

Relational Databases with MySQL Week 4 Coding 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 steps 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:

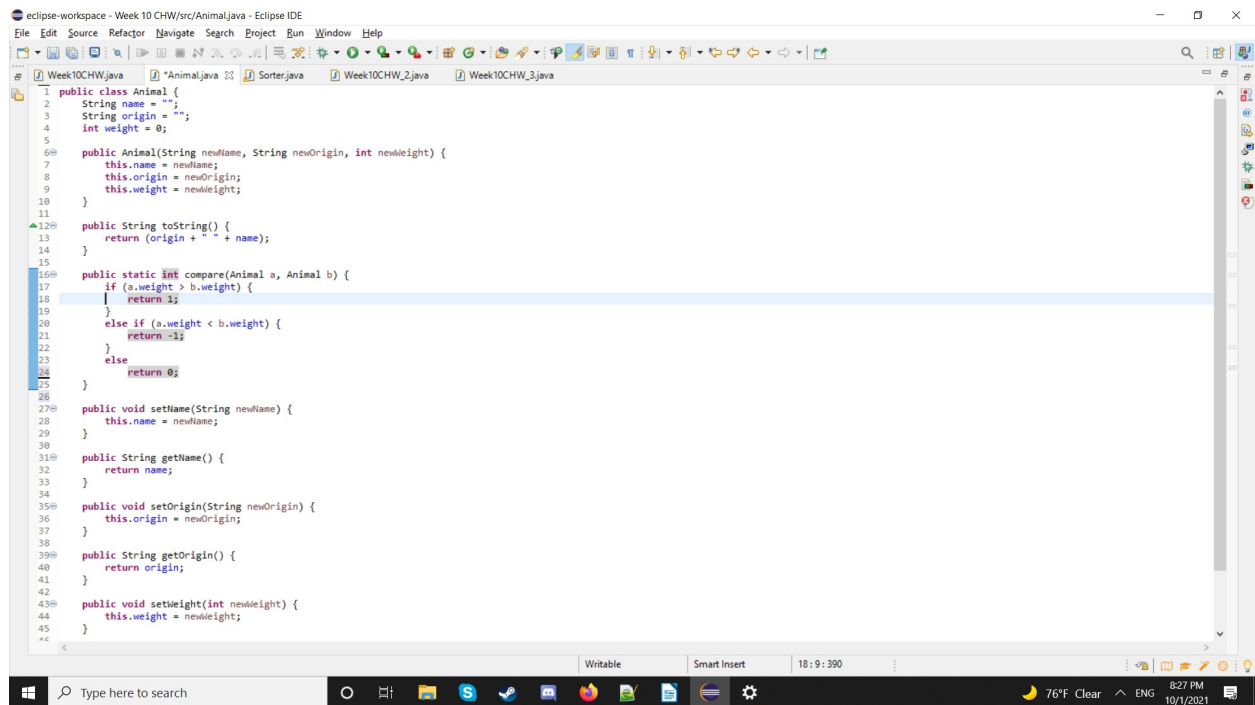
1. Create a class of whatever type you want (Animal, Person, Camera, Cheese, etc.).
 - 1.a. Do not implement the Comparable interface.
 - 1.b. Add a name instance variable so that you can tell the objects apart.
 - 1.c. Add getters, setters and/or a constructor as appropriate.
 - 1.d. Add a toString method that returns the name and object type (like "Pentax Camera").
 - 1.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".
 - 1.f. Create a static list of these objects, adding at least 4 objects to the list.
 - 1.g. In another class, write a method to sort the objects using a Lambda expression using the compare method you created earlier.
 - 1.h. Write a method to sort the objects using a Method Reference to the compare method you created earlier.
 - 1.i. Create a main method to call the sort methods.
 - 1.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.
 - 2.a. Create a Stream from the list of objects.
 - 2.b. Turn the Stream of object to a Stream of String (use the map method for this).
 - 2.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.)
 - 2.d. Collect the Stream and return a comma-separated list of names as a single String. Hint: use `Collectors.joining(", ")` for this.
 - 2.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.).
 - 3.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) {...}
```
 - 3.b. The method should throw a `NoSuchElementException` with a custom message if the object is not present.
 - 3.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).
 - 3.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())`.
 - 3.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.

