DBMS Assignment C1

Program:

```
package com.company;

import com.mongodb.DB;

import com.mongodb.client.FindIterable;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import com.mongodb.MongoClient;

import com.mongodb.ServerAddress;

import org.bson.Document;



import java.rmi.UnknownHostException;

import java.util.Iterator;

import java.util.List;

import java.util.Set;

public class Main {

   public static void main(String[] args) {

      MongoClient mongoClient = new MongoClient("localhost");

      List<String> databases = mongoClient.getDatabaseNames();

      MongoDatabase database = mongoClient.getDatabase("Student");

      //mescoe: database name

      // Retrieving a collection
```

```
    MongoCollection<Document> collection = database.getCollection("studentinfo");
    //te_comp: collection
    System.out.println("Collection student info selected successfully");

    // Getting the iterable object
    FindIterable<Document> iterDoc = collection.find();
    int i = 1;
    // Getting the iterator
    Iterator it = iterDoc.iterator();
    while (it.hasNext()) {
       System.out.println(it.next());
       i++;
    }
    mongoClient.close();

  }

}**********************************************************************************
*
```

Output:

```
Collection student info selected successfully
Document{{_id=5dce4e9ff68a9c2449e197b2, name=Dhruvil, roll=1, age=20}}
Document{{_id=5dce4e9ff68a9c2449e197b3, name=Soham, roll=2, age=20}}

**********************************************************************************
```

MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student: Dhruvil Shah	**Class: TE Comp 1**
Semester/Year: 5th/2020	**Roll No: 047**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Group C-01**

PART: C ASSIGNMENT NO: 01

AIM: To write a program to implement MongoDB database connectivity with PHP/ python/Java to implement Database navigation operations (add, delete, edit etc.) using ODBC/JDBC.

OBJECTIVES:

- To study Java and MongoDB connectivity using any Java application.
- To perform CRUD operations.

PRE-REQUISITES:

- Basics of NoSQL database-Mongodb
- Java and MongoDB server connectivity.
- MongoDB 2.2.3 or later.
- MongoDB-Java-Driver 2.10.1
- JDK 1.6
- Eclipse 4.2

THEORY:

MongoDB is the leading NoSQL database system which has become very popular for recent years due to its dynamic schema nature and advantages over big data like high performance, horizontal scalability, replication, etc. Unlike traditional relational database systems which provide JDBC-compliant drivers, MongoDB comes with its own non-JDBC driver called Mongo Java Driver. That means we cannot use JDBC API to interact with MongoDB from Java. Instead, we have to use its own Mongo Java Driver API. The official MongoDB Java Driver providing both synchronous and asynchronous interaction with MongoDB.

Downloading Mongo Java Driver

Download the JAR file name is **mongo-java-driver-VERSION.jar** . Copy the downloaded JAR file into your classpath. Online API documentation for Mongo Java Driver can be found at http://api.mongodb.com/java/current/index.html.

Using the Java driver is very simple. First, be sure to include the driver jar mongo.jar in your classpath.

Creating MongoDB Java Connection

MongoClient is the interface between our java program and MongoDB server. MongoClient is used to create connection, connect to database, retrieve collection names and create/read/update/delete database, collections, document etc.

One of the MongoDB java driver feature I like most is that it's thread safe, so we can create an instance of MongoClient once and reuse it. Even if multiple thread accesses it simultaneously, a connection is returned from the internal connection pool maintained by it.

For every request to the database (find, insert etc) the Java thread will obtain a connection from the pool, execute the operation, and release the connection. This means the connection (socket) used may be different each time.

- **Downloading Mongo Java Driver**

  Download latest version of Mongo Java Driver (version 2.11.1 as of this writing). The JAR file name is **mongo-java-driver-VERSION.jar** (around 400KB). Copy the downloaded JAR file into your classpath.

- **Connecting to MongoDB using MongoClient**

  The MongoClient class is used to make a connection with a MongoDB server and perform database-related operations. The MongoDB protocol is a simple socket-based, request-response style protocol. Connection with the client and Database server happens through a regular TCP/IP socket. MongoDB uses TCP as its transport layer protocol. The predefined default port for connection is **27017.**

  Here are some examples:

  Creating a MongoClient instance that connects to a default MongoDB server running on localhost and default port:

  ```
  MongoClient mongoClient = new MongoClient();
  ```

  Connecting to a named MongoDB server listening on the default port (27017):

```
MongoClient mongoClient = new MongoClient("localhost");
```

Or:

```
MongoClient mongoClient = new MongoClient("db1.server.com");
```

Connecting to a named MongoDB server listening on a specific port:

```
MongoClient mongoClient = new MongoClient("localhost", 27017);
```

- **Connection to a MongoDB Database**

  To connect database, you need to specify database name, if database doesn't exist then mongodb creates it automatically.

```
DB db = mongoClient.getDB( "database_name" );
System.out.println("Connect to database successfully")
```

- **Create a collection**

  To create a collection, **createCollection()** method of **com.mongodb.DB** class is used.

```
DBCollection coll = db.createCollection("collection_name");
System.out.println("Collection created successfully");
```

- **Getting/ selecting a collection**

  To get/select a collection from the database, getCollection() method of com.mongodb.DBCollection class is used.

```
DBCollection coll = db.getCollection("collection_name");
System.out.println("Collection selected successfully");
```

CONCLUSION:

QUESTIONS:

1. Write Steps for Connection to a MongoDB database.
2. Write installation step for MongoDB on Ubuntu.
3. What are different Packages needed to connect java with MongoDB?
4. What are unstructured database? Enlist advantages of NoSQL database?
5. Explain difference between two-tier and three-tier architectures.

Q1 Write steps for connection to a Mongo DB database

Ans Step 1: Install Mongo DB and start up instance to which you will connect
Step 2: Install your client
Step 3: Obtain your mongo DB connection string
Step 4: Connect to your Mongo DB instance

Q2 Write installation step for Mongo DB on Ubuntu

Ans Step 1: Import the Mongo DB repository
Step 2: Install the Mongo DB packages
Step 3: Launch Mongo DB as a service on Ubuntu 16.04
Step 4: Configure and Connect Mongo DB

Q3 What are the different packages needed to connect java with Mongo DB ?

Ans The different packages needed to connect java with Mongo DB are:

```
import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.ServerAddress;
import com.mongodb.MongoCredential;
import com.mongodb.MongoClientOptions;
```

Q4 What are unstructured database? Enlist advantages of NOSQL database?

Ans Unstructured data simply means that it is datasets that aren't stored in structured database format. Unstructured data has an internal structure, but its not predefined through data models.

Advantages of NoSQL database:

1) Supports hoc queries
2) Indexing
3) Replication
4) Duplication of Data
5) Providing high performance

Q5 Explain difference between two tier and three tier architectures

Ans

	Two Tier Database Architecture		Three Tier Database Architecture
1	It is Client Server Architecture	1	It is Web based Application
2	It is easy to build and maintain	2	It is complex to build and maintain
3	Runs slower	3	Runs faster
4	It has two layers : Client Tier and database	4	It has three layers : Client Tier, Business layer and data layer
5	It results in performance loss whenever the users increase rapidly	5	It results in performance loss whenever the system is run on internet but gives more performance than two-tier architecture.