Relational Databases with MySQL Week 11 Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: Complete the coding steps. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document to the repository. Additionally, push the Java project to the same repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

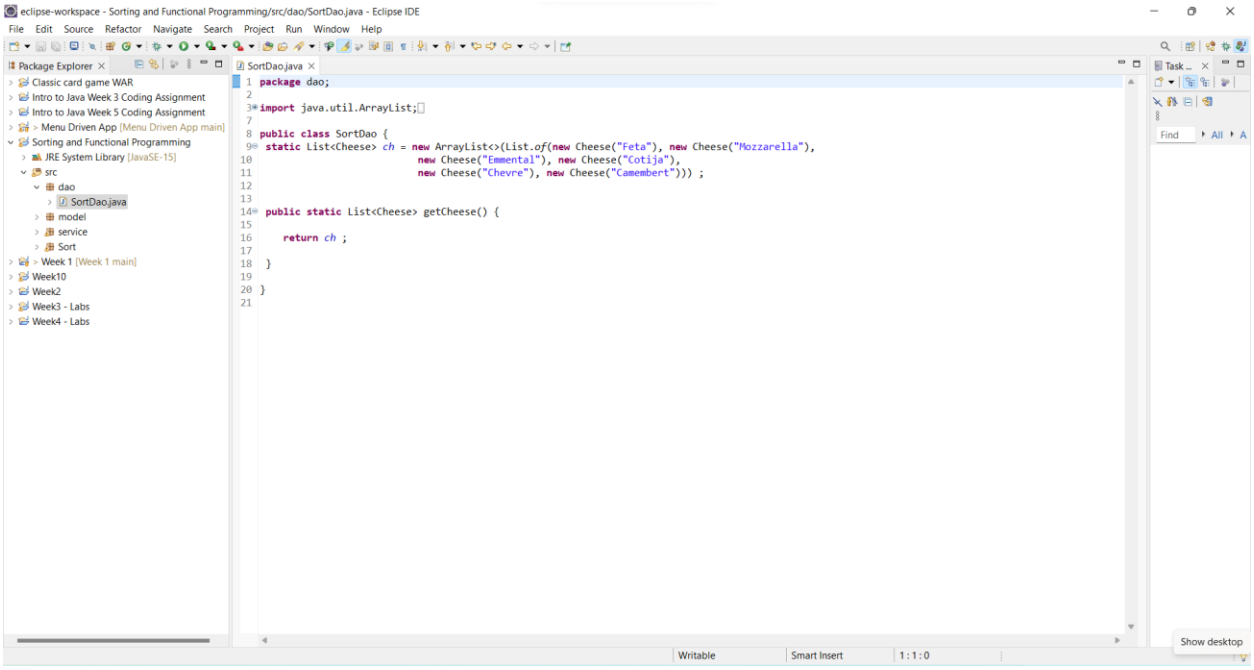
1. Create a class of whatever type you want (Animal, Person, Camera, Cheese, etc.).
    a. Do not implement the Comparable interface.
    b. Add a name instance variable so that you can tell the objects apart.
    c. Add getters, setters and/or a constructor as appropriate.
    d. Add a toString method that returns the name and object type (like "Pentax Camera").
    e. Create a static method named `compare` in the class that returns an int and takes two of the objects as parameters. Return -1 if parameter 1 is "less than" parameter 2. Return 1 if parameter 1 is "greater than" parameter 2. Return 0 if the two parameters are "equal".
    f. Create a static list of these objects, adding at least 4 objects to the list.
    g. In another class, write a method to sort the objects using a Lambda expression using the compare method you created earlier.
    h. Write a method to sort the objects using a Method Reference to the compare method you created earlier.
    i. Create a main method to call the sort methods.
    j. Print the list after sorting (System.out.println).

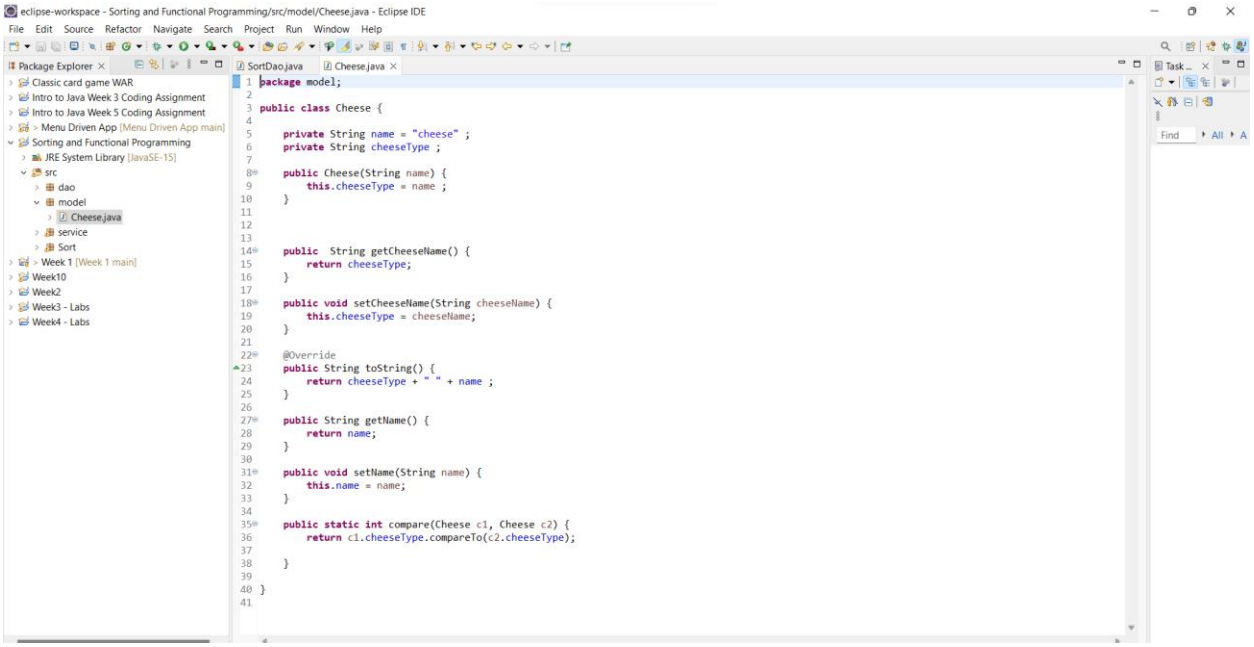
2. Create a new class with a main method. Using the list of objects you created in the prior step.
    a. Create a Stream from the list of objects.
    b. Turn the Stream of object to a Stream of String (use the map method for this).
    c. Sort the Stream in the natural order. (Note: The String class implements the Comparable interface, so you won't have to supply a Comparator to do the sorting.)
    d. Collect the Stream and return a comma-separated list of names as a single String. Hint: use Collectors.joining(", ") for this.
    e. Print the resulting String.
3. Create a new class with a main method. Create a method (method a) that accepts an Optional of some type of object (Animal, Person, Camera, etc.).
    a. The method should return the object unwrapped from the Optional if the object is present. For example, if you have an object of type Cheese, your method signature should look something like this:

