Quiz 3

1. **What is the difference between JDBC Statement and PreparedStatement?**
   ("……" +id) -> PreparedStatement
   ("……+?") -> JDBC Statement
   Prepared statement may cause exceptions, it is not safe.

2. **When using Spring JdbcTemplate query function to retrieve data from database, how do you convert the result into Java model class object?**
   RowMapper and extract data from result set.

3. **Why do we need NamedParameterJdbcTemplate since there is already JdbcTemplate?**
   NamedParameterJdbcTemplate will give a name for these parameters in sql so that they would never get confused of these parameters.

4. **Given the following Database Table, create the corresponding hibernate/JPA mapping class.**

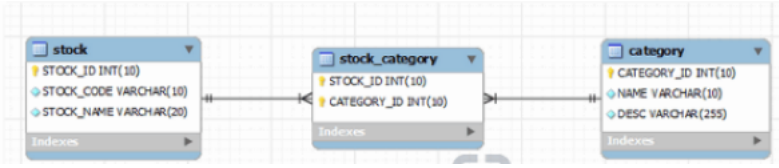
my_wp_postmeta

Field	Type	Null	Key	Default	Extra
meta_id	bigint(20) unsigned	NO	PRI		auto_increment
post_id	bigint(20) unsigned	NO	MUL	0	
meta_key	varchar(255)	YES	MUL		
meta_value	longtext	YES			

```
@Entity
@Table(name="my_wp_postmeta")
public class MyWpPostmeta {
    @Id
    @GeneratedValue
    private Long meta_id;

    @column(name="post_id")
    private Long postId;

    @column(name="meta_key")
    private String metaKey;

    @column(name="meta_value")
    private String metaValue;
}
```

5. **How do you map a composite primary key using ORM? Describe the details.**
   1) using @Embeddable to write a class of primary key and implements the Serializable interface.
   2) using @Entity to create a mapping class which related to the table.

6. **Given the following relationship in database, is this a "one-to-one", "one-to-many" or "many-to-many" relation? Write Hibernate mapping classes to reflect this relation with proper annotation.**



this is a "many-to-many" relation.

```
@Entity
@Table(name="stock")
public class Stock implements Serializable {
    private int stockId;
    private String stockCode;
    private String stockName;
    private Set<Category> category = new HashSet<Category>();

    @ManyToMany(fetch=FetchType.LAZY, cascadeType.All)
    @JoinTable(name="stock_category",
                joinColumns={@joinColumn(name="STOCK_ID")},
               inverseJoinColumns={@joinColumn(name="CATEGORY_ID")})

   public Set<Category> getCategory() {

        return this.category;

   }

}


@Entity
@Table(name="category")
public class Category implements Serializable {
    private int categoryId;
    private String name;
    private String DESC;
    private Set<Stock> stock = new HashSet<Stock>();
```

```
    @ManyToMany(fetch=FetchType.LAZY, mappedBy="category")
    public Set<Category> getStock() {
        return this.stock;
    }
}
```

7. **What is the difference between load() and get() in hibernate?**
   both of them be used to get data from a database but when the target object not exist in the database, load() method will throw objectNotFoundException while get() method will return null.

8. **List the Persistence context in hibernate and explain the meaning of them**.
   Transient: new an Object
   Persistent: data must be the same between database and hibernate
   persist()/save()/get()/load -> enter into persistent
   detached: evict()/close()/clear()
   removed: delete()/remove()

9. **Suppose you are given a transaction with methods: tx.beginTransaction(), tx.commit() and tx.rollback(), and there is another method to update data into database: session.save(user). Write a template code block to reflect this transaction flow.**

```
Session session = sessionFactory.openSession();
Transaction tx = session. beginTransaction()
try{
    // do something
    tx.commit();
    session.save(user);
} catch (Exception e) {
    if (tx.getTransaction().getStatus() == TransactionStatus.ACTIVE ||
TransactionStatus.MARKED_ROLLBACK) {
        tx.rollback();
    }
} finally {
    session.close();
    sessionFactory. close();
}
```

10. **You are given a "product" table in database. Now you need to read the product table and update the product "amount" field. In the meaning time, there is another thread trying to update the product amount field as well. How do you make sure that the field you are trying to update is not already updated by the other thread? Describe the Hibernate Option.**
    use @version, which could be a timestamp.

11. **Explain the Hibernate cache mechanism.**
    Hibernate use 2 level cache:
    1st level cache stays in session level whose purpose is to minimize database visit
    2nd level cache stays in sessionFactory level whose purpose is to cross session use.

12. **What is hibernate query cache?**
    hibernate use query cache to store query results. If a query under execution has previously cached results, then no SQL statement is sent to the database. Instead the query results are retrieved from the query cache, and then the cached entity identifiers are used to access the second level cache.

13. **What is hibernate Fetch Type and Cascade?**
    FetchType is a property used to define fetching strategies, that is LAZY/EAGER determines whether the related entity should be initialized only on demand or right away when the current entity is initialized.
    CascadeType is a property used to define cascading in a relationship between a parent and a child.

14. **What is the difference between CrudRepository and JpaRepository?**
    CrudRepository returns Iterable while JpaRepository returns a list.

15. **Suppose front end is trying to retrieve large amount of data from your rest api, how do you handle it so that the front end will not feel too much delay in back end response?**
    Pagination extends PagingandSorting Repository.

16. **Write a jpa repository to implement query: select * from user where firstname=? and age is not null.**
    *@Query(value=" select * from user where firstname=:fn and age is not null")*

17. **Given the following service class, implement a logic to print "get method is triggered" whenever the getUser() or getAnotherUser() method is called. Add all the necessary annotations.**

```
public class UserService{

        public String getUser(int id){
                throw new NoUserException();
        }

        public String getAnotherUser(User u){
                return u;
        }
}
```

18. **Following previous question, how do you handle the getUser() exception in controller level?**

    @ExceptionHandler

19. **When spring handle transaction for you, what is the annotation to add on top of your class or method?**

    @Trancactional

20. **Explain the meaning and common values of "propagation" and "isolation".**

    "propagation" defines how transactions relate to each other.

    "isolation" defines the data contract between transactions.

21. **If you want to cancel the transaction whenever the "CustomException" is occurred, what attribute to you need to set and what value to give to that attribute?**

    rollbackfor=" CustomException.class"

22. **Describe what is CICD and the tool to handle CICD pipeline.**