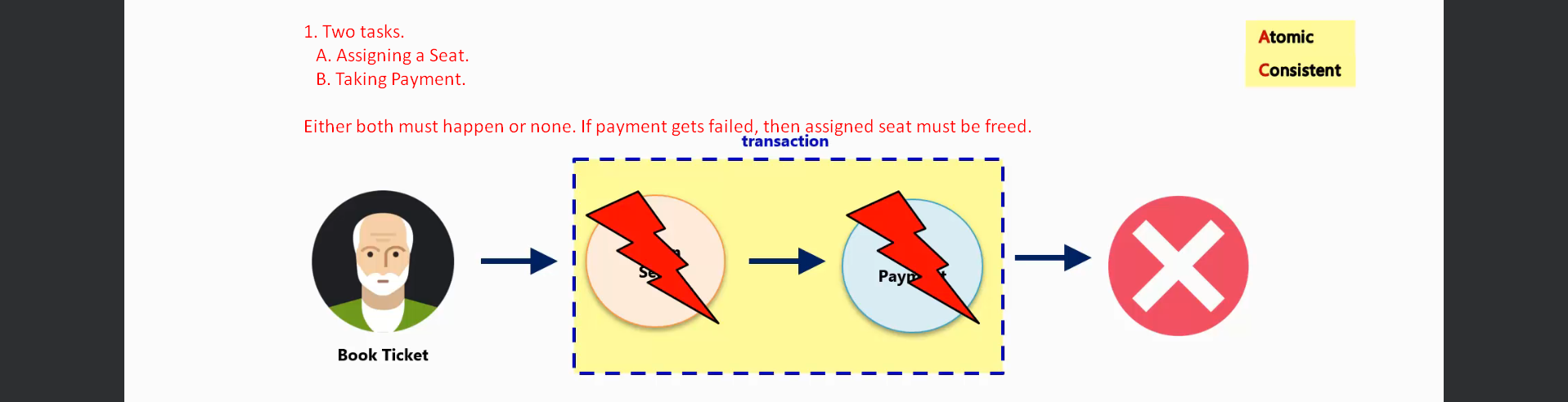
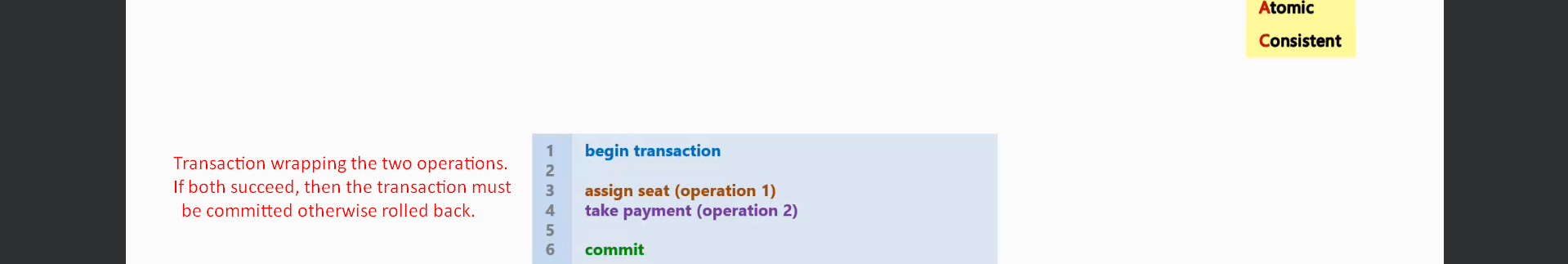
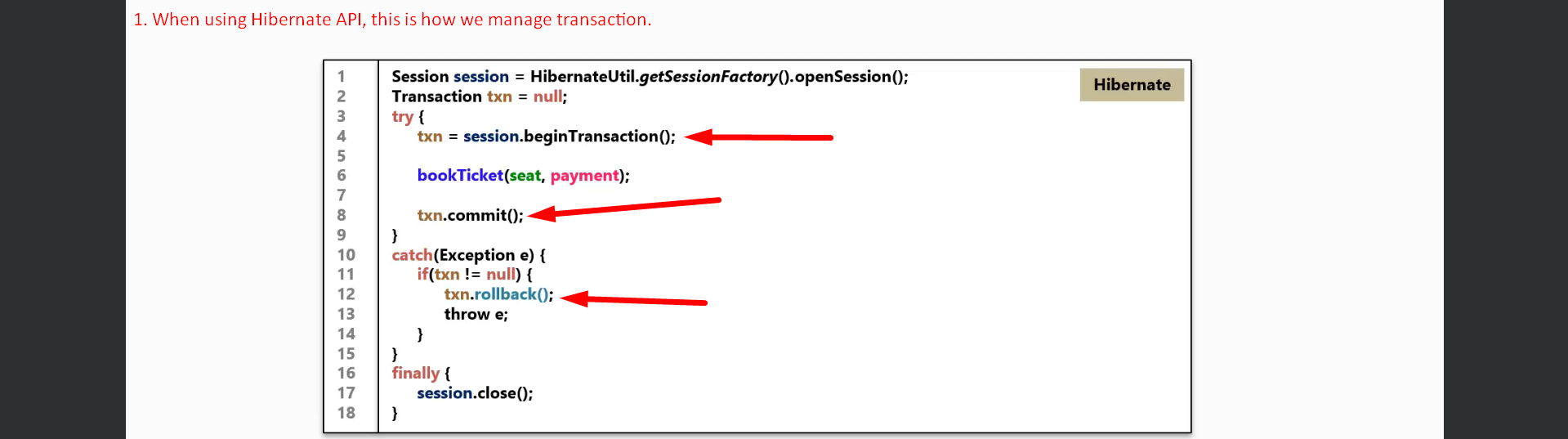
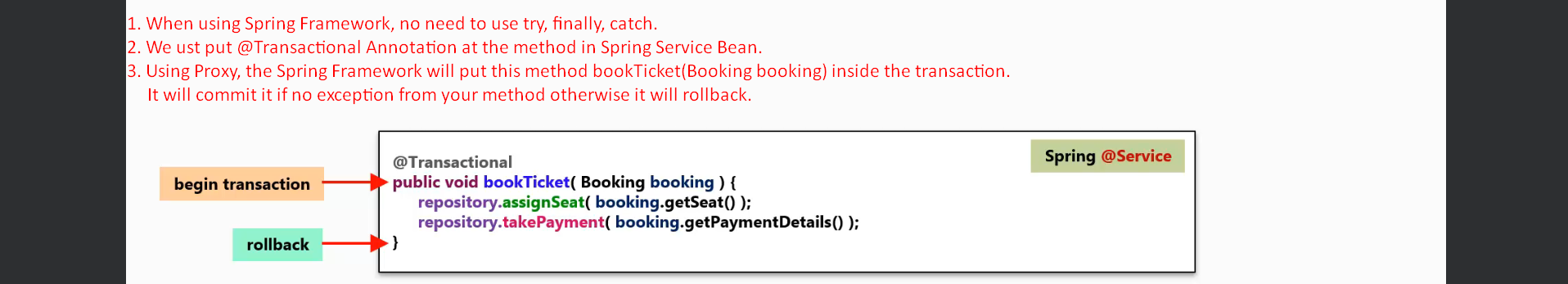
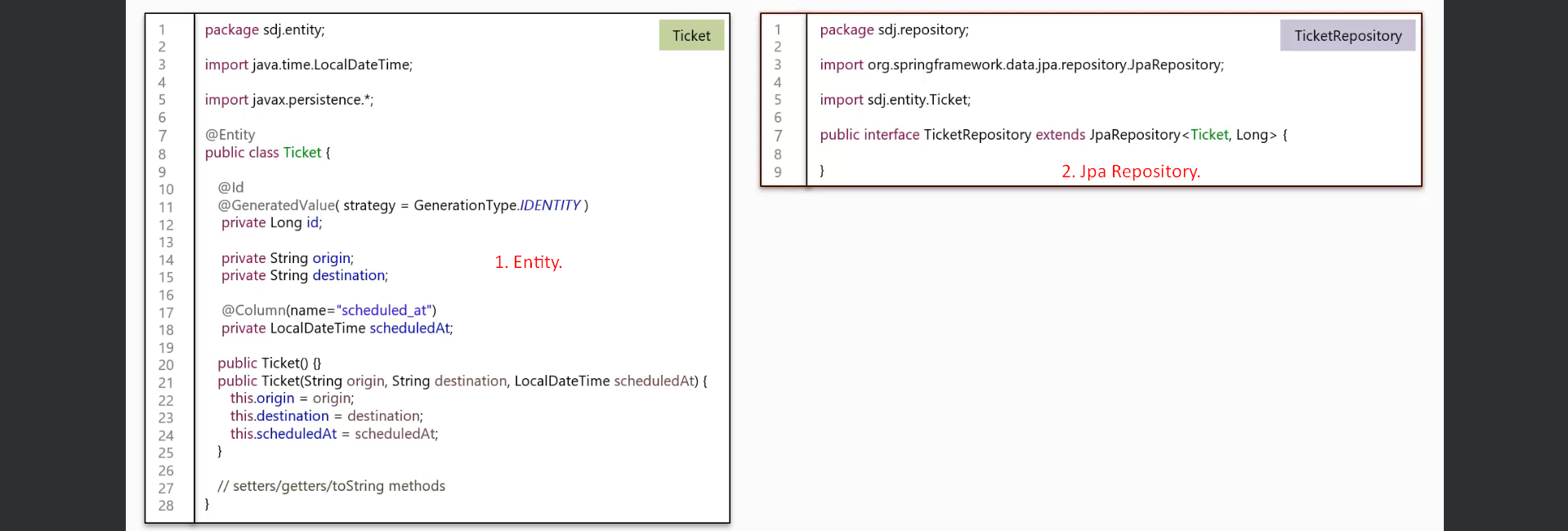
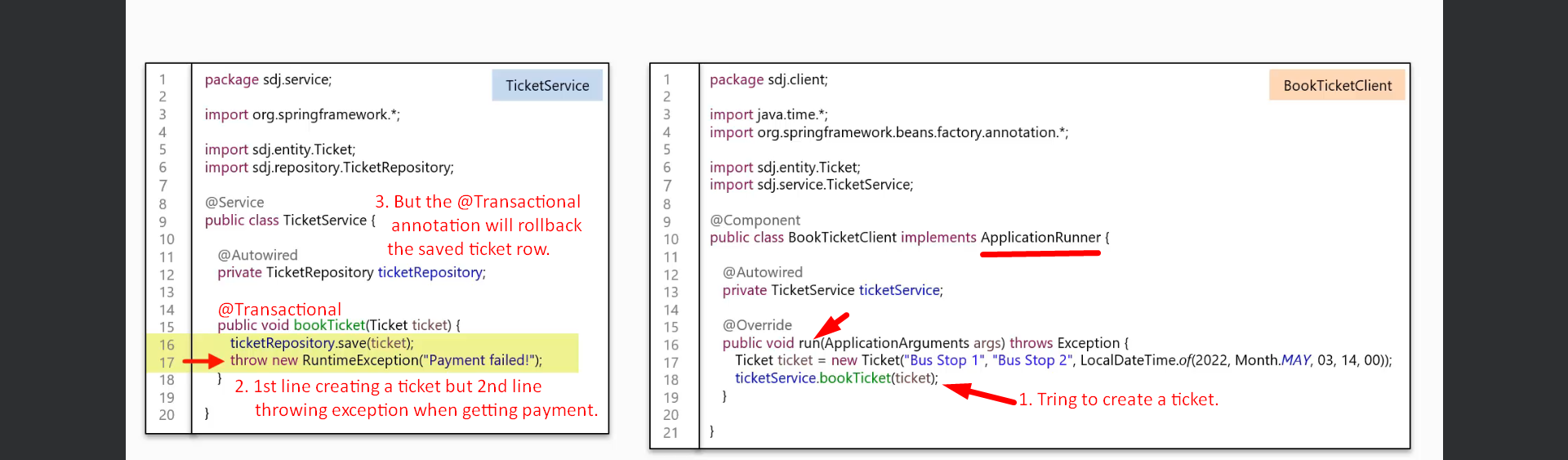
1. Before discussing how to handle transaction, let’s see what is Transaction.   
   
2. **Example**: Ticket Booking App for Bus Travelling.
3. 
4. If two transactions are running in parallel, then the effect of one transaction must not corrupt the state of another transaction.  
   In other word, one transaction must be **protected or isolated** from other transaction.  
   So, the transaction changes must be visible when it is committed.
5. Transaction must be durable.
6. If someone asks why a transaction is needed?
7. **Answer**: With it, the app will not function the way we want it to work.
8. Transaction provides us safety by four Guarantees called ACID (Atomic, Consistency, Isolation, Durability).
9. So, when performing business logic, a transaction helps us to avoid any **Data Corruption** in our DB.
10.   
    This is how we use a transaction to wrap a business method when using Hibernate.
11.   
    @Transactional Annotation gives transactional context to the method on which we put this annotation.
12. Concrete Example:  
      
      
    
13. Instructor also showcased it through Junit but I skipped as this was very fundamental.   
    He said we using MySQL but not H2 In-Memory DB as H2 will be gone when exception is thrown as app shuts down.