Day 4 – Collection API & JDBC API

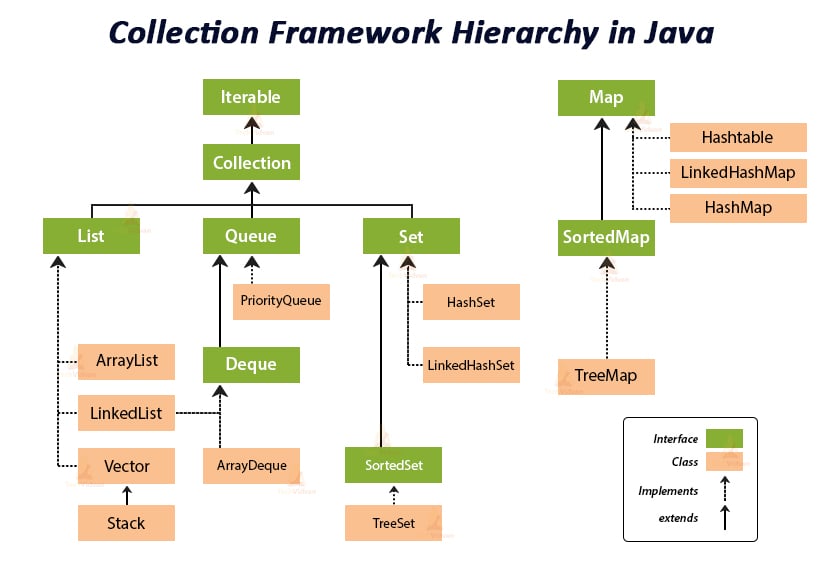
Day3 Revisit – OOPs concepts.

Class, Objects, Abstraction, Encapsulation, Inheritance & Polymorphism.

Collection API – It always works with Group of objects.

API – Application Programming Interface. (Reusing the existing code)

Java.util – Interface (



Stack – LIFO (Last In First Out) - Push (add) pop (removing)

Queue – FIFO (First In First Out) – Enque (Adding) Deque (removing)

List allows duplicates, maintains insertion order. (null values)

Set won’t allow duplicates. (ignore the duplicate value) – Won’t maintain the insertion order.

Default package in Java Lang is – java.lang (String, StringBuffer, StringBuilder, Integer, Boolean …)

Array – Fixed in Size,

ArrayList – Dynamically growing in size ( Stores data in Continuous memory location)

LinkedList -

Collection API always works with Object only (It won’t accept primitive)

Boxing – Converting primitive to its corresponding object notation.

AutoBoxing – When you pass primitive value to collection api, autoboxing will happen.

Auto-Unboxing – Converting object to it’s corresponding primitive value.

Collections, Arrays – utility classes.

JDBC – Java DataBase Connectivity. (API) [Interfaces & Classes]

Java.sql – package.

Java is a prog. Lang. RDBMS – DB Server (Client & Server) [Oracle, MySQL, DB2, Postgres, MS SQL…, SQLite]

1. Driver
2. Connection
3. Statement/PreparedStatement/CallableStatement
4. ResultSet/RowSet
5. DatabaseMetaData/ResultSetMetaData

Date, DriverManager(AbstractClass)

1. Install drivers
2. Connect the printer in Laptop/Desktop (physical connection – wired/wireless)
3. Test print
4. Actual Print
5. Turn the printer off.

Steps in JDBC

1. Loading & Registering the Driver
2. Establishing the Connection to RDBMS
3. Creating and Executing the Query
4. Processing the Result
5. Close all resources.

JDBC is a Specification – It contains Interfaces & abstract classes . (In-complete)

DB mfrs – will provide implementation (Drivers)

CREATE TABLE `sampledb`.`employee` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

`email` VARCHAR(45) NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) VISIBLE);

jdbc:sqlserver://myServer\SQLEXPRESS;databaseName=myDatabase;user=myUser;password=myPassword

<https://learn.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver17>

import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
  
public class SqlServerJdbcExample {  
  
 public static void main(String[] args) {  
 String connectionUrl = "jdbc:sqlserver://localhost:1433;databaseName=YourDatabaseName;encrypt=false;";  
 String username = "YourUsername";  
 String password = "YourPassword";  
  
 try (Connection connection = DriverManager.getConnection(connectionUrl, username, password);  
 Statement statement = connection.createStatement()) {  
  
 *// Example: Create a table*  
 String createTableSql = "CREATE TABLE Employees (Id INT PRIMARY KEY, Name VARCHAR(50))";  
 statement.executeUpdate(createTableSql);  
 System.out.println("Table 'Employees' created successfully.");  
  
 *// Example: Insert data*  
 String insertSql = "INSERT INTO Employees (Id, Name) VALUES (1, 'John Doe')";  
 statement.executeUpdate(insertSql);  
 System.out.println("Data inserted successfully.");  
  
 *// Example: Select data*  
 String selectSql = "SELECT \* FROM Employees";  
 ResultSet resultSet = statement.executeQuery(selectSql);  
 while (resultSet.next()) {  
 System.out.println("Employee ID: " + resultSet.getInt("Id") + ", Name: " + resultSet.getString("Name"));  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }