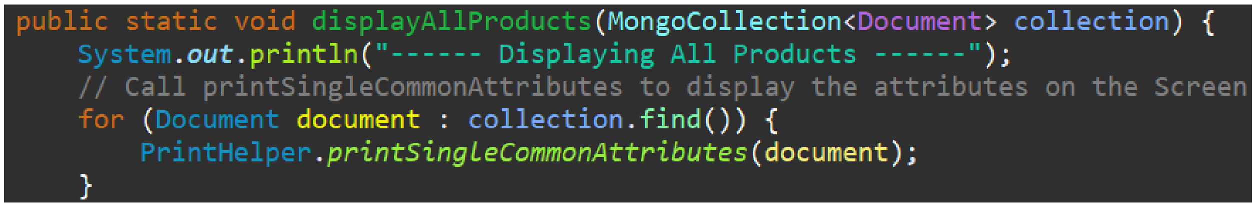
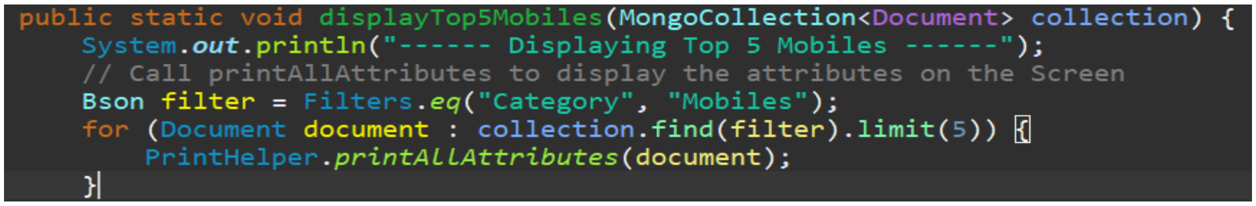
<u>Assignment Logic:</u>

1. Performed the ETL logic as mentioned below.
   Extraction - Connected to MySQL Database and extracted the data from all three tables (mobiles, cameras, and headphones).
   Transformation – Transformed the Flat data from MySQL data to the format as required by MongoDB using the Document Objects and added the extra attribute 'Category' to each data.
   Load – Loaded the transformed data in to the MongoDB.
2. Then updated the CRUDHelper.java as below
   1. **displayAllProducts**: Used the **collection.find()** method to fetch all the data from MongoDB



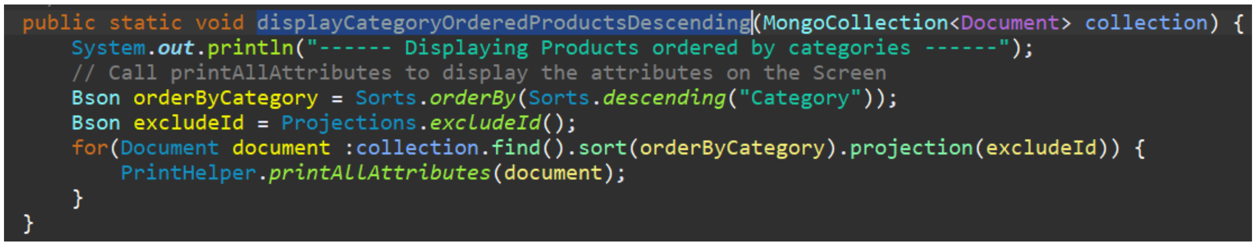
```java
public static void displayAllProducts(MongoCollection<Document> collection) {
    System.out.println("------ Displaying All Products ------");
    // Call printSingleCommonAttributes to display the attributes on the Screen
    for (Document document : collection.find()) {
        PrintHelper.printSingleCommonAttributes(document);
    }
```

   2. **displayTop5Mobiles**: Created the **BSON filter** to just extract the mobile data and the used the **find** method with filter object passed as a parameter to the find method and added the **limit** method to extract just the top 5 data.



