Day 10

HashMap<K,V>

- It is map collection
- It's implementation is based on Hashtable.
- It can not contain duplicate keys but it can contain duplicate values.
- In HashMap, key and value can be null.
- It is unsynchronized collection. Using Collections.synchronizedMap() method, we can make it synchronized.

```
Map m = Collections.synchronizedMap(new HashMap(...));
```

- It is introduced in jdk 1.2.
- Instantiation

```
Map<Integer, String> map = new HashMap<>( );
```

- Note : In HashMap, if we want to use element of non final type as a key then it should override equals() and hashCode() method.

LinkedHashMap<K,V>

- It is sub class of HashMap<K,V> class
- Its implementation is based on LinkedList and Hashtable.
- It is Map collection hence it can not contain duplicate keys but it can contain duplicate values.
- In LinkedHashMap, key and value can be null.
- It is unsynchronized collection. Using Collections.synchronizedMap() method we can make it synchronized.

```
Map m = Collections.synchronizedMap(new LinkedHashMap(...));
```

- LinkedHashMap maintains order of entries according to the key.
- Instantiation:

```
Map<Integer, String> map = new LinkedHashMap<>();
```

- It is introduced in jdk 1.4

TreeMap

- It is map collection.
- It can not contain duplicate keys but it can contain duplicate values.
- It TreeMap, key not be null but value can be null.
- Implementation of TreeMap is based on Red-Black Tree.
- It maintains entries in sorted form according to the key.
- It is unsynchronized collection. Using Collections.synchronizedSortedMap() method, we can make it synchronized.

```
SortedMap m = Collections.synchronizedSortedMap(new TreeMap(...));
```

- Instantiation:

```
Map<Integer, String> map = new TreeMap<>();
```

- It is introduced in jdk 1.2
- Note : In TreeMap, if we want to use element of non final type as a key then it should implement Comparable interface.

JDBC

- Java DataBase Connectivity
- Specification = { Abstract classes + Interfaces }
- JDBC is a specification defined by SUN/ORACLE.
- Implementating JDBC specification is a job of database vendor.
- Database connector(.jar) contains implementation of JDBC specification.
- e.g
    - mysql-connector-java-8.0.15.jar is connector for mysql database version 8.0.15.
- Using JDBC, we can access data from any RDBMS into java application.
- Main purpose of JDBC is to minimize database vendor dependancy.
- If we want use JDBC then we should import java.sql package.

JDBC Versions

- Since 1.8 : JDBC 4.2 API
- Since 1.7 : JDBC 4.1 API
- Since 1.6 : JDBC 4.0 API
- Since 1.4 : JDBC 3.0 API
- Since 1.2 : JDBC 2.0 API

JDBC Interfaces

1. Driver
2. Connection
3. Statement
4. PreparedStatement

5. CallableStatement
6. ResultSet
7. Blob
8. Clob
9. NClob
10. DatabaseMetaData
11. ResultSetMetaData
12. ParameterMetaData

JDBC Classes

1. DriverManager
2. Date
3. Time
4. Timestamp
5. Types

JDBC Exception

1. SQLException
2. SQLIntegrityConstraintViolationException

JDBC Driver:

- JDBC driver is a program, which converts Java request into SQL request and SQL response into Java response.
- All JDBC drivers must implement java.sql.Driver interface.
- Types of JDBC Driver:
    1. Type I
    2. Type II
    3. Type III
    4. Type IV

Type I Driver

- It is also called JDBC-ODBC bridge driver
- sun.jdbc.odbc.JdbcOdbcDriver
- Open DataBase Connectivity(ODBC)
- ODBC is specification defined by Microsoft to access data from databases.
- Type-I driver implicitly use ODBC driver.
- It comes with JDK.
- Vendor : Sun Microsystems
- It is database independant but platform dependant driver.

Advantages:

1. Since it comes with JDK, seperate installation is not required.

2. Easy to use.(Simple we need to configure System/User DSN).
3. It is database independant driver.

Disadvantages:

1. Since multiple conversions are involed, it is slower int performance.
2. Since it is based on ODBC, it is platform dependant(Mircrosoft specific).
3. Since jdk1.8 it is obsolete.

