Group B

Assignment No 13

Title of the Assignment: Database Connectivity

Write a program to implement Mongo DB database connectivity with any front-end language toimplement Database navigation operations (add, delete, edit etc.)

Objective of the Assignment: To understand the concept of Mongo DB database connectivity.

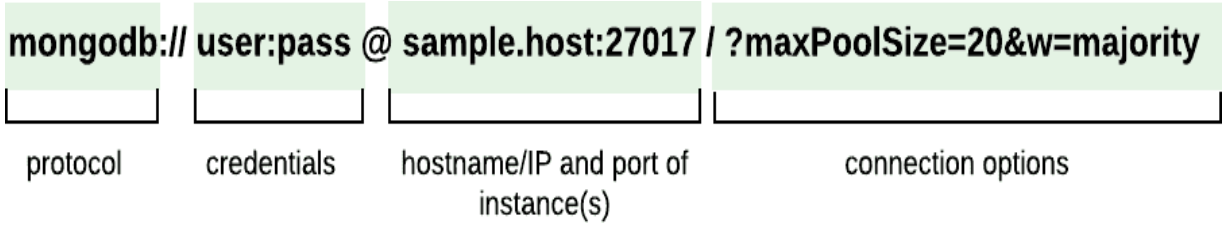
Outcome: Students will be able to learn and understand concept Map reduces operation with examples.

Theory:

Connect to MongoDB

MongoClient:
You can connect to and communicate with MongoDB using the MongoClient class.
Use the MongoClients.create() method to construct a MongoClient. As each MongoClient represents a thread-safe pool of connections to the database, most applications only require a single instance of a MongoClient, even across multiple threads. All resource usage limits, such as max connections, apply to individual MongoClient instances.

Connection URI:
The connection URI provides a set of instructions that the driver uses to connect to a MongoDB deployment. Itinstructs the driver on how it should connect to MongoDB and how it should behave while connected. The following figure explains each part of a sample connection URI:



This figure uses the Standard Connection String Format, mongodb for the protocol. You can also use the DNS Seed List Connection Format, mongodb+srv, if you want more flexibility of deployment and the ability to change the servers in rotation without reconfiguring clients.

The next part of the connection URI contains your credentials if you are using a password-based authentication mechanism. Replace the value of user with your username and pass with your password. If your authentication mechanism does not require credentials, omit this part of the connection URI.

The next part of the connection URI specifies the hostname or IP address, followed by the port of your MongoDB instance. In the example, sample.host represents the hostname and 27017 is the port number. Replace these values to refer to your MongoDB instance.

The last part of the connection URI contains connection options as parameters. In the example, we set two connection options: maxPoolSize=20 and w=majority. For more information on connection options, skip to the Connection Options section of this guide.

The next part of the connection URI contains your credentials if you are using a password-based authentication mechanism. Replace the value of user with your username and pass with your password. If your authentication mechanism does not require credentials, omit this part of the connection URI.

The next part of the connection URI specifies the hostname or IP address, followed by the port of your MongoDB instance. In the example, sample.host represents the hostname and 27017 is the port number. Replace these values to refer to your MongoDB instance.

The last part of the connection URI contains connection options as parameters. In the example, we set two connection options: maxPoolSize=20 and w=majority. For more information on connection options, skip to the Connection Options section of this guide.

Create Database:
test> use persons switched to
db persons Java MongoDB
Driver:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>net.java</groupId>
<artifactId>net.java</artifactId>
<version>0.0.1-SNAPSHOT</version>
<dependencies>
<dependency>
<groupId>org.mongodb</groupId>
<artifactId>mongo-java-driver</artifactId>
<version>3.12.0</version>
</dependency>
</dependencies>
</project>
```

1. Create Collection and insert records