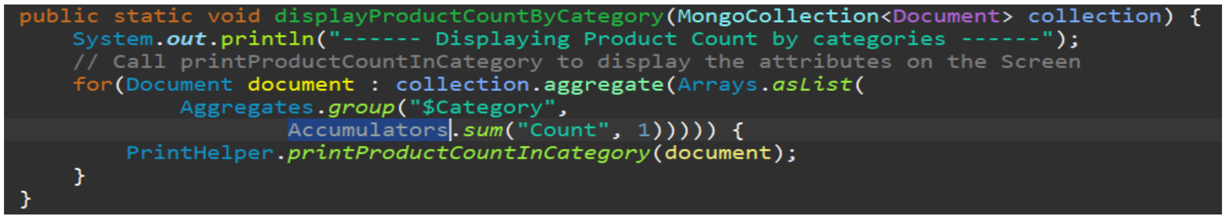
```java
public static void displayTop5Mobiles(MongoCollection<Document> collection) {
    System.out.println("------ Displaying Top 5 Mobiles ------");
    // Call printAllAttributes to display the attributes on the Screen
    Bson filter = Filters.eq("Category", "Mobiles");
    for (Document document : collection.find(filter).limit(5)) {
        PrintHelper.printAllAttributes(document);
    }
```

   3. **displayCategoryOrderedProductsDescending**: Created **two BSON objects one to sor**t the data using the category field in **descending** order and used the **Projections** class to **exclude** the **ids** from the result set.



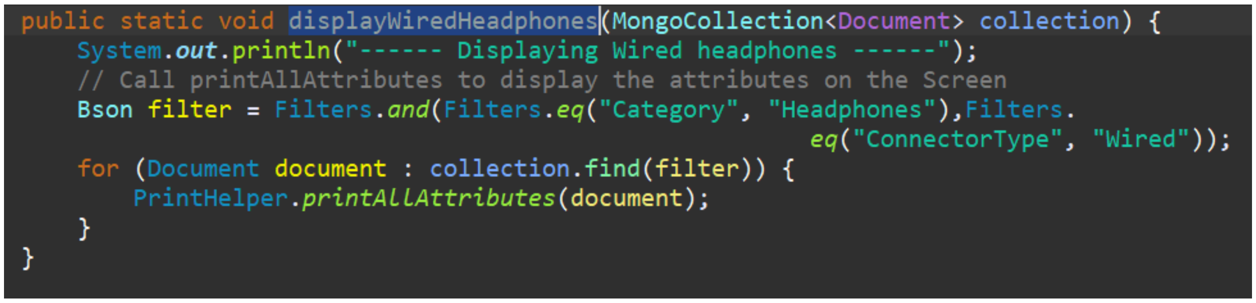
```java
public static void displayCategoryOrderedProductsDescending(MongoCollection<Document> collection) {
    System.out.println("------ Displaying Products ordered by categories ------");
    // Call printAllAttributes to display the attributes on the Screen
    Bson orderByCategory = Sorts.orderBy(Sorts.descending("Category"));
    Bson excludeId = Projections.excludeId();
    for(Document document :collection.find().sort(orderByCategory).projection(excludeId)) {
        PrintHelper.printAllAttributes(document);
    }
}
```

   4. **displayProductCountByCategory**: Used the **aggregate** method from the collections class to group the result set by Category and used the **Accumulators.sum** to count the number of objects in each category.



```java
public static void displayProductCountByCategory(MongoCollection<Document> collection) {
    System.out.println("------ Displaying Product Count by categories ------");
    // Call printProductCountInCategory to display the attributes on the Screen
    for(Document document : collection.aggregate(Arrays.asList(
            Aggregates.group("$Category",
                    Accumulators.sum("Count", 1))))) {
        PrintHelper.printProductCountInCategory(document);
    }
}
```

5. **displayWiredHeadphones**: Created **BSON object** to apply the filter on Category = ‘Headphones’ and the COnnectorType = ‘Wired’ which brought result of just wired headset.



```
public static void displayWiredHeadphones(MongoCollection<Document> collection) {
    System.out.println("------ Displaying Wired headphones ------");
    // Call printAllAttributes to display the attributes on the Screen
    Bson filter = Filters.and(Filters.eq("Category", "Headphones"),Filters.
                                                    eq("ConnectorType", "Wired"));
    for (Document document : collection.find(filter)) {
        PrintHelper.printAllAttributes(document);
    }
}
```