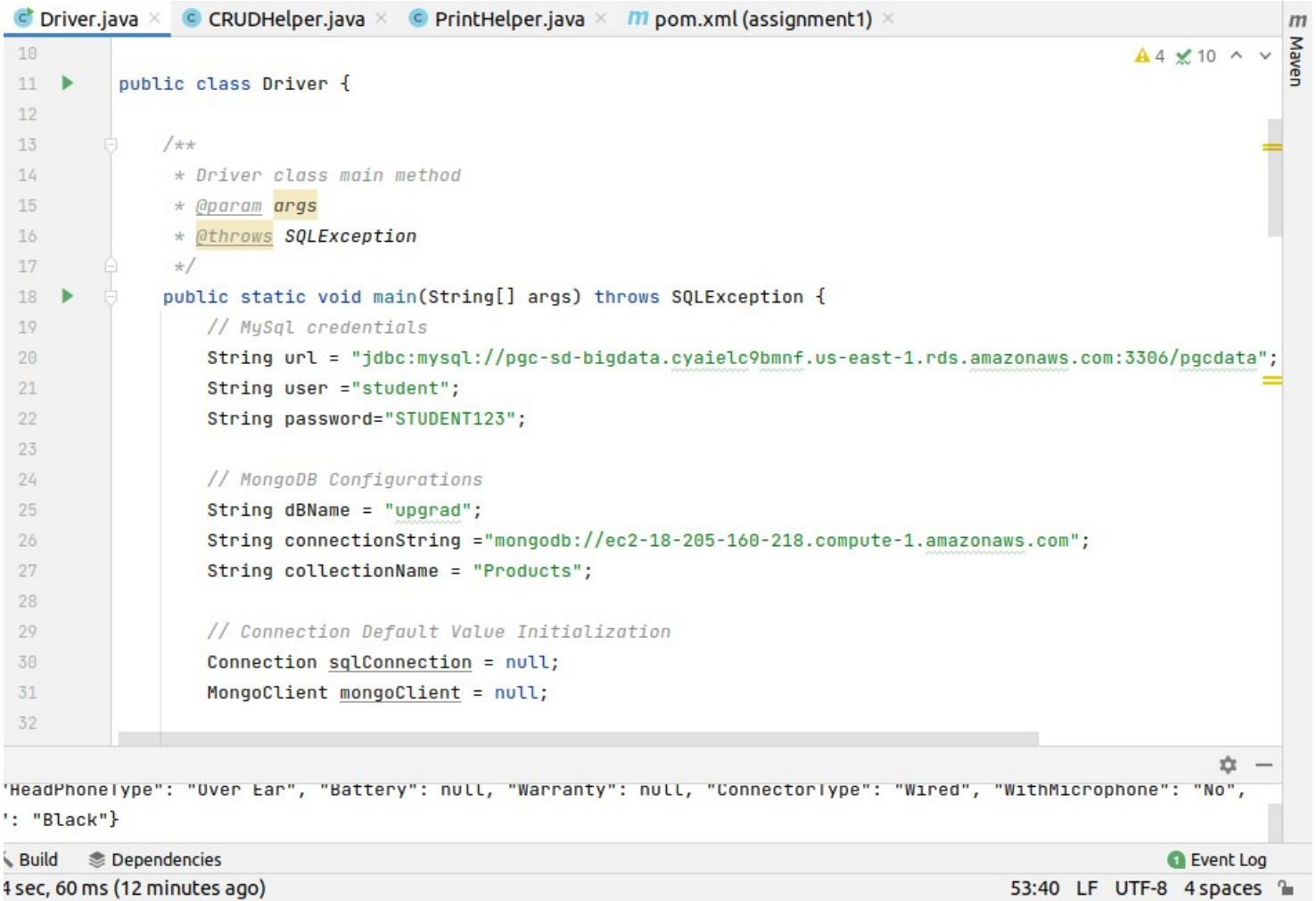
LOGIC

- To read the data from MySQL (RDS) to MongoDB using the Java API make both connections Mysql and mongodb.
- For mysql we requred a url,username and password.
- For mongodb we required database name,connecting string and collection name where connecting string is nothing but Public IPv4 DNS address which is running on ec2 instance.
- Both mysql and mongodb connection default value is null.


Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
Maven
4
10
public class Driver {
/**
* Driver class main method
* @param args
* @throws SQLException
*/
public static void main(String[] args) throws SQLException {
// MySql credentials
String url = "jdbc:mysql://pgc-sd-bigdata.cyaielc9bmnf.us-east-1.rds.amazonaws.com:3306/pgcdata";
String user ="student";
String password="STUDENT123";
// MongoDB Configurations
String dBName = "upgrad";
String connectionString ="mongodb://ec2-18-205-160-218.compute-1.amazonaws.com";
String collectionName = "Products";
// Connection Default Value Initialization
Connection sqlConnection = null;
MongoClient mongoClient = null;
'HeadPhoneType": "Over Ear", "Battery": null, "Warranty": null, "ConnectorType": "Wired", "WithMicrophone": "No",
': "Black"}
Build
Dependencies
Event Log
4 sec, 60 ms (12 minutes ago)
53:40 LF UTF-8 4 spaces

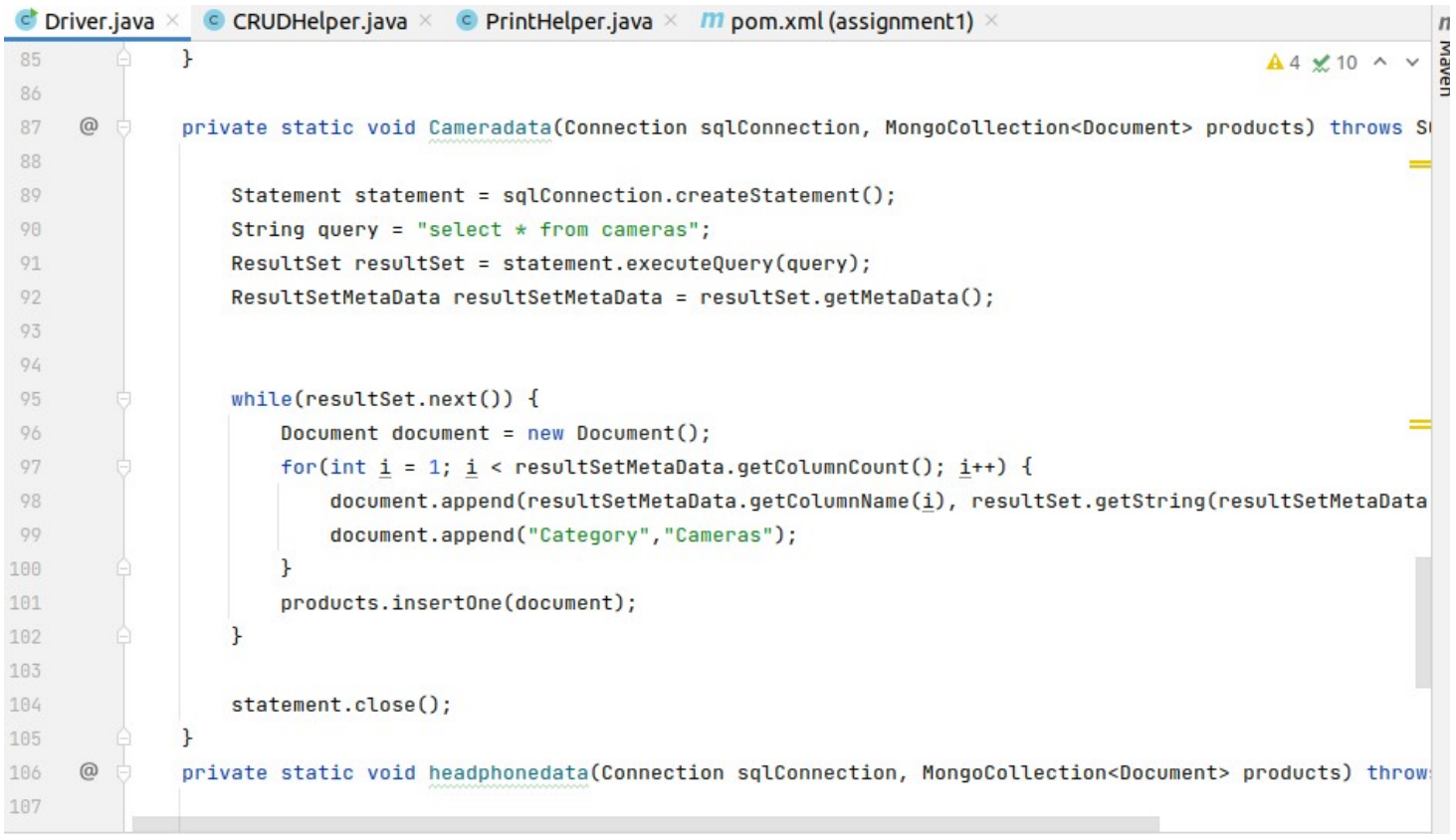
- Then after we will creating a database connection using url,user,and password to connect with database.
- Url is connectiong string which is provided by rds and ec2 instance.
- To import data into mongodb we create a document name as products.
- After reading data from rds we will import data into mongodb using user define functions.


Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
Maven
4
10
try {
// Creating database connections
sqlConnection = DriverManager.getConnection(url,user,password);
mongoClient = MongoClients.create(connectionString);
// Import data into MongoDb
MongoDatabase db = mongoClient.getDatabase(dBName);
MongoCollection<Document> Products = db.getCollection(collectionName);
Mobiledata(sqlConnection,Products);
Cameradata(sqlConnection,Products);
headphonedata(sqlConnection,Products);
// List all products in the inventory
CRUDHelper.displayAllProducts(Products);
// Display top 5 Mobiles
CRUDHelper.displayTop5Mobiles(Products);
// Display products ordered by their categories in Descending Order Without autogenerated Id
CRUDHelper.displayCategoryOrderedProductsDescending(Products);
// Display product count in each category
CRUDHelper.displayProductCountByCategory(Products);
// Display wired headphonesCRUDHelper.displayWiredHeadphones(Products);
}
catch(Exception ex) {
System.out.println("Got Exception.");

- The created method takes 2 parameters namely a SQL connection, Mongo Db Collection.
- We create a statement which will help to execute a query.
- “select *from mobiles” query will fetch all the data from mobiles category.
- Create a result set to store the data so we can do operations as per our requirements.
- Now to append resultset data we will create a document.
- Now add additional key-value pair “document.append("Category","Mobile")” to differentiate categories.
- I give mongodb collection name as products so we use “products.insertOne(document);” command to insert the Document in the mongo db collection
- Close a statement using statement.close();


Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
Maven
4
10
}
}
private static void Mobiledata(Connection sqlConnection,MongoCollection<Document> products) throws SQ
Statement statement = sqlConnection.createStatement();
String query = "select * from mobiles";
ResultSet resultSet = statement.executeQuery(query);
ResultSetMetaData resultSetMetaData = resultSet.getMetaData();
while(resultSet.next()) {
Document document = new Document();
for(int i = 1; i < resultSetMetaData.getColumnCount(); i++) {
document.append(resultSetMetaData.getColumnName(i), resultSet.getString(resultSetMetaData
document.append("Category","mobiles");
}
products.insertOne(document);
}
statement.close();
}

- We create three functions to perform perticular operations.
- For mobiles,cameras and headphones.and these all functions working are same.
- We use a statement for sql connection.
- We use a resultset to store result.
- ResultSetMetaData is an interface in java. sql package of JDBC API which is used to get the metadata about a ResultSet object.