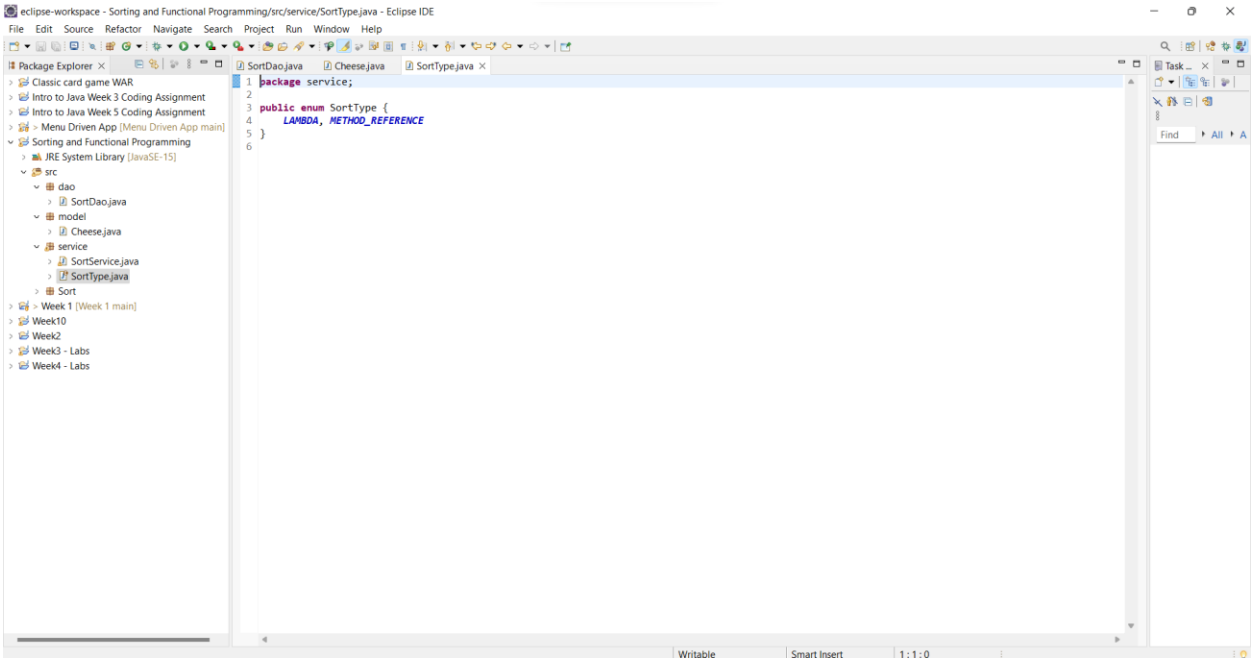
        ```
        public Cheese cheesyMethod(Optional<Cheese> optionalCheese) {...}
        ```

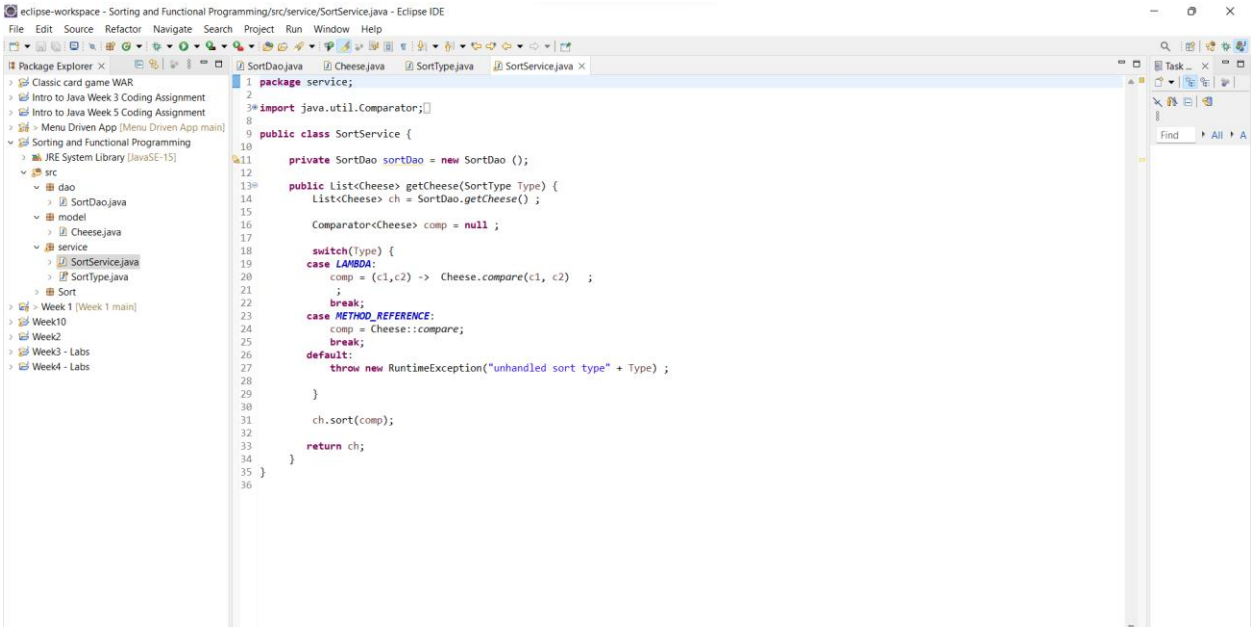
    b. The method should throw a NoSuchElementException with a custom message if the object is not present.
    c. Create another method (method b) that calls method a with an object wrapped by an Optional. Show that the object is returned unwrapped from the Optional (i.e., print the object).
    d. Method b should also call method a with an empty Optional. Show that a NoSuchElementException is thrown by method a by printing the exception message. Hint: catch the `NoSuchElementException` as parameter named "e" and do `System.out.println(e.getMessage()).`
    e. Note: your method should handle the Optional as shown in the video on Optionals using the orElseThrow method. For the missing object, you must use a Lambda expression in orElseThrow to return a NoSuchElementException with a custom message.

