Name: K.Pavan Kumar

Reg.No:22BCE9548

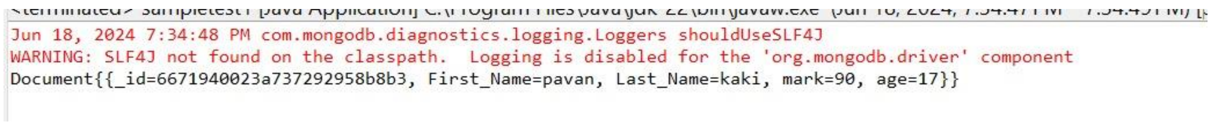
WEEK-2

Q1.

```
package mongo_db;
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.MongoIterable;

public class sampletest2{

	public static void main(String[] args) {
		MongoClient mongoClient = MongoClients.create
("mongodb://localhost:27017");
	    MongoDatabase database = mongoClient.getDatabase("saturday");
	    database.createCollection("employee");
	    MongoCollection<Document> collection = database.getCollection("employee");

	    Document document = new Document("First_Name", "pavan kumar")
	               .append("Last_Name", "kaki")
	             .append("salary", 3200)
```

```
                .append("age", 18)
                .append("_id",100 );
List<Document> documents = new ArrayList<Document>();
documents.add(new Document("First_Name", "deekshitha")
   .append("Last_Name", "bobburi")
   .append("salary", 2600)
   .append("age", 18)
   .append("_id",101 ));
documents.add(new Document("First_Name", "mohith")
   .append("Last_Name", "nandika")
    .append("salary",4500)
   .append("age",20)
   .append("_id",102 ));
documents.add(new Document("First_Name", "dheeraj")
                .append("Last_Name", "pavan")
                 .append("salary",2000)
                .append("age",20)
               .append("_id",103));
documents.add(new Document("First_Name", "pavan")
                .append("Last_Name", "kaki")
               .append("salary",3500)
                .append("age", 19)
                .append("_id",104));
documents.add(new Document("First_Name", "karthik")
                .append("Last_Name", "manne")
                 .append("salary",2800)
                .append("age",20)
               .append("_id",105 ));
documents.add(new Document("First_Name", "bharat")
                    .append("Last_Name", "manne")
                     .append("salary",3600)
                    .append("age",20)
                   .append("_id",106 ));
          collection.insertMany(documents);
          FindIterable<Document> allDocuments = collection.find().sort(new 
BasicDBObject("salary",1)).limit(1);
          for (Document doc : allDocuments) {
               System.out.println(doc);
          }
         }
}
```

Output:



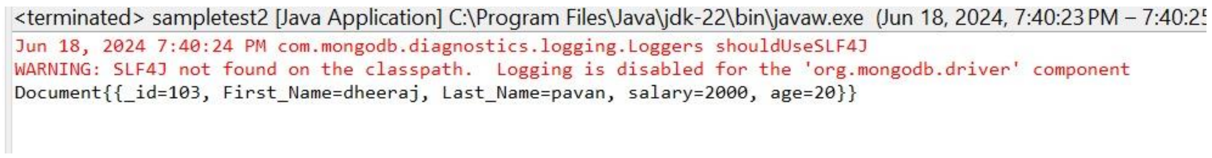
```
Jun 18, 2024 7:34:48 PM com.mongodb.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'org.mongodb.driver' component
Document{{_id=6671940023a737292958b8b3, First_Name=pavan, Last_Name=kaki, mark=90, age=17}}
```

Q2.

```
package mongo_db;
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.MongoIterable;
```

```
public class sampletest1{


            public static void main(String[] args) {
                        MongoClient mongoClient = MongoClients.create
("mongodb://localhost:27017");
                  MongoDatabase database = mongoClient.getDatabase("saturday");
                  database.createCollection("students");
                  MongoCollection<Document> collection = database.getCollection("students");


                  Document document = new Document("First_Name", "pavan")
                     .append("Last_Name", "kaki")
                     .append("mark", 90)
                     .append("age", 17);
                  collection.insertOne(document);
                  List<Document> documents = new ArrayList<Document>();


                  documents.add(new Document("First_Name", "deekshitha")
                     .append("Last_Name", "bobburi")
                     .append("mark", 80)
                     .append("age", 18));


                  documents.add(new Document("First_Name", "vyshu")
                     .append("Last_Name", "manne")
                    .append("mark", 70)
                     .append("age", 19));


                  documents.add(new Document("First_Name", "mohith")
                     .append("Last_Name", "nandika")
                      .append("mark",60)
                  .append("age",20));
                collection.insertMany(documents);
                FindIterable<Document> allDocuments = collection.find().sort(new
BasicDBObject("mark",-1)).limit(1);
                for (Document doc : allDocuments) {
                      System.out.println(doc);
                }
              }
}
```

OUTPUT:



```
<terminated> sampletest2 [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (Jun 18, 2024, 7:40:23 PM – 7:40:2
Jun 18, 2024 7:40:24 PM com.mongodb.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'org.mongodb.driver' component
Document{{_id=103, First_Name=dheeraj, Last_Name=pavan, salary=2000, age=20}}
```

TASK

Q1. 1)Price should be grater than either 799 or ram is greater than 12

db.products1.find({$or: [{ price: { $gt: 799 } },{ "spec.ram": { $gt: 12 } }]})

2)Find products that do not have "white" as a color and are priced below 800

db.products1.find({color: { $nin: ["white"] },price: { $lt: 800 }})

3)select products with either blue colour and storage not less than 128

db.products1.find({color: "blue","storage": { $gte: 128 }})

4)print the name and date of product whose ram is neither 4 nor the product price is

db.products1.find({$and: [{ "spec.ram": { $ne: 4 } },{ price: { $gte: 799 } }]}, { name: 1, releaseDate: 1 })

5)print the names of products whose screen is either greater than 7 or color is white

db.products1.find({$or: [{ "spec.screen": { $gt: 7 } },{ color: "white" }]}, { name: 1 })

6)print the name , screen size and color of products whose color has no gold in it.

```
db.products1.find({ color: { $nin: ["gold"] } }, { name: 1, "spec.screen": 1, color: 1 })
```

7)Find products that have either "white" or "black" as a color option and are priced below 800.

```
db.product1.find({$or: [{ color: "white" },{ color: "black" }],price: { $lt: 800 }})
```

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

```
db.products1.find({$and: [{ color: { $nin: ["gold"] } },{ $or: [{ price: { $lt: 700 } },{ storage: {
$in: [512] } }] }]})
```

9)Find products that have both a RAM size greater than 8GB and a CPU speed less than 2 GHz, or do not have a storage option of 256GB.

```
db.products.find({$or: [{ $and: [{ "spec.ram": { $gt: 8 } },{ "spec.cpu": { $lt: 2 } }] },{ storage: {
$nin: [256] } }]})
```

10) Price should be grater than either 799 or ram is greater than 12

```
db.products1.find({$or: [{ price: { $gt: 799 } },{ "spec.ram": { $gt: 12 } }]})
```