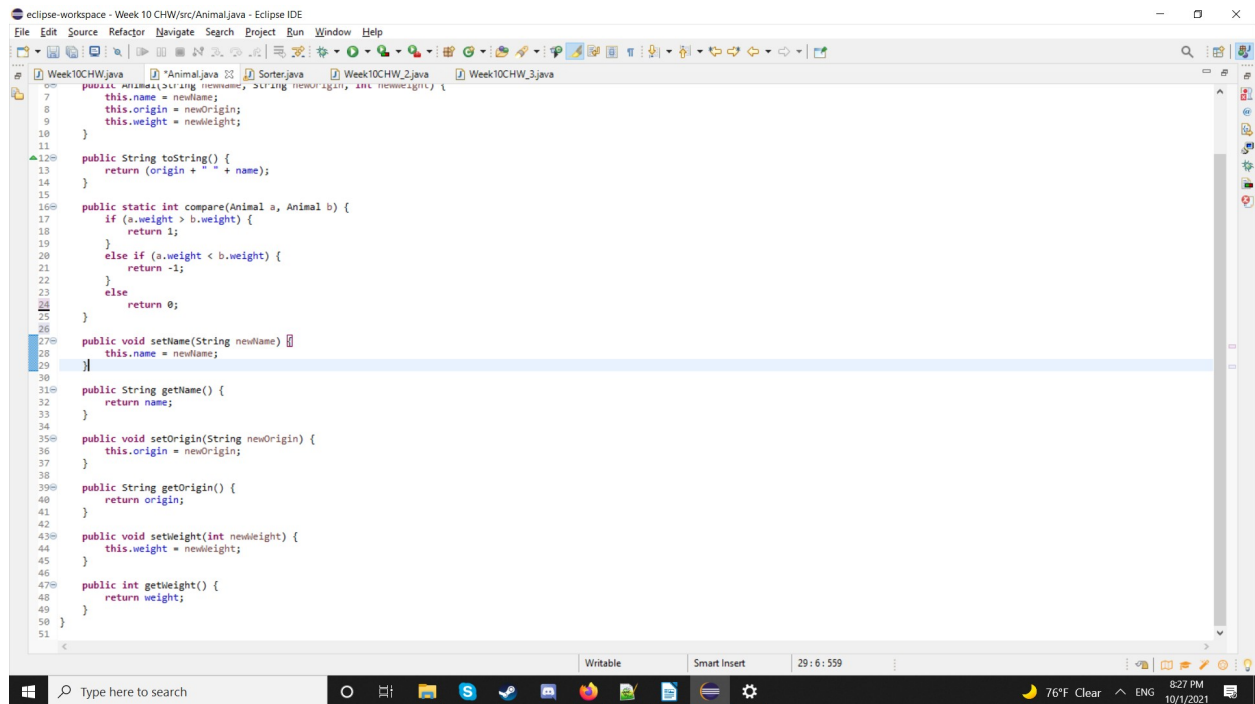
Screenshots:

Animal.java



This screenshot shows the Eclipse IDE with the file 'Animal.java' open. The code defines an 'Animal' class with attributes 'name', 'origin', and 'weight'. It includes methods for setting and getting these attributes, a 'toString()' method, and a static 'compare()' method for sorting animals by weight. The IDE interface includes a menu bar, a toolbar, and a project explorer on the left.