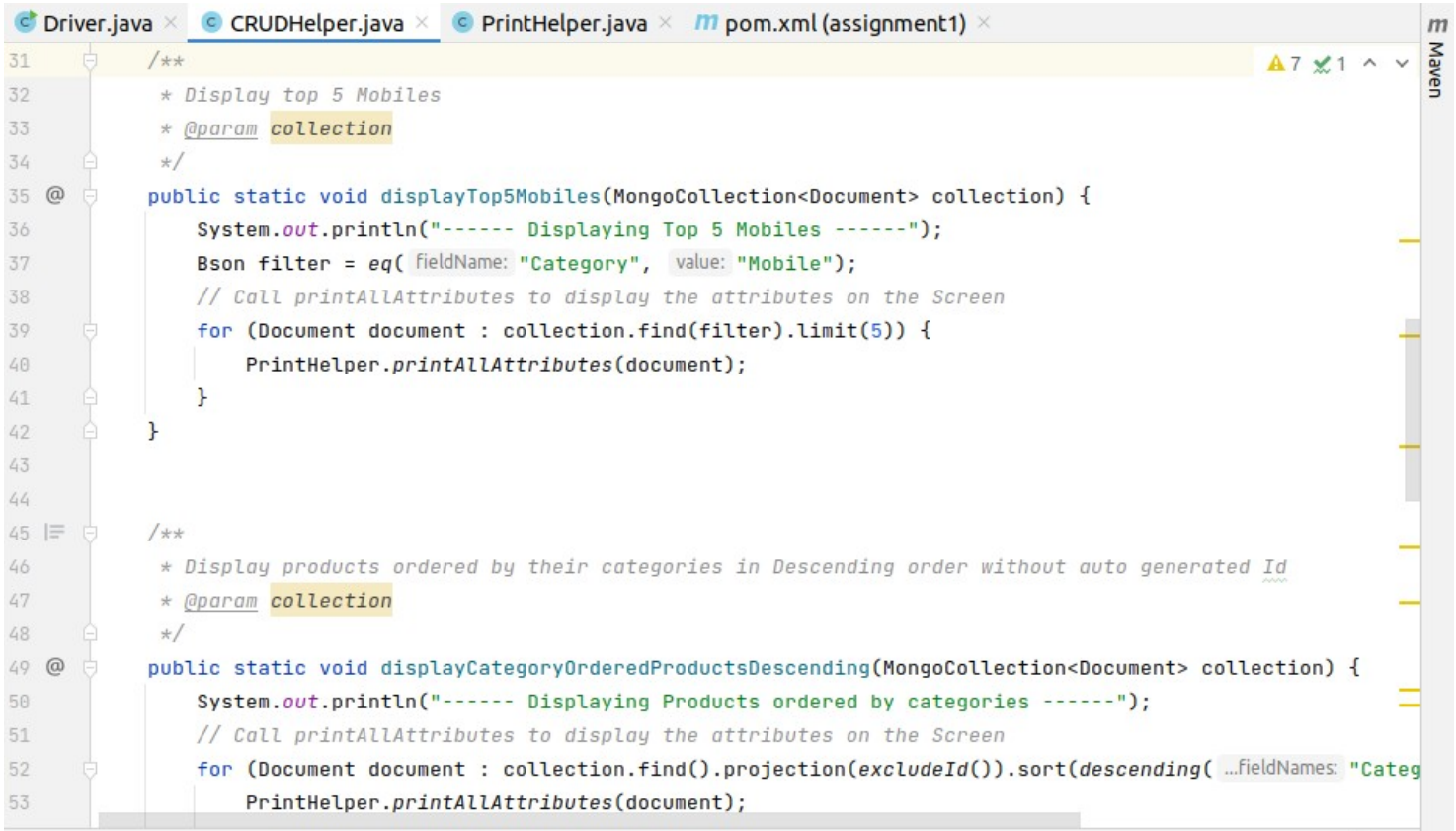

Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
Maven
4
10
}
private static void Cameradata(Connection sqlConnection, MongoCollection<Document> products) throws S
Statement statement = sqlConnection.createStatement();
String query = "select * from cameras";
ResultSet resultSet = statement.executeQuery(query);
ResultSetMetaData resultSetMetaData = resultSet.getMetaData();
while(resultSet.next()) {
Document document = new Document();
for(int i = 1; i < resultSetMetaData.getColumnCount(); i++) {
document.append(resultSetMetaData.getColumnName(i), resultSet.getString(resultSetMetaData
document.append("Category","Cameras");
}
products.insertOne(document);
}
statement.close();
}
private static void headphonedata(Connection sqlConnection, MongoCollection<Document> products) throw


Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
106 @ private static void headphonedata(Connection sqlConnection, MongoCollection<Document> p
4 10
107
108
109 Statement statement = sqlConnection.createStatement();
110 String query = "select * from headphones";
111 ResultSet resultSet = statement.executeQuery(query);
112 ResultSetMetaData resultSetMetaData = resultSet.getMetaData();
113
114
115 while(resultSet.next()) {
116 Document document = new Document();
117 for(int i = 1; i < resultSetMetaData.getColumnCount(); i++) {
118 document.append(resultSetMetaData.getColumnName(i), resultSet.getString(resultSetMetaData
119 document.append("Category","Headphones");
120 }
121 products.insertOne(document);
122 }
123
124 statement.close();
125 }
126
127 }

