Part B - Questions

1. ● What is JPA?
2. ● What is Hibernate?
3. ● What is the difference between JPA and Hibernate?
4. ● What is RDBMS?
5. ● What is the difference between RDBMS and Hibernate?
6. ● What are the advantages and disadvantages of Hibernate?
7. ● Draw the object/entity state diagram and explain how to change the statuses using methods.
8. ● What is ORM?
9. ● Explain the implementation of Hibernate and its methods.
10. ● What is lazy and eager fetching?
11. ● What are the methods used in eager and lazy fetching?
12. ● What are the two types of configuration in Hibernate's SessionFactory?
13. ● Explain the three types of relationships using "inverse and owning".
14. ● On which side is the "mapped by" attribute added?
15. ● What cascade types can be used in Hibernate, and why do we use them?
16. ● What are the query types used in Hibernate?
17. ● What are the differences between HQL and Native SQL?
18. ● Explain first-level and second-level cache.
19. ● Discuss the importance of password encryption and different algorithms used for secure password storage.

Answers

1. It is a specification in Java for managing relational data in applications using object-relational mapping (ORM). It defines how Java objects (entities) map to database tables and provides tools to interact with the database in an object-oriented way.

2. Hibernate is an Object-Relational Mapping (ORM) tool for Java. It simplifies the process of working with a relational database by allowing developers to interact with database tables using Java objects instead of writing complex SQL queries.

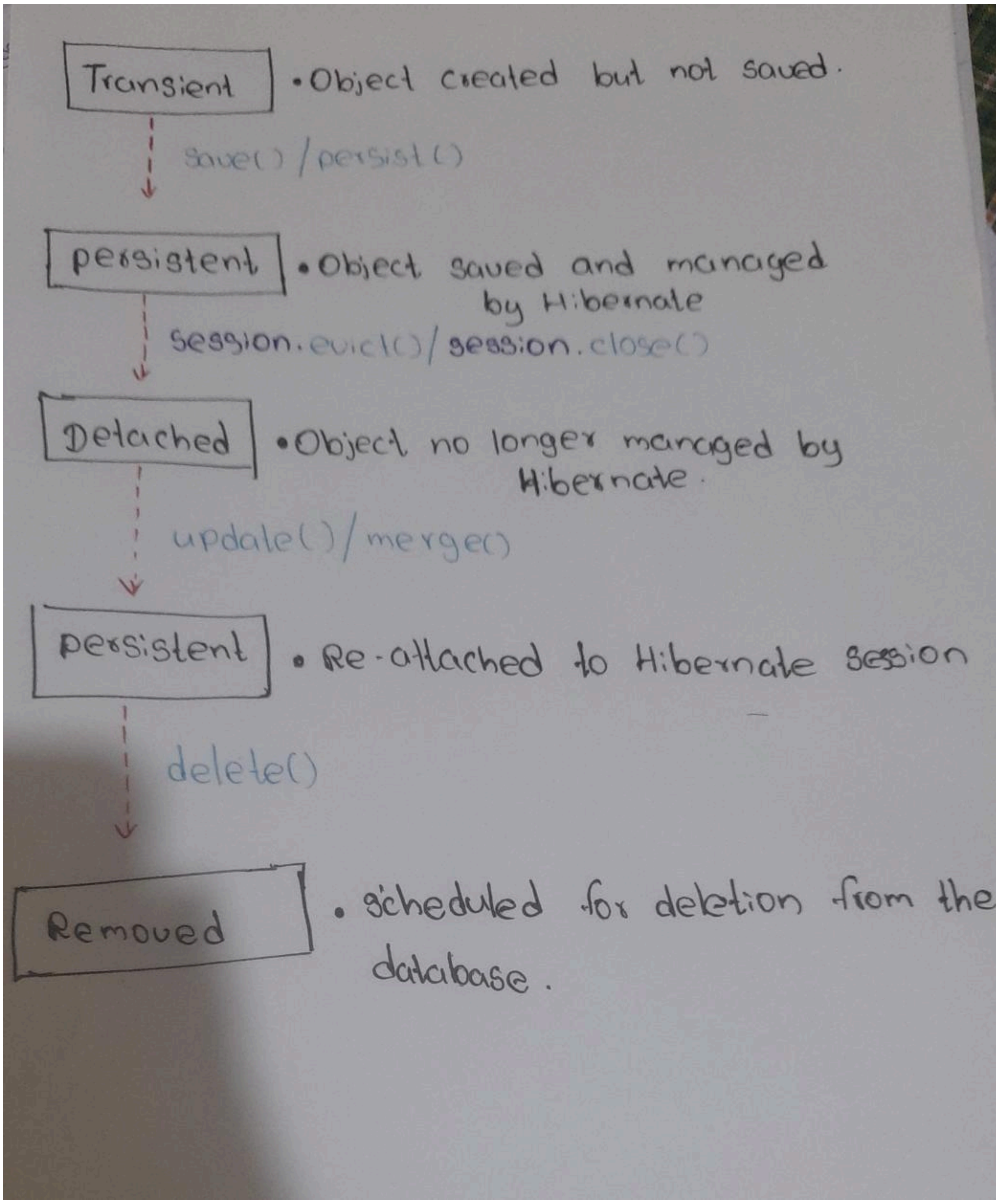
3. Hibernate provides the actual tools and framework to perform ORM but JPA provides guidelines and rules for ORM.
   Hibernate can be used directly as a standalone ORM framework but JPA cannot be use directly .
   JPA is like a blueprint for ORM, while Hibernate is one of the tools that implements this blueprint.

