Locking in hibernate
==========
If multiple apps or threads simultaneoulsy accessing and manipulating the records there is a possibility of getting
concurrency problem.
To Avoid this problem we need to use "Locking " of a record in hibernate.

Hibernate supports 2 levels of Locking
      a. Optimistic Locking
                => It allows secondthread simultaneously to access and modify the record, then first thread notices the
                     modification and throws "Exception".
                => To enable this locking we need to use "@Version" in our application.
                => if we use @Version then automatically optimisitic locking will be achieved.

      b. Pesimistic Locking
                => It will allow FirstThread to Lock the record itself,so if the second thread tries to access and modify the
                     record then it should throw "Exception".
                => To enable this locking we need to use
                              session.get(Class c,Serializable s,LOCKMODE.UPGRADE_NOWAIT) as the third argument value
                                    refer: HBOptimisticLockingApp, HBPesimisticLockingApp

Mapping in hibernate
==========
   In realtime applications, we use mapping to link two classes and these linking at the database side will happen in the form
   of primary-key foreign-key relationship.
   At the java side, we can link 2 classes through "Association".
   At the database side we don't have these linking,but we have something called as "Primary-Foreing" key relationship.
   To support this feature at the hibernate side we have "Mapping".
   There are 4 types of hibernate mapping
      a. One-One Association Mapping
      b. One-Many Association Mapping
      c. Many-One Association Mapping
      d. Many-Many Association Mapping

UniDirectional

One-One Association Mapping

It refers to relationship b/w 2 entities where one instance of one entity should be mapped with exactly one instance of
another entity.
                eg: One Employee has One Account
Annotation used is :: @OneToOne(cascade = CascadeType.ALL)
cascade specifies, if we delete employee table automatically account table also should be deleted

One-Many Association Mapping

It refers to relationship b/w 2 entities where one instance of one entity should be mapped with multiple instances of
another entity.

eg: Single Department has Multiple Employees
Annotation used is :: @OneToMany(cascade = CascadeType.ALL)

Many-One Association Mapping

It refers to relationship b/w 2 entities where mulitple instance of an entity should be mapped with exactly one instance of
another entity.
eg: Multiple Students have joined with Single Branch
Annotation used is :: @ManytoOne(cascade = CascadeType.ALL)

Many-Many Association Mapping

It refers to relationship b/w 2 entities where mulitple instances of an entity should be mapped with multiple instances of
another entity.
eg: Multiple Students have joined with Mulitple Courses.
Annotation used is :: @ManytoMany(cascade = CascadeType.ALL)

refer::
HB-Many-To-Many-Mapping
HB-Many-To-One-Mapping
HB-One-OneMapping
HB-One-To-Many-Mapping

Learn on ur own
---
1. Hiberante filters
2. Mapping(bi-directional)

Next Week

Spring core would start(SAT-SUN)
SpringBoot
    a. SpringMVC
    b. SpringDataJpa and SpringMongoDB
    c. SpringJDBC/SpringORM
    d. SpringAOP
    e. SpringRest
    f. Spring Mail
    g. SpringSecurity
    h. Microservices and deployment tools