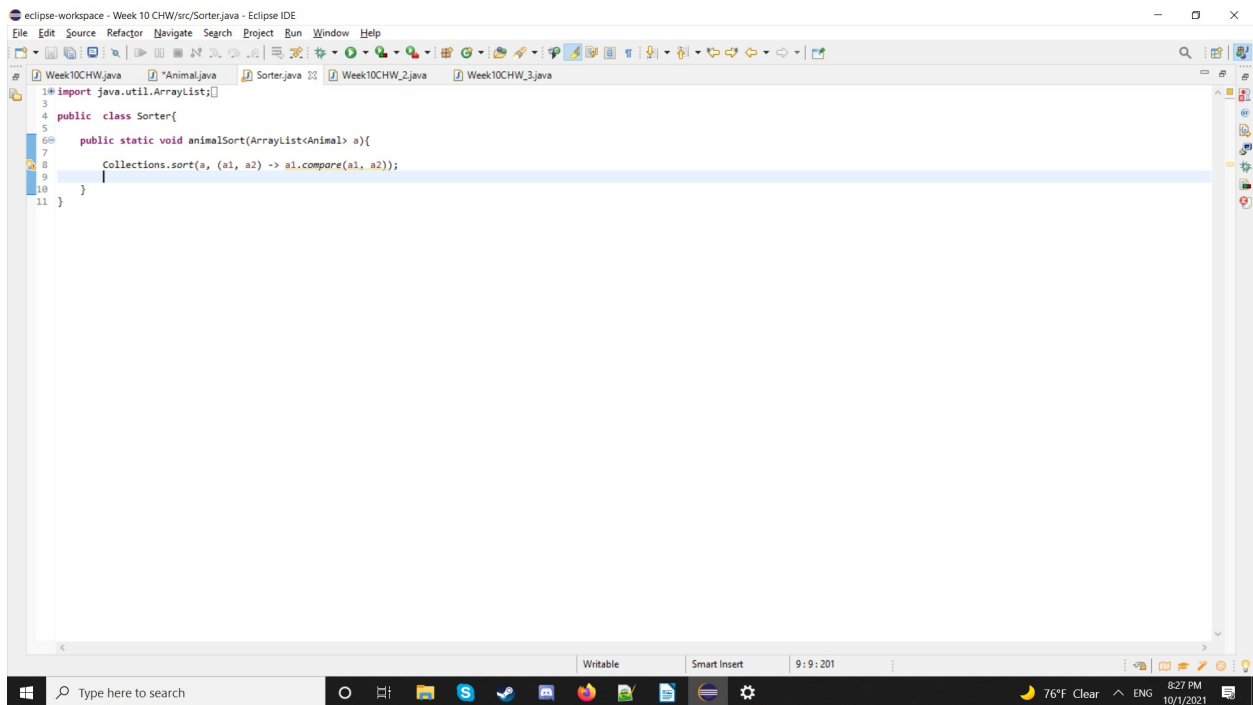
```
1 public class Animal {
2     String name = "";
3     String origin = "";
4     int weight = 0;
5
6     public Animal(String newName, String newOrigin, int newWeight) {
7         this.name = newName;
8         this.origin = newOrigin;
9         this.weight = newWeight;
10    }
11
12    public String toString() {
13        return (origin + " " + name);
14    }
15
16    public static int compare(Animal a, Animal b) {
17        if (a.weight > b.weight) {
18            return 1;
19        }
20        else if (a.weight < b.weight) {
21            return -1;
22        }
23        else
24            return 0;
25    }
26
27    public void setName(String newName) {
28        this.name = newName;
29    }
30
31    public String getName() {
32        return name;
33    }
34
35    public void setOrigin(String newOrigin) {
36        this.origin = newOrigin;
37    }
38
39    public String getOrigin() {
40        return origin;
41    }
42
43    public void setWeight(int newWeight) {
44        this.weight = newWeight;
45    }
46 }
```



This screenshot shows the same Eclipse IDE with 'Animal.java', but with an additional method 'getWeight()' added to the class. The code is identical to the previous screenshot, except for the new method at the bottom. The IDE interface remains the same.