4. RDBMS stands for Relational Database Management System. It is software that manages data stored in a relational database, where data is organized into tables with rows and columns.

5. RDBMS used to store, retrieve, and manage structured data directly using SQL but Hibernate simplifies database interaction in Java applications by automatically converting Java objects to database tables and vice versa.
   RDBMS relies on SQL for operations like creating tables, inserting, updating, and retrieving data but Hibernate uses HQL or Java methods for database operations, abstracting SQL complexity.

6. Advantage - Hibernate maps Java objects to database tables, simplifying database operations by reducing boilerplate code and it can automatically create and update tables based on entity classes, reducing database setup effort.
   Disadvantage - Hibernate can be more complex to set up compared to simpler database access frameworks like JDBC and debugging Hibernate-generated SQL queries can be difficult due to the abstraction layer.

7.


Transient
• Object created but not saved.
save() / persist()
persistent
• Object saved and managed by Hibernate
session.evict() / session.close()
Detached
• Object no longer managed by Hibernate.
update() / merge()
persistent
• Re-attached to Hibernate session
delete()
Removed
• scheduled for deletion from the database.

8. ORM stands for Object-Relational Mapping, which is a programming technique that allows developers to interact with a database using object-oriented programming languages.

9.
Step 01- Add Hibernate dependency - include hibernate libraries in our project.

Step 02- Create Hibernate configuration - set up a configuration file to connect our application to the database.

Step 03 - Create entity classes - entities are java classes that map to database table. Annotate them with hibernate annotations like @Entity , @Table , @Column.

Step 04 - Write hibernate methods -

i)open session - Use the Hibernate Session object to interact with the database. A SessionFactory provides sessions.
ii)perform operations - Save an Object To save an entity in the database, Retrieve an Object To fetch data by primary key, Update an Object To modify and update data, Delete an Object To remove an entity from the database.
ii)use HQL - To execute queries similar to SQL but using entity names.

Methods -
`save(Object)`: Persists an object in the database and returns the generated ID.
`get(Class<T>, Serializable id)`: Fetches an entity by its ID (eager fetching).
`load(Class<T>, Serializable id)`: Similar to `get` but uses lazy fetching (data isn't loaded until accessed).
`update(Object)`: Updates an existing object in the database.
`delete(Object)`: Deletes an object from the database.
`createQuery(String hql)`: Creates an HQL query for advanced operations.

10.
Lazy fetch - Data is loaded on demand. Related entities are not fetched from the database until they are explicitly accessed.
Eager fetch - Data is loaded immediately along with the parent entity.

11.
Lazy fetch - session.get(), session.load(), Hibernate.initialize(), @Transactional.
Eager fetch - FetchType.EAGER, fetch()

12.
i) XML-based configuration
ii) Programmatic (Java-based) configuration

13.
i)One-to-One Relationship

- **Owning Side**: The side that holds the foreign key and manages the association. This side is responsible for persisting and deleting the relationship.
- **Inverse Side**: The side that doesn't hold the foreign key but maps to the owning side. It is used for navigation and is not responsible for managing the association.

ii)One-to-Many Relationship

- **Owning Side**: The side that contains the foreign key. It is responsible for managing the relationship (e.g., persisting or deleting).
- **Inverse Side**: The side that does not contain the foreign key but refers to the owning side. It's used for navigation and querying but doesn't manage the relationship.

iii)Many-to-Many Relationship

- **Owning Side**: This side manages the relationship and has the responsibility for the association table that connects the two entities.
- **Inverse Side**: The side that doesn't manage the relationship but can access the owning side to navigate through the connection.

14.On the inverse side

15.

1. `CascadeType.PERSIST`: save operation to the associated entities.
2. `CascadeType.MERGE`: update operation to the associated entities.
3. `CascadeType.REMOVE`: delete operation to the associated entities.
4. `CascadeType.REFRESH`: refresh operation to the associated entities.
5. `CascadeType.DETACH`: detach operation to the associated entities.
6. `CascadeType.ALL`: Includes all the above operations.

And because cascade types are used to automate the propagation of operations from one entity to its related entities, ensuring data consistency and simplifying code by reducing the need for manual management of associated entities.

16.HQL, Native SQL, JPQL

17.HQL is abstract and works with entities, while Native SQL is database-specific and works directly with tables and columns.

18.

i) First-level cache - it is the default cache in Hibernate.It is used to avoid multiple database hits for the same entity within a session.

ii)Second-level cache - that is shared across multiple sessions.It helps reduce database access by caching data that can be used by any session, improving performance.

19.**Password encryption** is important to protect user data from unauthorized access and breaches. It ensures that even if attackers gain access to the database, they cannot easily retrieve users' original passwords