

Relational Databases with MySQL Week 5 Coding Assignment

Screenshots of Code:

Part 1

```
groceryListApp.java × GLDao.java groceryShop.java
1 package codingAssignmentPart1;
2
3 import java.util.Collections;
4 import java.util.List;
5 import groceryListDao.GLDao;
6 import groceryListDao.groceryShop;
7
8 public class groceryListApp {
9
10     private static GLDao gld = GLDao.getGLDao();
11
12     //Student Note
13     //Rather than following the rather pedantic example of the video
14     //I'm just going to pass the functional lambda and method reference
15     //to Collections.sort which serves to illustrate the requirements
16     //of the assignment. Construction an entire model layer is a bit
17     //excessive for this.
18
19     public static void main(String[] args) {
20         //Iterate List
21         System.out.println("--UnSorted---");
22         List<groceryShop> gList = gld.getGroceryShops();
23         gList.forEach((item) -> System.out.println(item));
24
25         //Making an immutable copy of our original list so that we can sort again
26         List<groceryShop> uList = List.copyOf(gList);
27
28         gLLambdaSort(gList);
29         System.out.println("--Sorted via Lambda---");
30         gList.forEach((item) -> System.out.println(item));
31
32         Collections.copy(gList,uList);
33         System.out.println("--UnSorted---");
34         gList.forEach((item) -> System.out.println(item));
35
36         System.out.println("--Sorted via Method Reference---");
37         gLMethodReferenceSort(gList);
38         gList.forEach((item) -> System.out.println(item));
39     }
40
41     private static void gLLambdaSort(List<groceryShop> shops) {
42         Collections.sort(shops,(s1,s2)->groceryShop.compare(s1,s2));
43         return;
44     }
45
46     private static void gLMethodReferenceSort(List<groceryShop> shops) {
47         Collections.sort(shops,groceryShop::compare);
48         return;
49     }
50 }
```

```

1 package groceryListDao;
2
3 public class groceryShop {
4
5     private String sName;
6
7     public String getName() {
8         return sName;
9     }
10
11     @Override
12     public String toString() {
13         return this.getClass() + " [shop Name = " + sName + "]\n";
14     }
15
16     public groceryShop(String str) {
17         this.sName = str;
18     }
19
20     public int compareGroceryShop(groceryShop g1, groceryShop g2) {
21         return g1.sName.compareTo(g2.sName);
22     }
23
24     //to meet assignment requirements create "compare"
25     public static int compare(groceryShop g1, groceryShop g2)
26     {
27         return g1.sName.compareTo(g2.sName);
28     }
29 }

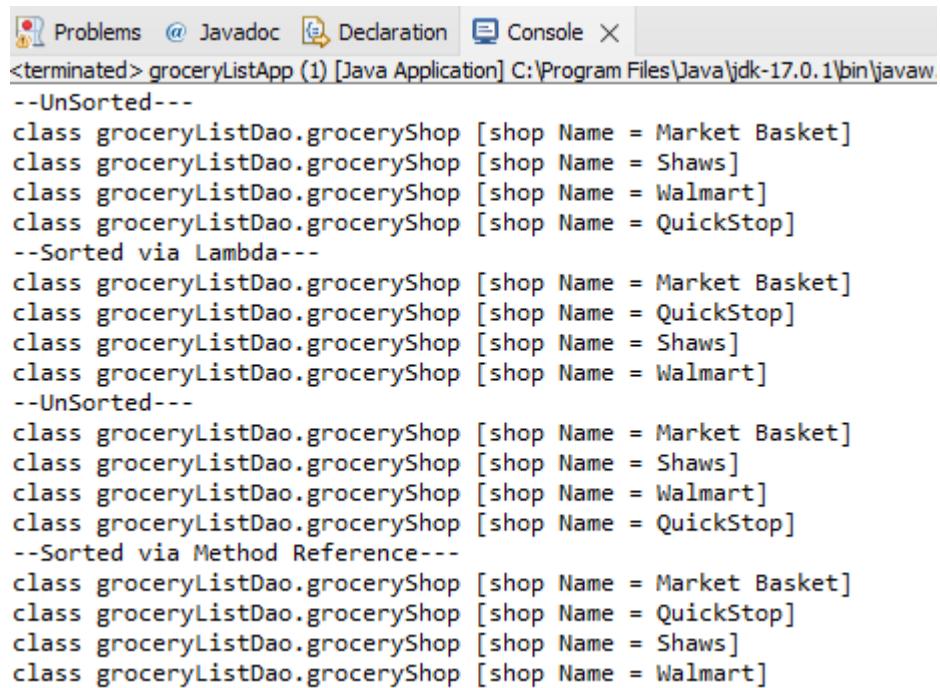
```

```

1 package groceryListDao;
2
3
4 import java.util.ArrayList;
5
6
7
8 public class GLDao {
9
10     private static GLDao instance;
11     private static List<groceryShop> shops;
12
13     private GLDao () {
14         shops = new ArrayList<groceryShop>(Arrays.asList(
15             new groceryShop("Market Basket"),
16             new groceryShop("Shaws"),
17             new groceryShop("Walmart"),
18             new groceryShop("QuickStop")));
19     }
20
21     public static GLDao getGLDao() {
22         if ( instance == null) {
23             instance = new GLDao();
24         }
25         return instance;
26     }
27
28     public List<groceryShop> getGroceryShops() {
29         return shops;
30     }
31
32 }

