Name: Aryan Sachan
Reg No: 22BCE5023

Saturday Assignment

1. **Create database name Saturday, then create collection name student, then insert four documents containing firstname, lastname ,marks and age then find the greatest marks out of four document.**

Sol:
```
import java.util.ArrayList;
import java.util.List;

import org.bson.Document;

import com.mongodb.BasicDBObject;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

public class testTask {
   public static void main(String[] args) {
      // Creating a Mongo client
      MongoClient mongoClient =
MongoClients.create("mongodb://localhost:27017");
      MongoDatabase database =
mongoClient.getDatabase("saturday");

      // Get the collection
      MongoCollection<Document> collection =
database.getCollection("student");
      List<Document> documents = new ArrayList<>();

      documents.add(new Document("firstname", "John")
         .append("lastname", "Doe")
         .append("marks", 85)
         .append("age", 20));

      documents.add(new Document("firstname", "Jane")
         .append("lastname", "Smith")
         .append("marks", 92)
         .append("age", 22));
```

```java
        documents.add(new Document("firstname", "Michael")
           .append("lastname", "Johnson")
           .append("marks", 78)
           .append("age", 19));

        documents.add(new Document("firstname", "Emily")
           .append("lastname", "Williams")
           .append("marks", 92)
           .append("age", 21));

        collection.insertMany(documents);

        // Find the highest marks
        int highestMarks = 0;
        for (Document doc : collection.find()) {
           int marks = doc.getInteger("marks");
           if (marks > highestMarks) {
              highestMarks = marks;
           }
        }

        // Print all documents with the highest marks
        FindIterable<Document> allDocuments =
collection.find(new BasicDBObject("marks", highestMarks));
        for (Document document : allDocuments) {
           System.out.println(document);
        }

        System.out.println("Documents inserted.");
    }
}
```

2. Create database name Saturday, then create collection name employee, then insert four documents containing firstname, lastname ,salary and age then find the lowest salary within the age group of 30-40 out of those four documents.

Sol:
```java
import java.util.ArrayList;
import java.util.List;

import org.bson.Document;

import com.mongodb.BasicDBObject;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;

public class employeeTask {
    public static void main(String[] args) {
        // Creating a Mongo client
        MongoClient mongoClient = 
MongoClients.create("mongodb://localhost:27017");
        MongoDatabase database = 
mongoClient.getDatabase("saturday");

        // Get the collection
        MongoCollection<Document> collection = 
database.getCollection("employee");
        List<Document> documents = new ArrayList<>();

        // Insert four documents
        documents.add(new Document("firstname", "Aryan")
                .append("lastname", "Sachan")
                .append("salary", 50000)
                .append("age", 35));

        documents.add(new Document("firstname", "Deva")
                .append("lastname", "Kashyap")
                .append("salary", 55000)
                .append("age", 32));

        documents.add(new Document("firstname", "Radhika")
                .append("lastname", "Sachdeva")
                .append("salary", 45000)
                .append("age", 38));

        documents.add(new Document("firstname", "Sunitha")
                .append("lastname", "Williams")
                .append("salary", 48000)
                .append("age", 34));

        collection.insertMany(documents);

        // Find the lowest salary within the age group of 30-
40
        BasicDBObject ageQuery = new BasicDBObject("age", new 
BasicDBObject("$gte", 30).append("$lte", 40));
        FindIterable<Document> results = 
collection.find(ageQuery);

        int lowestSalary = Integer.MAX_VALUE;
        for (Document doc : results) {
            int salary = doc.getInteger("salary");
```

```
                if (salary < lowestSalary) {
                    lowestSalary = salary;
                }
            }

            System.out.println("The lowest salary within the age
group of 30-40 is: " + lowestSalary);

            mongoClient.close();
        }
}
```

Tuesday Assignment

1. Find product where price between 400 and 900 or ram as 4 units
Sol1:
db.products1.find({$or:[{$and:[{price:{$gt:400}},{price:{$lt:900}}]},{"spec.ram":4}]})

2. Find product where price is not 699 and ram between 4 and 8 or storage is 16?
Sol2:
db.products1.find({$or:[{$and:[{price:{$not:{$eq:699}}},{$and:[{"spec.ram":{$gt:4}},{"spec.ram":{$lt:8}}]}]}]},{storage:{$eq:16}}]})

3. Find product name where price does not exist and screen is less than 10 ?
Sol3:
db.products1.find({$and:[{price:{$exists:false}},{"spec.screen":{$lt:10}}]})

4) Find products that have either "white" or "black" as a color option and are priced below 800.
Sol4:
db.products.find({$and:[{$or:[{color:{$ne:"white"}},{color:{$ne:"black"}}]},{price:{$lt:800}}]})

5)Find products that do not have "gold" as a color and are priced below 700 or have a storage option of 512GB.
Sol5:

```
db.products1.find({$or:[{$and:[{color:{$not:{$eq:"gold"}}},{price:{$lt:700
}}]},{storage:512}]})
```

6.select phone with screen size not greater than 9.5 and ram not in 4,8
Sol 6:

```
db.products.find({$and:[{"spec.screen":{$lt:9.5}},{"spec.screen":{$nin:[4,
8]}}]})
```

7)select products with cpu power not less than 2.66 nor more than 3.66
Sol7:

```
db.products1.find({$nor:[{"spec.cpu":{$not:{$lt:2.66}}},{"spec.cpu":{$not
:{$gt:3.66}}}]})
```

8) select products with either white colour and storage not less than 128
Sol 8

```
db.products1.find({$and:[{color:"white"},{storage:{$not:{$lt:128}}}]})
```