8)Hibernate, JPA and Java Spring Data Frameworks

1) Hibernate

Hibernate is an object relational mapper, which lets you map the classes to database tables, for out kredibizden examples, user, product, bank classes have its own tables and the properties of the classes correspond to database table columns.

```
@Entity
@Table(name=users)
public class User {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Long id;
        @Column(name = "name")
        private String name;
        @Column(name = "email")
        private String email;
        @Column(name = "phone")
        private String phone;

    }
```

Above class has a corresponding users table in database with id, name, email, and phone columns.

2) JPA

JPA stands for Java Persistence API, which serves as a Java feature facilitating the connection between databases and Java objects. It offers a standard way to retrieve and store data object.

Thanks to JPA, data can be accessed using Java objects without the need to write SQL queries. Additionally, thanks to the utilization of JPA, transitioning between databases can be achieved with minimal effort.

JPA consists of three fundamental components: entities, entity manager, and persistence provider. Entities represent Java objects that mirror tables in the database. The entity manager serves as a central hub for entity management, encompassing tasks such as data creation, addition, deletion, and updating. The persistence provider acts as the framework responsible for translating JPA operations into corresponding database commands. Hibernate, a widely-used implementation of JPA, serves as a tool adhering to JPA specifications.

The following code provides an example of adding a customer object to its corresponding table:

```
EntityManager em = PersistenceManager.createEntityManagerFactory("myPersistenceUnit").createEntityManager();

em.getTransaction().begin();

User user = new User();
user.setName("Burcu Kundum");
user.setEmail("burcu@test.com");

em.persist(user);

em.getTransaction().commit();

em.close();
```

3) Java Spring Data

The Spring framework facilitates data operations, offering infrastructure independent of the data storage structure. This provides a development environment that simplifies processes.
There are two fundamental concepts: repositories and annotations. Repositories provide interfaces for CRUD (Create, Read, Update, Delete) operations. Spring automatically establishes this structure for entity classes. Spring Data relies on additional annotations to define data access logic. Common additional annotations include @Entity for entities, @Repository for repository interfaces, and @Id for primary key fields.

```
@Entity
public class Product {
  @Id
  private Long id;
  private String name;
  private double price;
}

@Repository
public interface ProductRepository extends CrudRepository<Product, Long> {
  // Custom methods
}
```