```

Screenshots of Running Application Results:



```
<terminated> groceryListApp (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw
--UnSorted---
class groceryListDao.groceryShop [shop Name = Market Basket]
class groceryListDao.groceryShop [shop Name = Shaws]
class groceryListDao.groceryShop [shop Name = Walmart]
class groceryListDao.groceryShop [shop Name = QuickStop]
--Sorted via Lambda---
class groceryListDao.groceryShop [shop Name = Market Basket]
class groceryListDao.groceryShop [shop Name = QuickStop]
class groceryListDao.groceryShop [shop Name = Shaws]
class groceryListDao.groceryShop [shop Name = Walmart]
--UnSorted---
class groceryListDao.groceryShop [shop Name = Market Basket]
class groceryListDao.groceryShop [shop Name = Shaws]
class groceryListDao.groceryShop [shop Name = Walmart]
class groceryListDao.groceryShop [shop Name = QuickStop]
--Sorted via Method Reference---
class groceryListDao.groceryShop [shop Name = Market Basket]
class groceryListDao.groceryShop [shop Name = QuickStop]
class groceryListDao.groceryShop [shop Name = Shaws]
class groceryListDao.groceryShop [shop Name = Walmart]
```

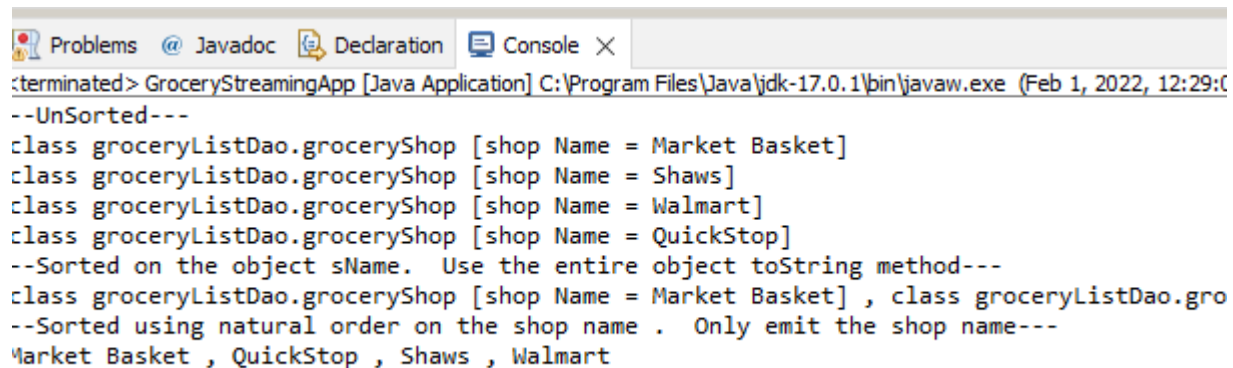
Screenshots of Code:

Part Two

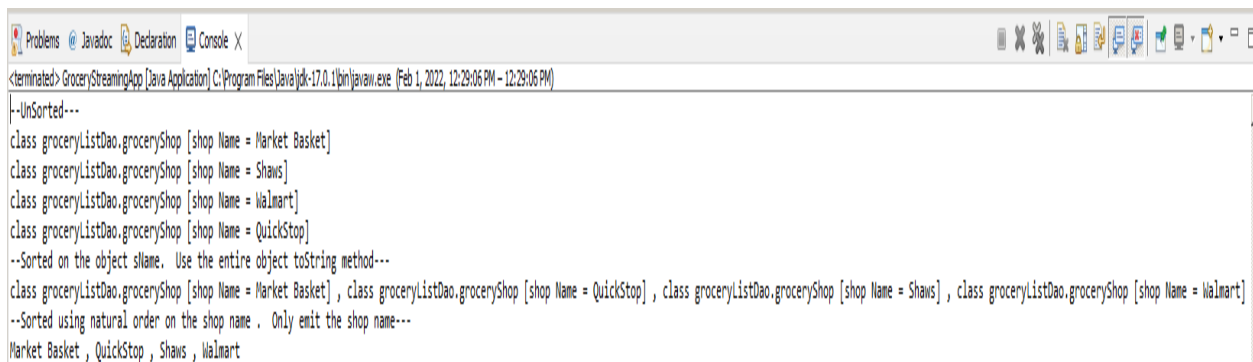
//Student Note : Resused the GLDao and groceryShop classes from Part 1

```
GroceryStreamingApp.java X
1 package codingAssignmentPart2;
2
3 import java.util.Comparator;
4
5
6
7
8
9 public class GroceryStreamingApp {
10     private static GLDao gld = GLDao.getGLDao();
11
12     public static void main(String[] args) {
13
14         System.out.println("--UnSorted---");
15         List<groceryShop> gList = gld.getGroceryShops();
16         gList.forEach(System.out::println);
17         //
18         // Convert to a Stream using my compare vs natural compare / Sort/ add a , between the elements
19         // Using my compare as the groceryShop.toString() emits the className and some formating as well as the name element.
20         // grpceryShop.compare does sort in natural order based on the sName (shop name) member of the groceryShop class
21         System.out.println("--Sorted on the object sName. Use the entire object toString method---");
22         String s2 = gList.stream().sorted((o1, o2)->groceryShop.compare(o1,o2)).map(e -> e.toString())
23             .collect(Collectors.joining(" ", ""));
24         System.out.println(s2);
25         //map with groceryShop.getSname which only emits the shop name then sort then collect
26         //This is the order the assignment wanted so here it is.
27         System.out.println("--Sorted using natural order on the shop name . Only emit the shop name---");
28         String s3 = gList.stream().map(e -> e.getSname()).sorted(Comparator.naturalOrder())
29             .collect(Collectors.joining(" ", ""));
30         System.out.println(s3);
31     }
32 }
33 }
34
```

Screenshots of Running Application Results:



```
<terminated> GroceryStreamingApp [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Feb 1, 2022, 12:29:06 PM)
--UnSorted---
class groceryListDao.groceryShop [shop Name = Market Basket]
class groceryListDao.groceryShop [shop Name = Shaws]
class groceryListDao.groceryShop [shop Name = Walmart]
class groceryListDao.groceryShop [shop Name = QuickStop]
--Sorted on the object sName. Use the entire object toString method---
class groceryListDao.groceryShop [shop Name = Market Basket] , class groceryListDao.gro
--Sorted using natural order on the shop name . Only emit the shop name---
Market Basket , QuickStop , Shaws , Walmart
```



```
<terminated> GroceryStreamingApp [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Feb 1, 2022, 12:29:06 PM - 12:29:06 PM)
--UnSorted---
class groceryListDao.groceryShop [shop Name = Market Basket]
class groceryListDao.groceryShop [shop Name = Shaws]
class groceryListDao.groceryShop [shop Name = Walmart]
class groceryListDao.groceryShop [shop Name = QuickStop]
--Sorted on the object sName. Use the entire object toString method---
class groceryListDao.groceryShop [shop Name = Market Basket] , class groceryListDao.groceryShop [shop Name = QuickStop] , class groceryListDao.groceryShop [shop Name = Shaws] , class groceryListDao.groceryShop [shop Name = Walmart]
--Sorted using natural order on the shop name . Only emit the shop name---
Market Basket , QuickStop , Shaws , Walmart
```

