Relational Databases with MySQL Week 5 Coding Assignment

Screenshots of Code:

Part 1

groceryListApp.java | GLDao.java | groceryShop.java

```java
package codingAssignmentPart1;

import java.util.Collections;
import java.util.List;
import groceryListDao.GLDao;
import groceryListDao.groceryShop;

public class groceryListApp {

    private static GLDao gld = GLDao.getGLDao();

    //Student Note
    //Rather than following the rather pedantic example of the video
    //I'm just going to pass the functional lamba and method reference
    //to Collections.sort which serves to illustrate the requirements
    //of the assignment.  Construction an entire model layer is a bit
    //excessive for this.

    public static void main(String[] args) {
        //Iterate List
        System.out.println("--UnSorted---");
        List<groceryShop> gList = gld.getGroceryShops();
        gList.forEach((item) -> System.out.println(item));

        //Making an immutable copy of our original list so that we can sort again
        List<groceryShop> uList  = List.copyOf(gList);

        glLambaSort(gList);
        System.out.println("--Sorted via Lambda---");
        gList.forEach((item) -> System.out.println(item));

        Collections.copy(gList,uList);
        System.out.println("--UnSorted---");
        gList.forEach((item) -> System.out.println(item));

        System.out.println("--Sorted via Method Reference---");
        glMethodReferenceSort(gList);
        gList.forEach((item) -> System.out.println(item));
    }

    private static void glLambaSort(List<groceryShop> shops) {
        Collections.sort(shops,(s1,s2)->groceryShop.compare(s1,s2));
        return;
        }

    private static void glMethodReferenceSort(List<groceryShop> shops) {
        Collections.sort(shops,groceryShop::compare);
        return;
        }
}
```

groceryListApp.java | GLDao.java | groceryShop.java

```java
package groceryListDao;

public class groceryShop {

    private String sName;

    public String getsName() {
        return sName;
    }

    @Override
    public String toString() {
        return this.getClass() +" [shop Name = " + sName + "]";
    }

    public groceryShop(String str) {
        this.sName = str;
        }

    public int compareGroceryShop(groceryShop g1, groceryShop g2) {
        return g1.sName.compareTo(g2.sName);
    }

    //to meet assignment requirements create "compare"
    public static int compare(groceryShop g1, groceryShop g2)
    {
        return  g1.sName.compareTo(g2.sName);
    }
}
```

GLDao.java | groceryShop.java | groceryListApp.java

```java
package groceryListDao;


import java.util.ArrayList;

public class GLDao {

    private static GLDao instance;
    private static List<groceryShop> shops;

    private GLDao () {
        shops = new ArrayList<groceryShop>(Arrays.asList(
                new groceryShop("Market Basket"),
                new groceryShop("Shaws"),
                new groceryShop("Walmart"),
                new groceryShop("QuickStop")));
    }

    public static GLDao getGLDao()  {
        if ( instance == null) {
            instance = new GLDao();
        }
        return instance;
    }

    public List<groceryShop> getGroceryShops()  {
        return shops;
    }


}
```

Screenshots of Running Application Results:

```
Problems  @ Javadoc  Declaration  Console ×
<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

```java
package codingAssignmentPart2;

import java.util.Comparator;

public class GroceryStreamingApp {
    private static GLDao gld = GLDao.getGLDao();

    public static void main(String[] args) {

        System.out.println("--UnSorted---");
        List<groceryShop> gList = gld.getGroceryShops();
        gList.forEach(System.out::println);
        //
        // Convert to a Stream using my compare vs natural compare / Sort/ add a , between the elements
        // Using my compare as the groceryShop.toString() emits the className and some formating as well as the name element.
        // grpceryShop.compare does sort in natural order based on the sName (shop name) member of the groceryShop class
        System.out.println("--Sorted on the object sName.  Use the entire object toString method---");
        String s2 = gList.stream().sorted((o1, o2)->groceryShop.compare(o1,o2)).map(e -> e.toString())
                .collect(Collectors.joining (" , "));
        System.out.println(s2);
        //map with groceryShop.getSname which only emits the shop name then sort then collect
        //This is the order the assignment wanted so here it is.
        System.out.println("--Sorted using natural order on the shop name .  Only emit the shop name---");
        String s3 = gList.stream().map(e -> e.getsName()).sorted(Comparator.naturalOrder())
                .collect(Collectors.joining (" , "));
        System.out.println(s3);
    }