Type II Driver

- It is also called Native API Driver.
- e.g : Oracle OCI(Oracle Call Interface) Driver.
- In context of java, C/C++ code is called native code.
- If we want to use native code into Java code then we should use JNI.
- Type -II driver is based on JNI.
- Type-II driver do not use ODBC driver rather it uses vendor provided database specific native library.

Advantages:

1. It is faster than Type-I driver
2. It is more portable than Type-I driver

Disadvantages:

1. It is database dependant as well as platform dependant driver
2. Not all the vendors provide native library hence its use is limited.

Type III Driver

- It is also called as Network protocol driver or middleware driver.
- It follows n-tier architecture.
- e.g RMI Web Logic Driber
- It is database independant as well as platform independant driver.
- It is pure java driver which requires network setup.

Advantages:

1. Since it is written in java, it is truly portable.
2. Since it is database independant driver, we can use it to communicate with multiple databases
3. No need to use ODBC driver and vendor provided database specific native API lib.

Disadvantages:

1. It requires seperate network setup. Hence expensive to use.

Type IV Driver

- It is also called as Database protocol driver / thin driver / pure java driver.
- Since it is written in java it is truly portable.
- It is database dependant driver.
- We can use it with CUI/GUI as well as web application
- Either database vendor / third party can provide it.
- It implicitly use socket programming.
- e.g : com.mysql.jdbc.Driver

Advantages:

1. It is portable driver
2. Since implicitly use socket programming, no need to use odbc driver or database specific native lib
3. Easily available
4. Installation is easy.
5. We can use it with CUI as well as web application

Disadvantages:

1. It is database dependant driver

Steps to connect java application to the database.

- Step 0 : Add database connector(.jar) in buildpath
- Step 1 : Load and Register driver
- Step 2 : Establish Connection Using Users Credential
- Step 3 : Create Statement/PreparedStatement/CallableStatement object to execute query.
- Step 4 : Prepare and execute Query.
- Step 5 : Close resources

Configuration Information

- Database Server : MySQL 8.0.17
- Database/Schema : dac_db
- User Name : root
- Password : mysql@8017
- Driver : com.mysql.cj.jdbc.Driver
- URL : jdbc:mysql://localhost:3306/dac_db;
- To execute DML statements/queries (Insert/Update/Delete) we should use executeUpdate() method and to execute DQL statement(select) we should use executeQuery() method.
- executeUpdate() and executeQuery() are methods of java.sql.Statement interface.

```
Statement stmt = con.createStatement();
```

ResultSet

- It is interface declared in java.sql package.
- "ResultSet executeQuery(String sql) throws SQLException " is a method of Statement interface which returns ResultSet.
- ResultSet implementation object contains records returned by select query.
- A ResultSet object maintains a cursor pointing to its current row of data. Initially the cursor is positioned before the first row. The next() method moves the cursor to the next row, and because it returns false when there are no more rows in the ResultSet object, it can be used in a while loop to iterate through the result set.
- If we want to access data from ResultObject then we should use getXXX() method.
- To access data either we should use column index or column lable(name).

Object Relational Mapping(ORM)

- In SQL, Table is also called as Relation
- SQL terminologies
    1. Database
    2. Table
    3. Column
    4. Row
    5. Primary key / Foreign key
- OO Terminologies
    1. Class
    2. Field
    3. Instance / Object
- ORM Database Java Table Class Column Field Record Instance Primary key Identity Field
- If we want to map database enities with oo enity then we should define POJO class.

Plain Old Java Object(POJO)

- It is a class which do not implement interface or do not extend any class.
- It is also called as Data Transfer Object(DTO) / Business Object(BO) / Value Object(VO) / Entity.
- Rules to define POJO class

1. It must be packaged public class
2. It should contain default constructor
3. For every column it should contain private field inside class.
4. For every private field, it should contain getter and setter methods.
5. It is should not contain business logic but it can contain toString(), equals() or hashCode() method.

Data Access Object (DAO) Layer

- If we want to seperate data manipulation logic from business logic then we should define DAO class.
- Per table we should define one POJO and DAO.
- Rules to define DAO class.

1. It must be packaged public class.
2. It should contain default constructor
3. It should contain data manipulation logic(INSERT,UPDATE,DELETE, SELECT statement)