**Assignment No:9**

**Title:** Write a program to implement MongoDB database connectivity with any front-end language to implement database navigation operations (add, delete, edit, etc.).

**Problem Statement:** Implement the Java program that implements MongoDB database connectivity and performs basic CRUD (Create, Read, Update, Delete) operations such as add, delete, edit, and view using the MongoDB Java Driver.

**Objective:** To gain knowledge of NoSQL databases for processing unstructured data.

**Outcome:** Construct a two-tier and three-tier application using a suitable programming language and database.

**Tools Required:** Ubuntu OS, MongoDB, Eclipse, Mongo Java Driver.

**Theory:**

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

**1. Theory Steps for MongoDB Database Connectivity Program**

* 1. **MongoDB Setup**:
  + **Install MongoDB**: Ensure that MongoDB is installed and running on your system. By default, it runs on localhost and listens on port 27017.
  + **Java Setup**: Make sure you have **Java** and an IDE like **Eclipse** or **IntelliJ IDEA** set up properly.

**1.2 Dependencies**:

* + The program uses the **MongoDB Java Driver** to interact with the MongoDB database. You need to include the required **MongoDB Java Driver** dependency either via Maven or by manually downloading and adding the JAR files.

**1.3 Program Initialization**:

* + The program connects to the **MongoDB database** and initializes the MongoDatabase and MongoCollection objects which represent the database and collection within MongoDB.
  + The MongoClient is responsible for establishing the connection between the Java application and the MongoDB server.
  1. **CRUD Operations**:
  + **Create**: We use the insertOne() method to add a new document to the MongoDB collection.
  + **Read**: The find() method retrieves all documents from the collection. These are iterated and printed.
  + **Update**: The updateOne() method updates the existing document based on the specified filter (i.e., document \_id).
  + **Delete**: The deleteOne() method deletes a document based on a filter.

**1.5 User Interaction**:

* + The program runs in a loop, presenting the user with a menu of CRUD operations to choose from (Add, View, Update, Delete, Exit).
  + The program executes the corresponding CRUD operation based on the user's input.

**1.6 Closing the Connection**:

* + After the operations are completed, the MongoDB connection is closed using mongoClient.close() to free up resources.

**2. Package**

* 1. **com.mongodb.client.MongoClient**: This is the entry point for all interactions with MongoDB. The MongoClient object is used to establish a connection to the MongoDB server. It is used to access databases and collections.

MongoClient mongoClient = MongoClients.create(URI);

* 1. **com.mongodb.client.MongoDatabase**: Represents the database within MongoDB. Once a connection is established with MongoClient, you use this class to interact with a specific database.

MongoDatabase database = mongoClient.getDatabase(DATABASE\_NAME);

* 1. **com.mongodb.client.MongoCollection**: Represents a collection within a database in MongoDB. MongoDB stores data in collections, and this class is used to perform operations (insert, read, update, delete) on those collections.

MongoCollection<Document> collection = database.getCollection(COLLECTION\_NAME);

* 1. **org.bson.Document**: This class represents a BSON document (Binary JSON). It is used to represent the data structure in MongoDB (essentially the document or record). Each document can contain fields that store various types of data.

java

Document document = new Document("name", name) .append("email", email);

**com.mongodb.client.model.Filters**:

* 1. **com.mongodb.client.model:** This class is used to create filter criteria for querying documents. For example, it can be used to find a document based on certain conditions (like finding a document by its \_id).

Filters.eq("\_id", id); // Filters for documents matching the given ID.

* 1. **com.mongodb.client.model.Updates**: This class provides various methods to update documents. It is used to define updates that should be performed on documents. For example, you can update a field value within a document.

Updates.set("name", name).set("email", email);

**Conclusion:**

We have successfully implemented Java program for MongoDB database connectivity and performed basic CRUD (Create, Read, Update, Delete) operations.