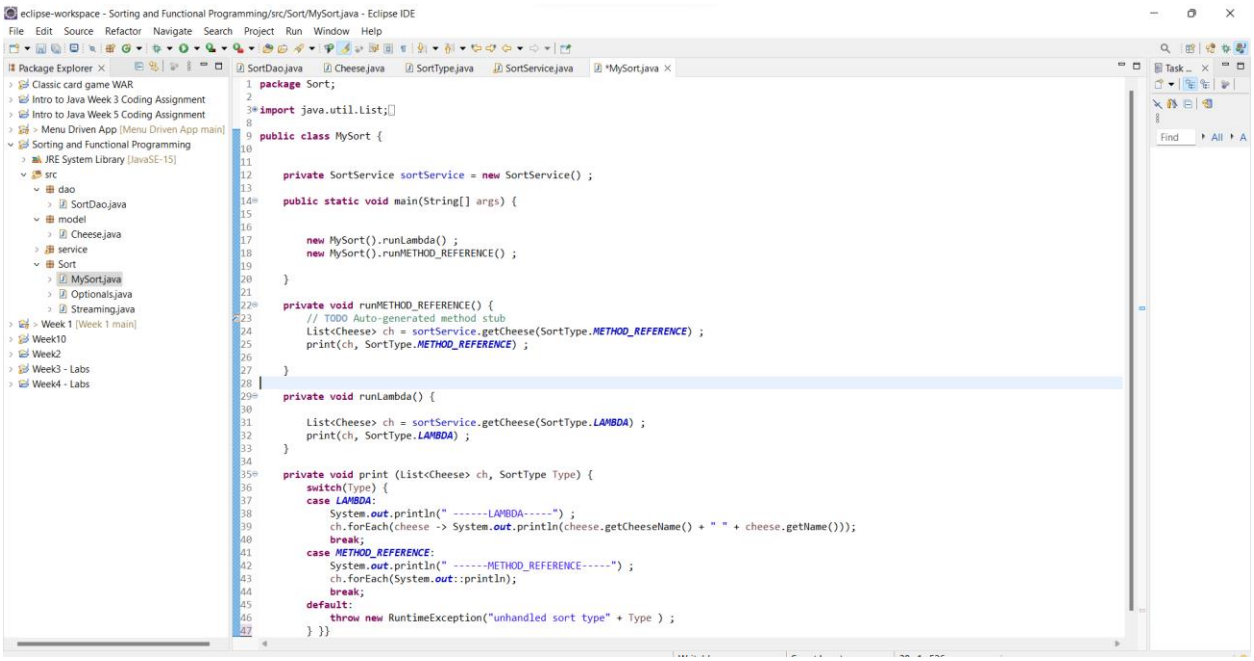
Screenshots of Code:

