**Hands-on 3**

**How object to relational database mapping is done using Hibernate annotations**

In annotation based configuration, we use Java annotations instead of XML files to map a Java class to a database table. An annotation is a special symbol that starts with @, and it is written above the class, field, or method to tell Hibernate how to treat it. For example, we use @Entity to mark a class as a database table and @Table(name="table\_name") to link it with a specific table name. We use @Id for the primary key and @Column(name="column\_name") to map a field to a specific column in the table. With annotations, all the configuration is done directly inside the Java class, which makes the code shorter and easier to understand.

**Hibernate configuration file details**

Even when using annotations, we still need a configuration file to tell Hibernate how to connect to the database. This is usually called hibernate.cfg.xml. In this file, we define the driver class, database URL, username, password, and dialect. We also mention the name of the annotated Java classes that should be used by Hibernate. This file is required for Hibernate to know how to start and where to look for mappings.

**Hibernate database properties used in the configuration**

In the hibernate.cfg.xml file, we add properties that define the database connection and behavior. The driver class tells Hibernate which JDBC driver to use. The URL specifies the database location. The username and password are used for logging in to the database. The dialect tells Hibernate which type of SQL to use based on the database, such as MySQL or Oracle. These settings help Hibernate communicate with the correct database using the right SQL commands.

**Explanation of commonly used annotations**

***@Entity***

@Entity is used to tell Hibernate that this Java class should be mapped to a database table. Without this annotation, Hibernate will not recognize the class as something to be stored in the database.

***@Table(name="table\_name")***

@Table is used along with @Entity to give the actual table name in the database. If we do not use it, Hibernate assumes the table name is the same as the class name.

***@Id***

@Id is used to mark a field as the primary key of the table. This is needed for Hibernate to uniquely identify each row.

***@GeneratedValue***

@GeneratedValue tells Hibernate to automatically generate the value of the primary key. It is commonly used with databases that use auto-increment for primary keys.

***@Column(name="column\_name")***

@Column is used to map a Java field to a specific column in the database. It allows us to use a different column name than the field name in the class.