```
package net.java;
import java.util.ArrayList; import
org.bson.Document;
import com.mongodb.MongoCommandException;import
com.mongodb.client.MongoClients; import
com.mongodb.client.MongoCollection; public class
ConnectToDB {
public static void main(String args[])
{try (var mongoClient =
MongoClients.create("mongodb://localhost:27017")) {var database =
mongoClient.getDatabase("persons");
System.out.println("database name -> " + database.getName());for
(String name: database.listCollectionNames()) {
System.out.println(name);
}
try {
database.createCollection("users");
System.out.println("Collection created successfully");
} catch (MongoCommandException e)
{
database.getCollection("users").drop();
}
var docs = new ArrayList < Document >
(); MongoCollection < Document >
collection =
database.getCollection("users");
        var d1 = new Document("_id", 1);
        d1.append("_firstName",
        "Prashant");
        d1.append("_lastName", "Kadam");
        docs.add(d1);
        var d2 = new Document("_id", 2);
        d2.append("_firstName",
        "Vishal");
        d2.append("_lastName",
        "Pawar"); docs.add(d2);
        var d3 = new Document("_id", 3);
        d3.append("_firstName",
        "Vibhas");
        d3.append("_lastName",
        "Kadam");docs.add(d3);
        collection.insertMany(docs);
        System.out.println("Inserted records successfully");
        }
        }
        }
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console ×  Coverage
<terminated> ConnectToDB [Java Application] /home/dipali/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.4.v20221004-1257/jre/bin/java (03-Nov-2022, 8:12:24
Nov 03, 2022 8:12:25 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500
database name -> persons
Nov 03, 2022 8:12:25 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Nov 03, 2022 8:12:25 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:29}] to localhost:27017
Nov 03, 2022 8:12:25 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, v
Nov 03, 2022 8:12:25 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:30}] to localhost:27017

users
Inserted records successfully
Nov 03, 2022 8:12:25 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Closed connection [connectionId{localValue:2, serverValue:30}] to localhost:27017 because the pool has been closed.
```

2. Read Documents

```
package net.java;
import com.mongodb.client.MongoClients;
import
com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import org.bson.Document;
import java.util.ArrayList;
public class MongoReadAll
{
public static void main(String[] args)
{try (var mongoClient =
MongoClients.create("mongodb://localhost:27017")
) {var database =
mongoClient.getDatabase("persons");
                MongoCollection < Document > collection
                =database.getCollection("users");
                try (MongoCursor < Document > cur
                =collection.find().iterator()) {
                while (cur.hasNext())
                {var doc = cur.next();
                var users = new ArrayList < >
                (doc.values());System.out.printf("%s:
                %s%n", users.get(1), users.get(2));
                }
                }
                }
                }
                }
```

OUTPUT:

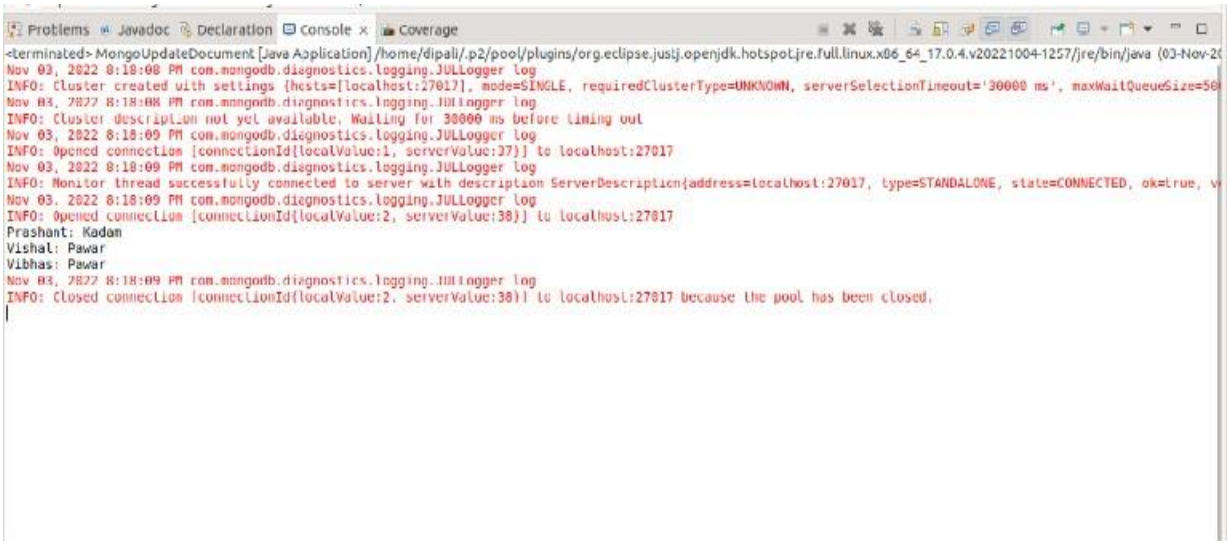
Output:

```
Problems  Javadoc  Declaration  Console ×  Coverage
<terminated> MongoReadAll [Java Application] /home/dipali/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.4.v20221004-1257/jre/bin/java (03-Nov-2022, 8:16:0
Nov 03, 2022 8:16:07 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=50
Nov 03, 2022 8:16:07 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Nov 03, 2022 8:16:08 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:31}] to localhost:27017
Nov 03, 2022 8:16:08 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, v
Nov 03, 2022 8:16:08 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:32}] to localhost:27017
Prashant: Kadam
Vishal: Pawar
Vibhas: Kadam
Nov 03, 2022 8:16:08 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Closed connection [connectionId{localValue:2, serverValue:32}] to localhost:27017 because the pool has been closed.
```

3. Update package net.java;

```
import com.mongodb.client.MongoClients;
import
com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import org.bson.Document;
import
java.util.ArrayList
; public class
MongoReadAll {
public static void
main(String[] args)
 {try
(var mongoClient =
MongoClients.create("mongodb://localhost:27017")) {var
database = mongoClient.getDatabase("persons");
MongoCollection < Document > collection =
database.getCollection("users");
try (MongoCursor <
Document > cur =
collection.find().iterator())
{
while
(cur.hasNex
t()) {var doc
= cur.next();
var users = new ArrayList < >
(doc.values());
System.out.printf("%s: %s%n",
users.get(1), users.get(2));
}
}
}
}
}
```

OUTPUT:

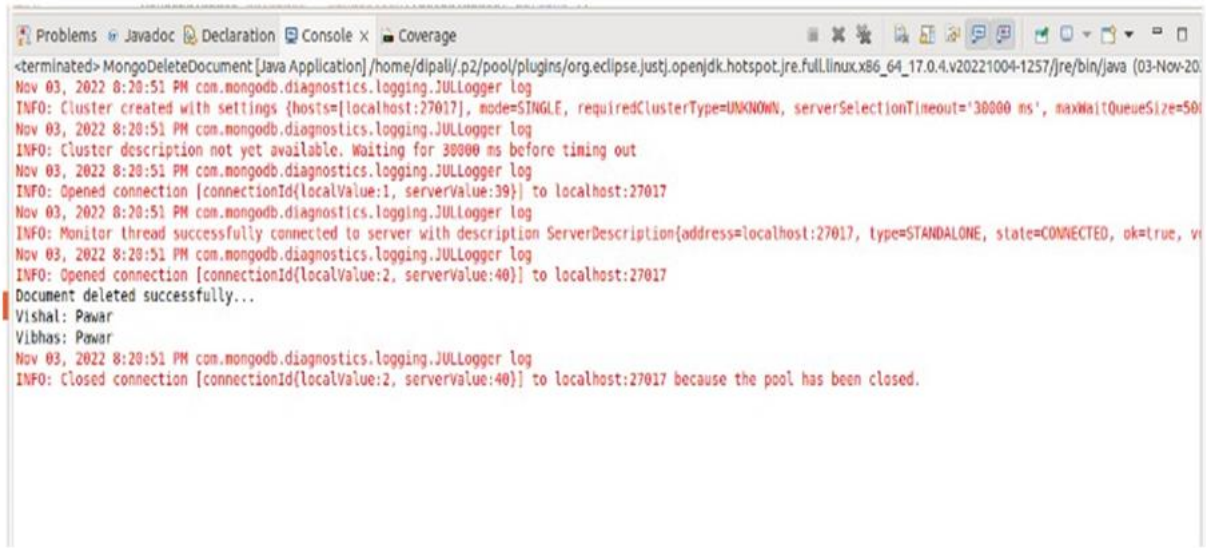


4. Delete package net.java;

```
import java.util.ArrayList; import
org.bson.Document;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;

public class
MongoDeleteDocument {
public static void main(String[]
args) {
// Creating a Mongo clienttry (var mongoClient =
MongoClients.create("mongodb://localhost:27017")) {
// Accessing the database
MongoDatabase database = mongoClient.getDatabase("persons");
// Retieving a collection
MongoCollection < Document >
collection =
database.getCollection("users");
// Deleting the documents
collection.deleteOne(Filters.eq("_id", 1));
System.out.println("Document deleted
successfully...");try (MongoCursor < Document
> cur = collection.find().iterator()) {
while
(cur.hasNext())
{var doc =
cur.next();
var users = new ArrayList < >
(doc.values());System.out.printf("%s:
```

```
%s%n", users.get(1), users.get(2));
}
}
}
}
}
```



Conclusion: Performed implementation of Mongo DB database connectivity.

Viva Question:

- What is MongoDB?
- What are applications of MongoDB?
- Write advantages of MongoDB?
- Write disadvantages of MongoDB?Write the features for MongoDB?

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator:	