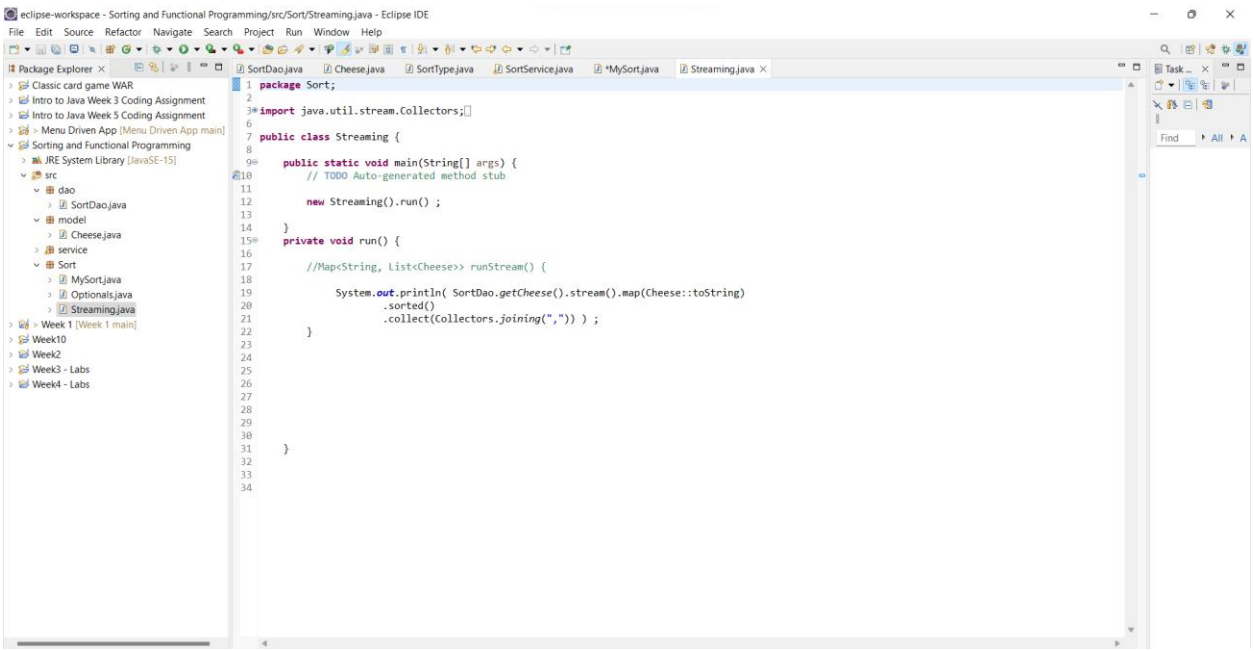

eclipse-workspace - Sorting and Functional Programming/src/dao/SortDao.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
> Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
SortDao.java
model
service
Sort
> Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
package dao;
import java.util.ArrayList;
public class SortDao {
static List<Cheese> ch = new ArrayList<>(List.of(new Cheese("Feta"), new Cheese("Mozzarella"),
new Cheese("Emmental"), new Cheese("Cotija"),
new Cheese("Chevre"), new Cheese("Camembert"))) ;
public static List<Cheese> getCheese() {
return ch ;
}
}
Writable
Smart Insert
1 : 1 : 0
Show desktop

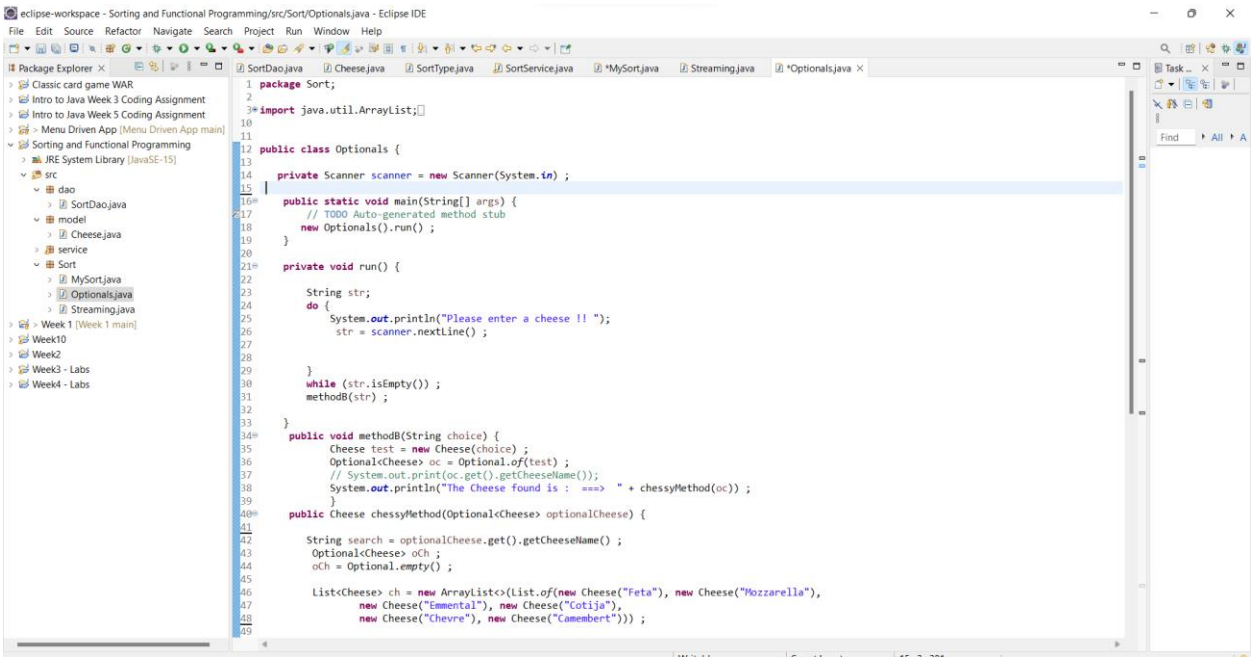

eclipse-workspace - Sorting and Functional Programming/src/model/Cheese.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
> Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
model
Cheese.java
service
Sort
> Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
Cheese.java
package model;
public class Cheese {
private String name = "cheese" ;
private String cheeseType ;
public Cheese(String name) {
this.cheeseType = name ;
}
public String getCheeseName() {
return cheeseType;
}
public void setCheeseName(String cheeseName) {
this.cheeseType = cheeseName;
}
@Override
public String toString() {
return cheeseType + " " + name ;
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public static int compare(Cheese c1, Cheese c2) {
return c1.cheeseType.compareTo(c2.cheeseType);
}
}