}
```

Screenshots of Running Application Results:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> GroceryStreamingApp [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (Feb 1, 2022, 12:29:0
--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
```

```
Problems  @ Javadoc  Declaration  Console ×
<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

```java
package codingAssignmentPart3;

import java.util.NoSuchElementException;
import java.util.Optional;

public class OptionalApp {

    public static void main(String[] args) {
        //Model all three (3) modes of passing optionals -
        //Complete optional object, Empty optional, Empty optional object
        //the assignment only asks for the first and third of these.
        Model model_1 = new Model("Part3");
        Model model_2 = null;
        Model model_3 = new Model(null);
        Optional<Model> opt1 = Optional.ofNullable(model_1);
        Optional<Model> opt2 = Optional.ofNullable(model_2); //this results in the same functionality as using .empty(
        Optional<Model> optE = Optional.empty();
        Optional<Model> opt3 = Optional.ofNullable(model_3);

        //Go through and test calling unwrap on all these models -- ugly try/catch blocks
        Model m;
        try {
            m = model_1.unwrap(opt1);  //Should really make the unwrap a static rather than carrying it around
            System.out.println(m);
        } catch (NoSuchElementException e)
        {
            System.out.println(e.getMessage());
        }
        try {
            m = model_1.unwrap(opt2); //need to use a real object to access the method --  should have been static !
            System.out.println(m);
        } catch (NoSuchElementException e)
        {
            System.out.println(e.getMessage());
        }
        try {
            m = model_1.unwrap(optE); //need to use a real object to access the method --  should have been static !
            System.out.println(m);
        } catch (NoSuchElementException e)
        {
            System.out.println(e.getMessage());
        }
        try {
            m = model_3.unwrap(opt3); //need to use a real object to access the method --  should have been static !
            System.out.println(m);
        } catch (NoSuchElementException e)
        {
            System.out.println(e.getMessage());
        }

        //Abstract calling into the object optional and the object itself
        //still having to use an instantiated Model to access the methods /frownface
        System.out.println("\n" + "Use MethodB to call unwrap and then print" +"\n");
        model_1.methodB(opt1);
        model_1.methodB(opt2);
        model_1.methodB(opt3);
    }

}
```

OptionalApp.java | Model.java

```java
package codingAssignmentPart3;

import java.util.NoSuchElementException;
import java.util.Optional;

public class Model {

    private String name;

    @Override
    public String toString() {
        return name;
    }

    public Model(String name) {
        this.name = name;
    }

    public Model unwrap(Optional<Model> optionalModel) {
//      if (optionalModel.isEmpty()) {
//          throw new IllegalStateException ("This space for rent");
//      }
//      return optionalModel.get();
        Model m = optionalModel.orElseThrow(()->new NoSuchElementException ("-- Unwrap :: This space for rent--"));
        return m;
    }

    public void methodB(Optional<Model> opt) {
        try {
//          Model m = opt.orElseThrow(()->new NoSuchElementException ("-- MethodB :: This space for rent--"));
            Model m = unwrap(opt);
            // great we have  valid model object do we have a valid name in the model?
            Optional<String> ostr = Optional.of(Optional.ofNullable(m.toString()).orElseThrow(
                    ()-> new NoSuchElementException ("-- MethodB ::  Null name in Model")));
            System.out.println(ostr.get());
        } catch (NoSuchElementException e)
        {
            System.out.println(e.getMessage());
        }

    }
}
```

Screenshots of Running Application Results:

Problems | Javadoc | Declaration | Console

```
<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