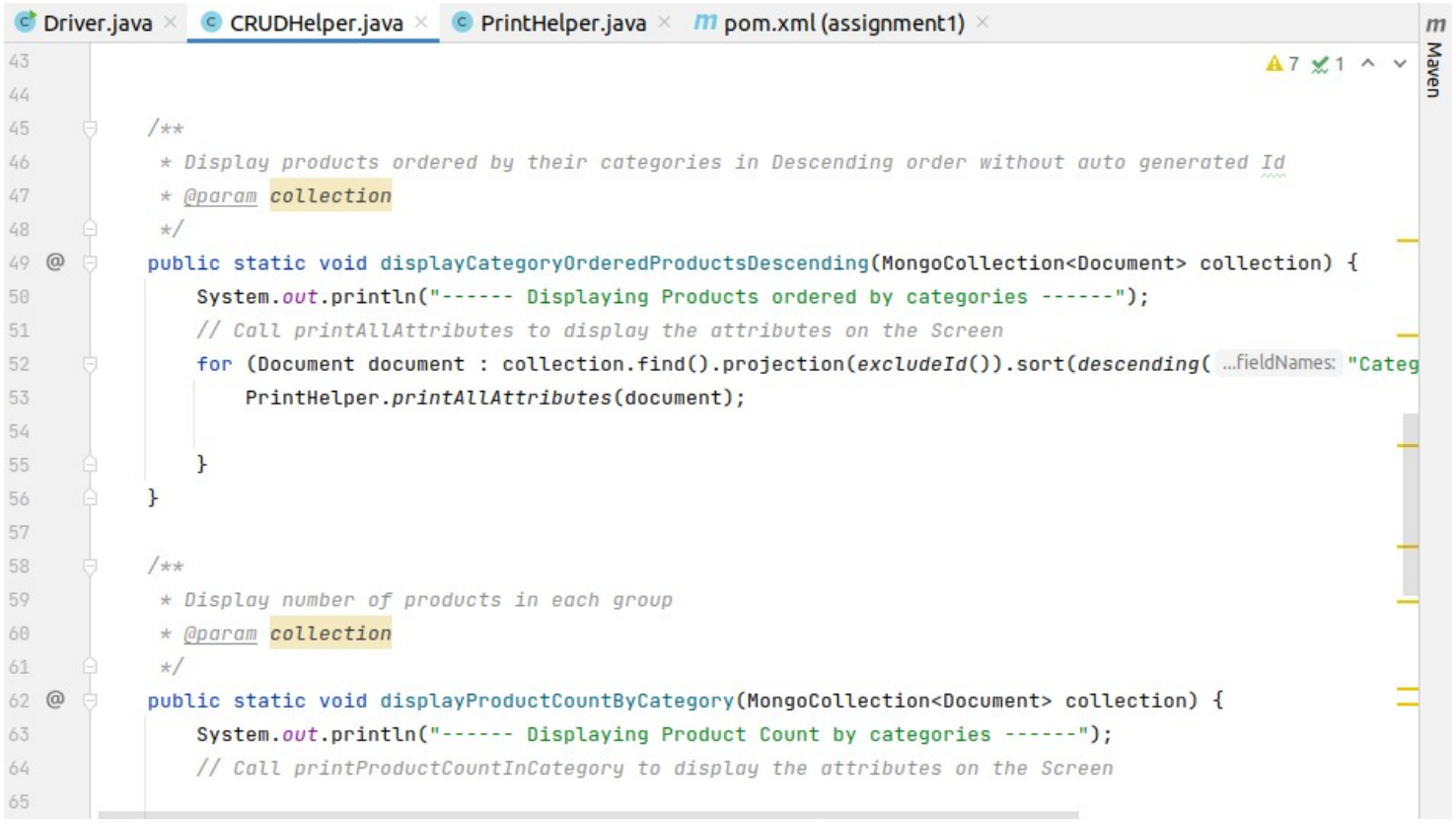
- “collection.find().projection(include("_id","ProductId","Category","Manufacturer","Title")))”
- This is our first query which will print common field across database.
- It is fetch id,productid,category,manufacturer and title from collection.


Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
Maven
public class CRUDHelper {
/**
* Display ALl products
* @param collection
*/
public static void displayAllProducts(MongoCollection<Document> collection) {
System.out.println("------ Displaying All Products ------");
// // Call printSingleCommonAttributes to display the attributes on the Screen
for (Document document : collection.find().projection(include(...fieldNames: "_id","ProductId","Category
PrintHelper.printAllAttributes(document);
}
}
/**
* Display top 5 Mobiles
* @param collection
*/
public static void displayTop5Mobiles(MongoCollection<Document> collection) {
System.out.println("------ Displaying Top 5 Mobiles ------");
Bson filter = eq(fieldName: "Category", value: "Mobile");
// Call printAllAttributes to display the attributes on the Screen

- “collection.find(filter).limit(5)”
- This is our second query it will create a bson filter using filter we can use limit function to find limited data.
- This query Display top 5 Mobiles.


Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
Maven
/**
* Display top 5 Mobiles
* @param collection
*/
public static void displayTop5Mobiles(MongoCollection<Document> collection) {
System.out.println("------ Displaying Top 5 Mobiles ------");
Bson filter = eq(fieldName: "Category", value: "Mobile");
// Call printAllAttributes to display the attributes on the Screen
for (Document document : collection.find(filter).limit(5)) {
PrintHelper.printAllAttributes(document);
}
}
/**
* Display products ordered by their categories in Descending order without auto generated Id
* @param collection
*/
public static void displayCategoryOrderedProductsDescending(MongoCollection<Document> collection) {
System.out.println("------ Displaying Products ordered by categories ------");
// Call printAllAttributes to display the attributes on the Screen
for (Document document : collection.find().projection(excludeId()).sort(descending(...fieldNames: "Categ
PrintHelper.printAllAttributes(document);

- collection.find().projection(excludeId()).sort(descending("Category")))
- This is our third query.which will display products ordered by their categories in Descending order without auto generated Id.
- excludeId() will exclude the id field and will fetch all the other fields data
- sort(descending(“Category”))) method will sort the fetched data in descending order based on the categories of the products.


Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
Maven
43
44
45 /**
46 * Display products ordered by their categories in Descending order without auto generated Id
47 * @param collection
48 */
49 public static void displayCategoryOrderedProductsDescending(MongoCollection<Document> collection) {
50 System.out.println("------ Displaying Products ordered by categories ------");
51 // Call printAllAttributes to display the attributes on the Screen
52 for (Document document : collection.find().projection(excludeId()).sort(descending(...fieldNames: "Categ
53 PrintHelper.printAllAttributes(document);
54
55 }
56 }
57
58 /**
59 * Display number of products in each group
60 * @param collection
61 */
62 public static void displayProductCountByCategory(MongoCollection<Document> collection) {
63 System.out.println("------ Displaying Product Count by categories ------");
64 // Call printProductCountInCategory to display the attributes on the Screen
65

- This is our forth query where we use aggregation method.
- cursor.hasNext() print the values till cursor has a next value.
- Mongo cursor is document type and then aggregate the field by grouping them based on Category and find the count of all the products in all the categories.


Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
Maven
}
}
/**
* Display number of products in each group
* @param collection
*/
public static void displayProductCountByCategory(MongoCollection<Document> collection) {
System.out.println("------ Displaying Product Count by categories ------");
// Call printProductCountInCategory to display the attributes on the Screen
MongoCursor<Document> cursor = collection.aggregate(
Arrays.asList(
Aggregates.group(id: "$Category",
Accumulators.sum(fieldName: "Count", expression: 1)
))).cursor();
while(cursor.hasNext()) {
PrintHelper.printProductCountInCategory(cursor.next());
}
}
/**
* Display Wired Headphones

- collection.find(filter);
- This is our fifth query where we use Bson filter to keep condition.
- Bson filter =eq("ConnectorType","Wired");
- Using Above filter we can get all Headphones whose connector type is wired.


Driver.java
CRUDHelper.java
PrintHelper.java
pom.xml (assignment1)
Maven
7
1
PrintHelper.printProductCountInCategory(cursor.next());
}
}
/**
* Display Wired Headphones
* @param collection
*/
public static void displayWiredHeadphones(MongoCollection<Document> collection) {
System.out.println("------ Displaying Wired headphones ------");
// Call printAllAttributes to display the attributes on the Screen
Bson filter =eq(fieldName: "ConnectorType", value: "Wired");
for (Document document : collection.find(filter)) {
PrintHelper.printAllAttributes(document);
}
}
}