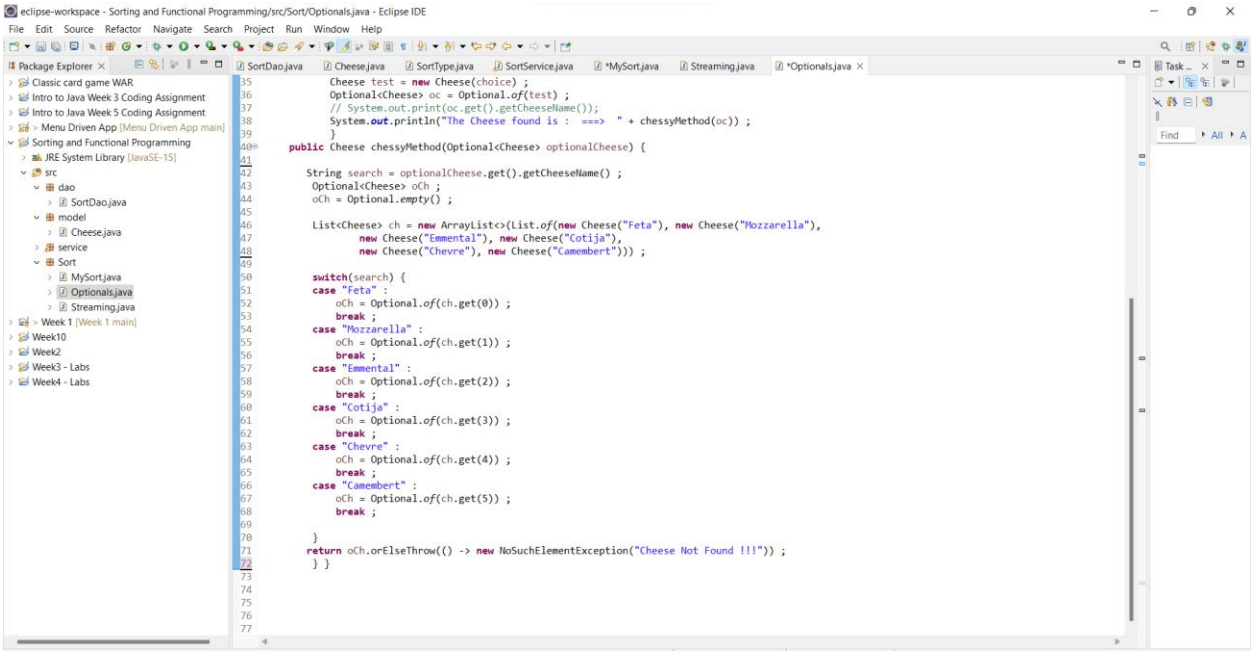

eclipse-workspace - Sorting and Functional Programming/src/service/SortType.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
> Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
SortDao.java
model
Cheese.java
service
SortService.java
SortType.java
Sort
> Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
Cheese.java
SortType.java
1 package service;
2
3 public enum SortType {
4 LAMBDA, METHOD_REFERENCE
5 }
6
Writable
Smart Insert
1 : 1 : 0


eclipse-workspace - Sorting and Functional Programming/src/service/SortService.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
> Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
SortDao.java
model
Cheese.java
service
SortService.java
SortType.java
Sort
> Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
Cheese.java
SortType.java
SortService.java
1 package service;
2
3 import java.util.Comparator;
8
9 public class SortService {
10
11 private SortDao sortDao = new SortDao ();
12
13 public List<Cheese> getCheese(SortType Type) {
14 List<Cheese> ch = SortDao.getCheese() ;
15
16 Comparator<Cheese> comp = null ;
17
18 switch(Type) {
19 case LAMBDA:
20 comp = (c1,c2) -> Cheese.compare(c1, c2) ;
21 ;
22 break;
23 case METHOD_REFERENCE:
24 comp = Cheese::compare;
25 break;
26 default:
27 throw new RuntimeException("unhandled sort type" + Type) ;
28
29 }
30
31 ch.sort(comp);
32
33 return ch;
34 }
35 }
36

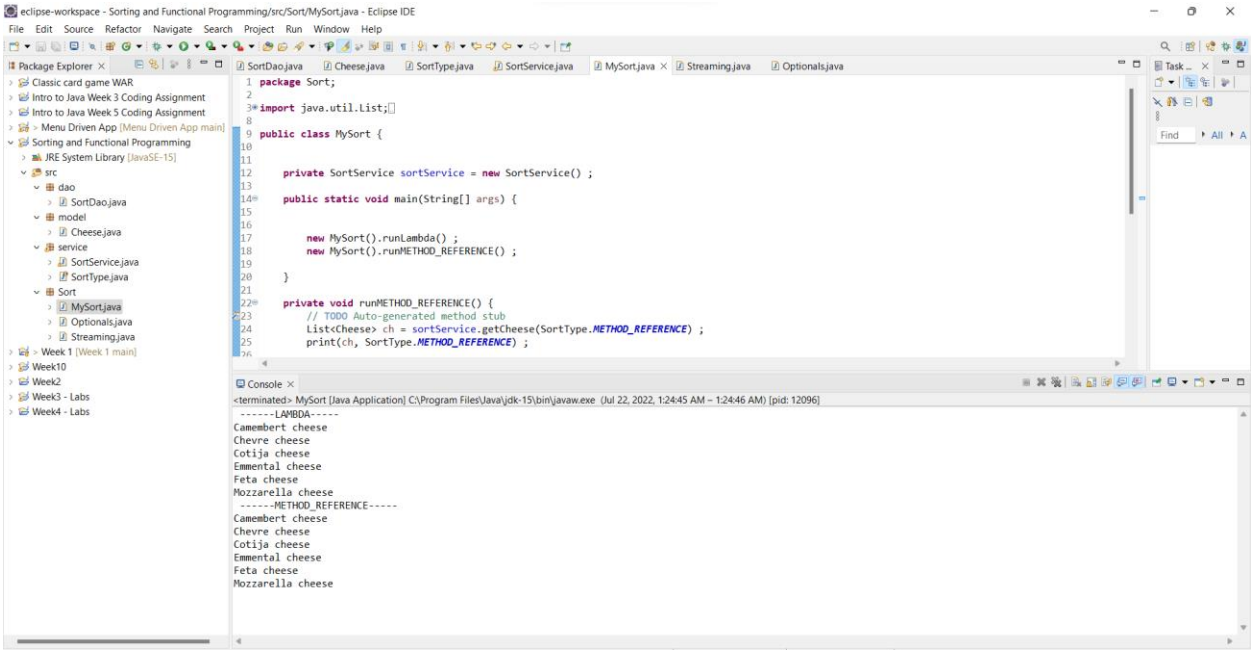

eclipse-workspace - Sorting and Functional Programming/src/Sort/MySort.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
SortDao.java
model
Cheese.java
service
Sort
MySort.java
Optionals.java
Streaming.java
Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
Cheese.java
SortType.java
SortService.java
*MySort.java
1 package Sort;
2
3 import java.util.List;
8
9 public class MySort {
10
11
12 private SortService sortService = new SortService() ;
13
14 public static void main(String[] args) {
15
16
17 new MySort().runLambda() ;
18 new MySort().runMETHOD_REFERENCE() ;
19
20 }
21
22 private void runMETHOD_REFERENCE() {
23 // TODO Auto-generated method stub
24 List<Cheese> ch = sortService.getCheese(SortType.METHOD_REFERENCE) ;
25 print(ch, SortType.METHOD_REFERENCE) ;
26
27 }
28
29 private void runLambda() {
30
31 List<Cheese> ch = sortService.getCheese(SortType.LAMBDA) ;
32 print(ch, SortType.LAMBDA) ;
33 }
34
35 private void print (List<Cheese> ch, SortType Type) {
36 switch(Type) {
37 case LAMBDA:
38 System.out.println(" ------LAMBDA-----") ;
39 ch.forEach(cheese -> System.out.println(cheese.getCheeseName() + " " + cheese.getName()));
40 break;
41 case METHOD_REFERENCE:
42 System.out.println(" ------METHOD_REFERENCE-----") ;
43 ch.forEach(System.out::println);
44 break;
45 default:
46 throw new RuntimeException("unhandled sort type" + Type) ;
47 } }
Task ...
Find
All
A