Screenshots of Code:

Part Three

```
OptionalApp.java × Model.java
1 package codingAssignmentPart3;
2
3 import java.util.NoSuchElementException;
4 import java.util.Optional;
5
6 public class OptionalApp {
7
8     public static void main(String[] args) {
9         //Model all three (3) modes of passing optionals -
10        //Complete optional object, Empty optional, Empty optional object
11        //the assignment only asks for the first and third of these.
12        Model model_1 = new Model("Part3");
13        Model model_2 = null;
14        Model model_3 = new Model(null);
15        Optional<Model> opt1 = Optional.ofNullable(model_1);
16        Optional<Model> opt2 = Optional.ofNullable(model_2); //this results in the same functionality as using .empty()
17        Optional<Model> optE = Optional.empty();
18        Optional<Model> opt3 = Optional.ofNullable(model_3);
19
20        //Go through and test calling unwrap on all these models -- ugly try/catch blocks
21        Model m;
22        try {
23            m = model_1.unwrap(opt1); //Should really make the unwrap a static rather than carrying it around
24            System.out.println(m);
25        } catch (NoSuchElementException e)
26        {
27            System.out.println(e.getMessage());
28        }
29        try {
30            m = model_1.unwrap(opt2); //need to use a real object to access the method -- should have been static !
31            System.out.println(m);
32        } catch (NoSuchElementException e)
33        {
34            System.out.println(e.getMessage());
35        }
36        try {
37            m = model_1.unwrap(optE); //need to use a real object to access the method -- should have been static !
38            System.out.println(m);
39        } catch (NoSuchElementException e)
40        {
41            System.out.println(e.getMessage());
42        }
43        try {
44            m = model_3.unwrap(opt3); //need to use a real object to access the method -- should have been static !
45            System.out.println(m);
46        } catch (NoSuchElementException e)
47        {
48            System.out.println(e.getMessage());
49        }
50
51        //Abstract calling into the object optional and the object itself
52        //still having to use an instantiated Model to access the methods //from face
53        System.out.println("\n" + "Use MethodB to call unwrap and then print" + "\n");
54        model_1.methodB(opt1);
55        model_1.methodB(opt2);
56        model_1.methodB(opt3);
57    }
58
59 }
```

```

OptionalApp.java  Model.java X
1  package codingAssignmentPart3;
2
3  import java.util.NoSuchElementException;
4  import java.util.Optional;
5
6  public class Model {
7
8      private String name;
9
10     @Override
11     public String toString() {
12         return name;
13     }
14
15     public Model(String name) {
16         this.name = name;
17     }
18
19     public Model unwrap(Optional<Model> optionalModel) {
20         // if (optionalModel.isEmpty()) {
21         //     throw new IllegalStateException ("This space for rent");
22         // }
23         // return optionalModel.get();
24         Model m = optionalModel.orElseThrow(()->new NoSuchElementException ("-- Unwrap :: This space for rent--"));
25         return m;
26     }
27
28     public void methodB(Optional<Model> opt) {
29         try {
30             // Model m = opt.orElseThrow(()->new NoSuchElementException ("-- MethodB :: This space for rent--"));
31             Model m = unwrap(opt);
32             // great we have valid model object do we have a valid name in the model?
33             Optional<String> ostr = Optional.of(Optional.ofNullable(m.toString()).orElseThrow(
34                 ()-> new NoSuchElementException ("-- MethodB :: Null name in Model")));
35             System.out.println(ostr.get());
36         } catch (NoSuchElementException e)
37         {
38             System.out.println(e.getMessage());
39         }
40     }
41 }
42 }

```

Screenshots of Running Application Results:

```

Problems  @ Javadoc  Declaration  Console X
<terminated> OptionalApp [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.
Part3
-- Unwrap :: This space for rent--
-- Unwrap :: This space for rent--
null

Use MethodB to call unwrap and then print

Part3
-- Unwrap :: This space for rent--
-- MethodB :: Null name in Model

```

URL to GitHub Repository:

<https://github.com/david2joh/sqlWeek5.git>