```
1 public class Animal {
2     String name = "";
3     String origin = "";
4     int weight = 0;
5
6     public Animal(String newName, String newOrigin, int newWeight) {
7         this.name = newName;
8         this.origin = newOrigin;
9         this.weight = newWeight;
10    }
11
12    public String toString() {
13        return (origin + " " + name);
14    }
15
16    public static int compare(Animal a, Animal b) {
17        if (a.weight > b.weight) {
18            return 1;
19        }
20        else if (a.weight < b.weight) {
21            return -1;
22        }
23        else
24            return 0;
25    }
26
27    public void setName(String newName) {
28        this.name = newName;
29    }
30
31    public String getName() {
32        return name;
33    }
34
35    public void setOrigin(String newOrigin) {
36        this.origin = newOrigin;
37    }
38
39    public String getOrigin() {
40        return origin;
41    }
42
43    public void setWeight(int newWeight) {
44        this.weight = newWeight;
45    }
46
47    public int getWeight() {
48        return weight;
49    }
50 }
51 }
```

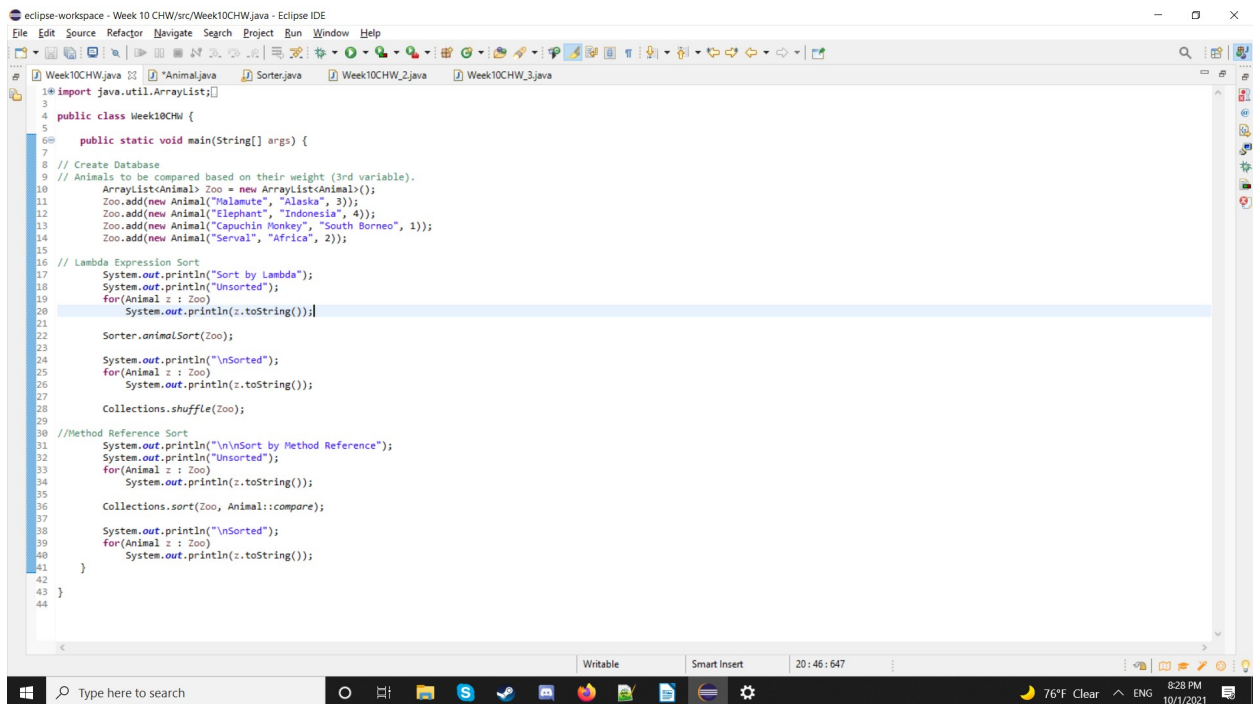
Sorter.java



The screenshot shows the Eclipse IDE with the file 'Sorter.java' open. The code defines a class 'Sorter' with a static method 'animalSort' that takes an 'ArrayList<Animal>' and sorts it using 'Collections.sort' with a lambda expression for comparison.