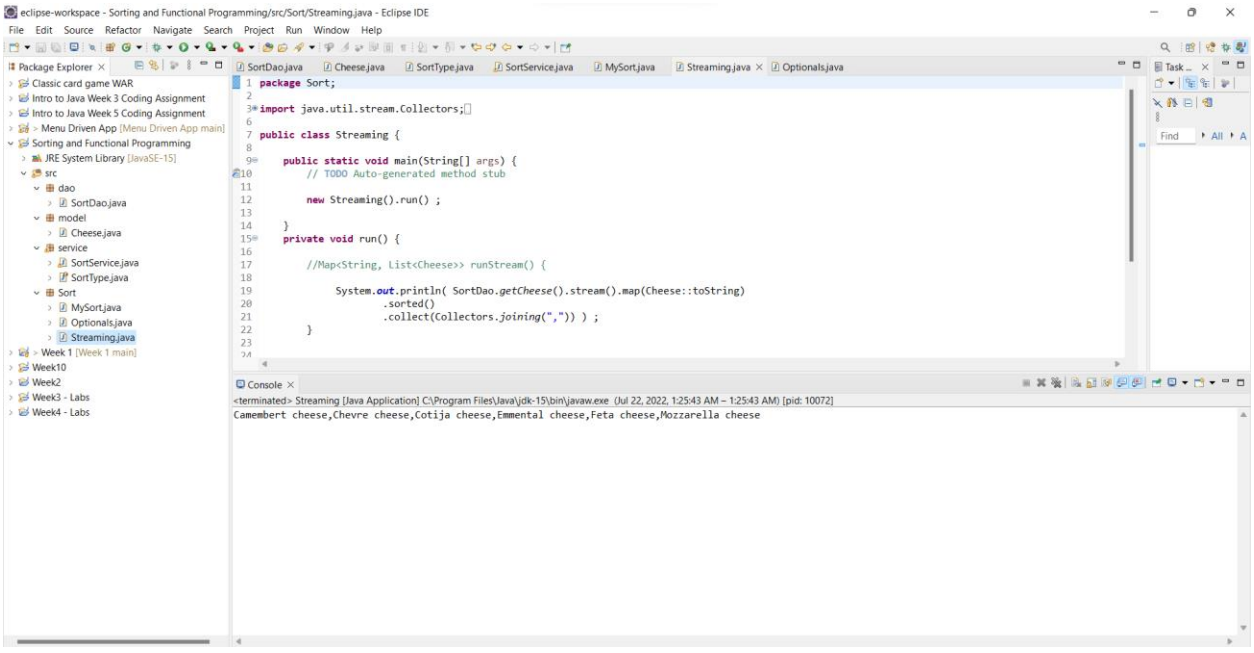

eclipse-workspace - Sorting and Functional Programming/src/Sort/Streaming.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
SortDao.java
model
Cheese.java
service
Sort
MySort.java
Optionals.java
Streaming.java
Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
Cheese.java
SortType.java
SortService.java
*MySort.java
Streaming.java
1 package Sort;
2
3 import java.util.stream.Collectors;
6
7 public class Streaming {
8
9 public static void main(String[] args) {
10 // TODO Auto-generated method stub
11
12 new Streaming().run() ;
13
14 }
15 private void run() {
16
17 //Map<String, List<Cheese>> runStream() {
18
19 System.out.println(SortDao.getCheese().stream().map(Cheese::toString)
20 .sorted()
21 .collect(Collectors.joining(","))) ;
22 }
31 }
Task ...
Find
All
A

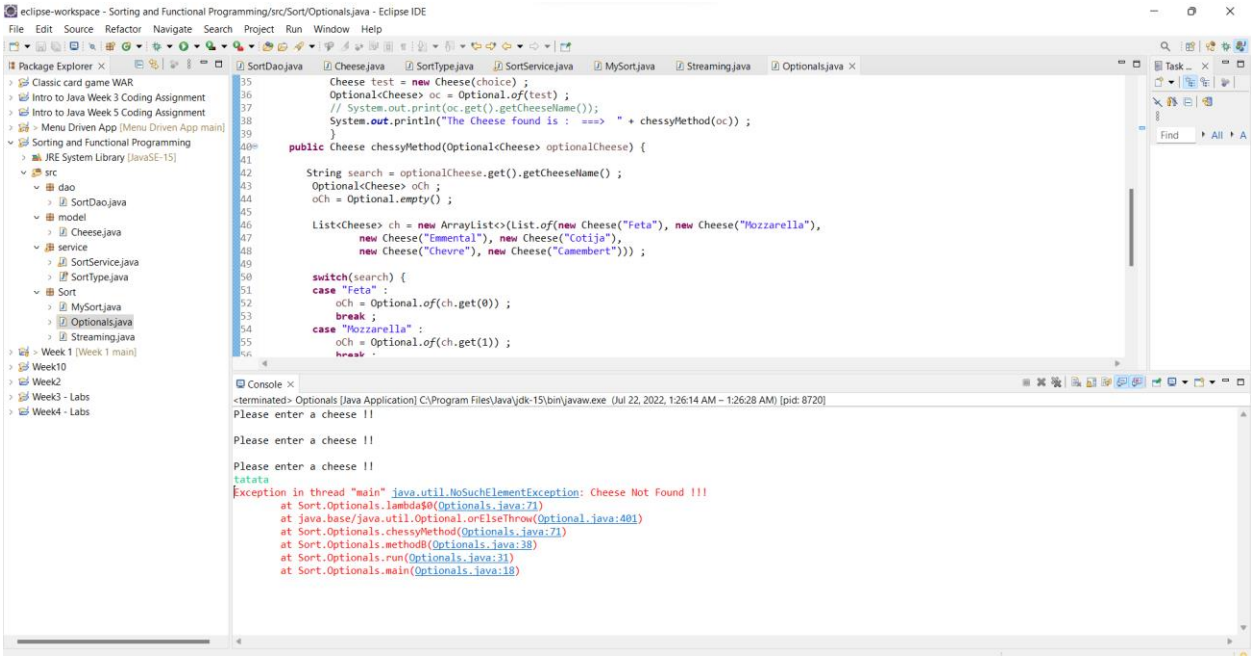


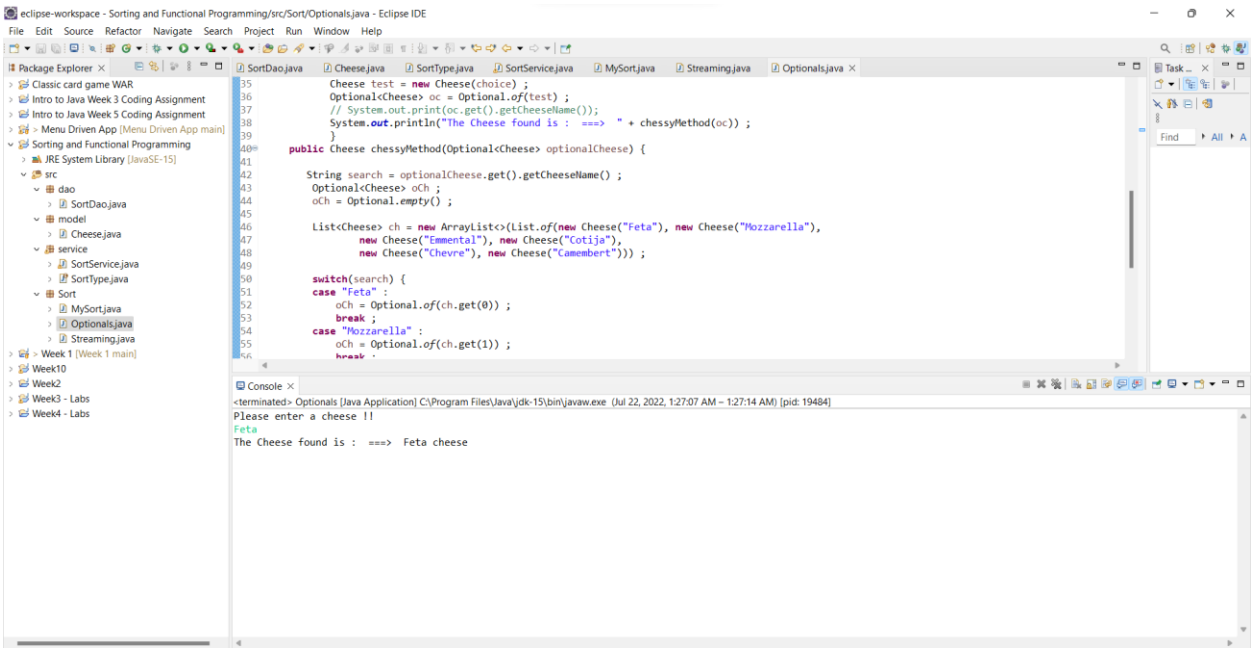


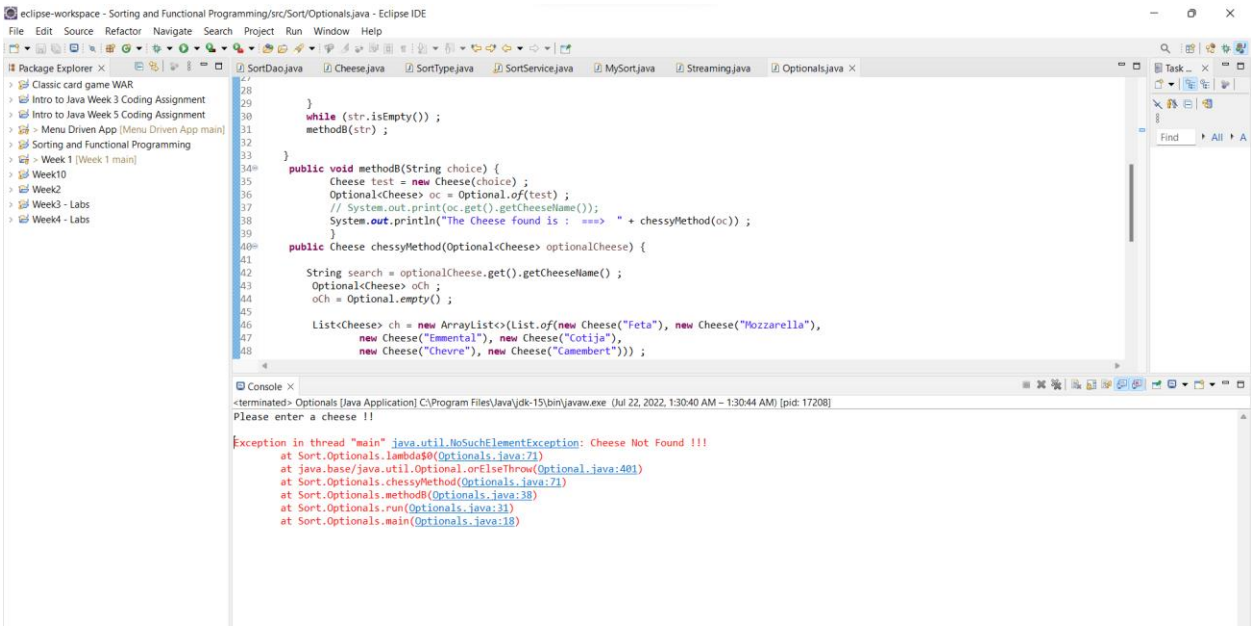
Screenshots of Running Application Results:


eclipse-workspace - Sorting and Functional Programming/src/Sort/MySort.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
SortDao.java
model
Cheese.java
service
SortService.java
SortType.java
Sort
MySort.java
Optionals.java
Streaming.java
Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
Cheese.java
SortType.java
SortService.java
MySort.java
Streaming.java
Optionals.java
package Sort;
import java.util.List;
public class MySort {
private SortService sortService = new SortService() ;
public static void main(String[] args) {
new MySort().runLambda() ;
new MySort().runMETHOD_REFERENCE() ;
}
private void runMETHOD_REFERENCE() {
// TODO Auto-generated method stub
List<Cheese> ch = sortService.getCheese(SortType.METHOD_REFERENCE) ;
print(ch, SortType.METHOD_REFERENCE) ;
Console
<terminated> MySort [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Jul 22, 2022, 1:24:45 AM – 1:24:46 AM) [pid: 12096]
------LAMBDA-----
Camembert cheese
Chevre cheese
Cotija cheese
Emmental cheese
Feta cheese
Mozzarella cheese
------METHOD_REFERENCE-----
Camembert cheese
Chevre cheese
Cotija cheese
Emmental cheese
Feta cheese
Mozzarella cheese


eclipse-workspace - Sorting and Functional Programming/src/Sort/Streaming.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
SortDao.java
model
Cheese.java
service
SortService.java
SortType.java
Sort
MySort.java
Optionals.java
Streaming.java
Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
Cheese.java
SortType.java
SortService.java
MySort.java
Streaming.java
Optionals.java
package Sort;
import java.util.stream.Collectors;
public class Streaming {
public static void main(String[] args) {
// TODO Auto-generated method stub
new Streaming().run() ;
}
private void run() {
//Map<String, List<Cheese>> runStream() {
System.out.println(SortDao.getCheese().stream().map(Cheese::toString)
.sorted()
.collect(Collectors.joining(","))) ;
}
Console
<terminated> Streaming [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Jul 22, 2022, 1:25:43 AM – 1:25:43 AM) [pid: 10072]
Camembert cheese,Chevre cheese,Cotija cheese,Emmental cheese,Feta cheese,Mozzarella cheese


eclipse-workspace - Sorting and Functional Programming/src/Sort/Optionals.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
SortDao.java
model
Cheese.java
service
SortService.java
SortType.java
Sort
MySort.java
Optionals.java
Streaming.java
Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
Cheese.java
SortType.java
SortService.java
MySort.java
Streaming.java
Optionals.java
Cheese test = new Cheese(choice) ;
Optional<Cheese> oc = Optional.of(test) ;
// System.out.print(oc.get().getCheeseName());
System.out.println("The Cheese found is : ===> " + chessyMethod(oc)) ;
}
public Cheese chessyMethod(Optional<Cheese> optionalCheese) {
String search = optionalCheese.get().getCheeseName() ;
Optional<Cheese> oCh ;
oCh = Optional.empty() ;
List<Cheese> ch = new ArrayList<>(List.of(new Cheese("Feta"), new Cheese("Mozzarella"),
new Cheese("Emmental"), new Cheese("Cotija"),
new Cheese("Chevre"), new Cheese("Camembert"))) ;
switch(search) {
case "Feta" :
oCh = Optional.of(ch.get(0)) ;
break ;
case "Mozzarella" :
oCh = Optional.of(ch.get(1)) ;
Console
<terminated> Optionals [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Jul 22, 2022, 1:26:14 AM – 1:26:28 AM) [pid: 8720]
Please enter a cheese !!
Please enter a cheese !!
Please enter a cheese !!
tatata
Exception in thread "main" java.util.NoSuchElementException: Cheese Not Found !!!
at Sort.Optionals.lambda$0(Optionals.java:71)
at java.base/java.util.Optional.orElseThrow(Optional.java:401)
at Sort.Optionals.chessyMethod(Optionals.java:71)
at Sort.Optionals.methodB(Optionals.java:38)
at Sort.Optionals.run(Optionals.java:31)
at Sort.Optionals.main(Optionals.java:18)


eclipse-workspace - Sorting and Functional Programming/src/Sort/Optionals.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Classic card game WAR
Intro to Java Week 3 Coding Assignment
Intro to Java Week 5 Coding Assignment
Menu Driven App [Menu Driven App main]
Sorting and Functional Programming
JRE System Library [JavaSE-15]
src
dao
SortDao.java
model
Cheese.java
service
SortService.java
SortType.java
Sort
MySort.java
Optionals.java
Streaming.java
Week 1 [Week 1 main]
Week10
Week2
Week3 - Labs
Week4 - Labs
SortDao.java
Cheese.java
SortType.java
SortService.java
MySort.java
Streaming.java
Optionals.java
Cheese test = new Cheese(choice) ;
Optional<Cheese> oc = Optional.of(test) ;
// System.out.print(oc.get().getCheeseName());
System.out.println("The Cheese found is : ===> " + chessyMethod(oc)) ;
}
public Cheese chessyMethod(Optional<Cheese> optionalCheese) {
String search = optionalCheese.get().getCheeseName() ;
Optional<Cheese> oCh ;
oCh = Optional.empty() ;
List<Cheese> ch = new ArrayList<>(List.of(new Cheese("Feta"), new Cheese("Mozzarella"),
new Cheese("Emmental"), new Cheese("Cotija"),
new Cheese("Chevre"), new Cheese("Camembert"))) ;
switch(search) {
case "Feta" :
oCh = Optional.of(ch.get(0)) ;
break ;
case "Mozzarella" :
oCh = Optional.of(ch.get(1)) ;
Console
<terminated> Optionals [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Jul 22, 2022, 1:27:07 AM – 1:27:14 AM) [pid: 19484]
Please enter a cheese !!
Feta
The Cheese found is : ===> Feta cheese



URL to GitHub Repository:

https://github.com/cel90/Week-11-Sorting-and-Functional-Programming