```
1 import java.util.ArrayList;
2
3 public class Sorter{
4
5     public static void animalSort(ArrayList<Animal> a){
6
7         Collections.sort(a, (a1, a2) -> a1.compare(a1, a2));
8     }
9 }
10
11 }
```

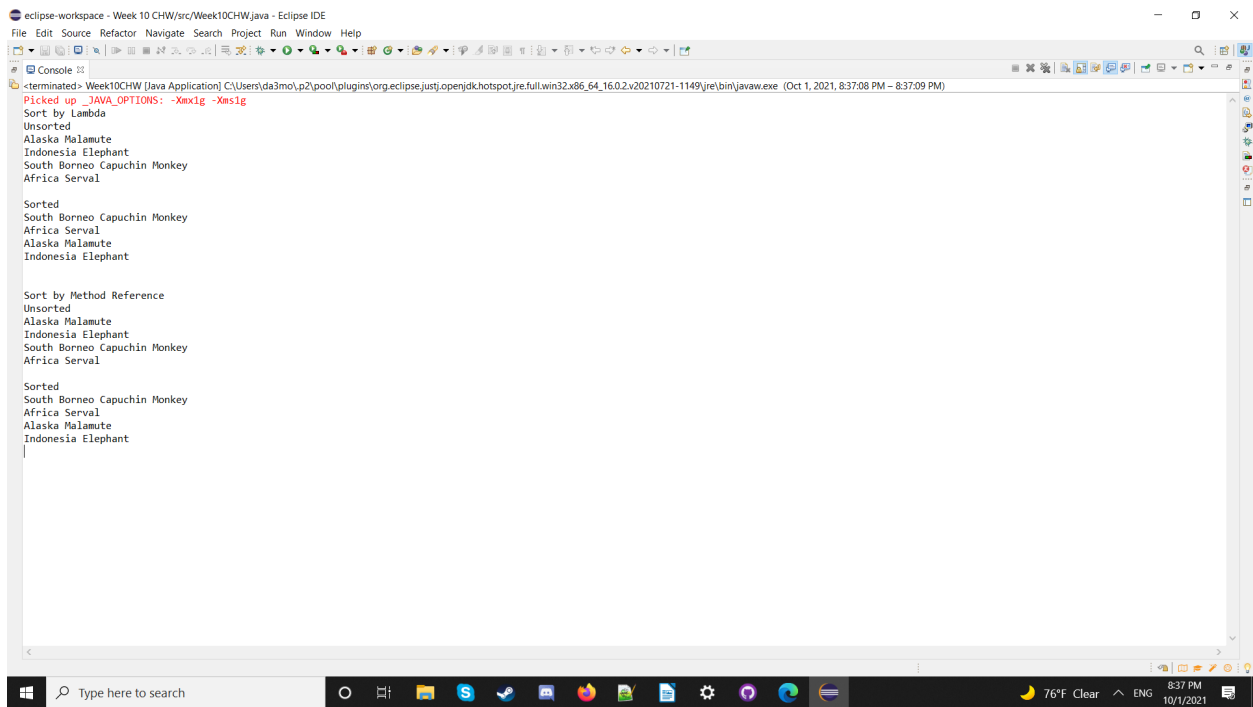
Week10CHW_1 (Coding Assignment # 1)



The screenshot shows the Eclipse IDE with the file 'Week10CHW_1.java' open. The code defines a class 'Week10CHW_1' with a 'main' method. It creates a 'Zoo' (ArrayList<Animal>) and adds several animals. It then demonstrates two sorting methods: 'Lambda Expression Sort' and 'Method Reference Sort', both using the 'Sorter' class's 'animalSort' method. The code also includes 'System.out.println' statements to display the state of the zoo before and after sorting.

```
1 import java.util.ArrayList;
2
3 public class Week10CHW_1 {
4
5     public static void main(String[] args) {
6
7         // Create Database
8         // Animals to be compared based on their weight (3rd variable).
9         ArrayList<Animal> Zoo = new ArrayList<Animal>();
10         Zoo.add(new Animal("Malamute", "Alaska", 3));
11         Zoo.add(new Animal("Elephant", "Indonesia", 4));
12         Zoo.add(new Animal("Capuchin Monkey", "South Borneo", 1));
13         Zoo.add(new Animal("Serval", "Africa", 2));
14
15         // Lambda Expression Sort
16         System.out.println("Sort by Lambda");
17         System.out.println("Unsorted");
18         for (Animal z : Zoo)
19             System.out.println(z.toString());
20
21         Sorter.animalSort(Zoo);
22
23         System.out.println("\nSorted");
24         for (Animal z : Zoo)
25             System.out.println(z.toString());
26
27         Collections.shuffle(Zoo);
28
29         //Method Reference Sort
30         System.out.println("\n\nSort by Method Reference");
31         System.out.println("Unsorted");
32         for (Animal z : Zoo)
33             System.out.println(z.toString());
34
35         Collections.sort(Zoo, Animal::compare);
36
37         System.out.println("\nSorted");
38         for (Animal z : Zoo)
39             System.out.println(z.toString());
40     }
41 }
42
43 }
```

#1 Results



The screenshot shows the Eclipse IDE interface with the console window open. The console output displays the results of a Java application execution. The output is as follows:

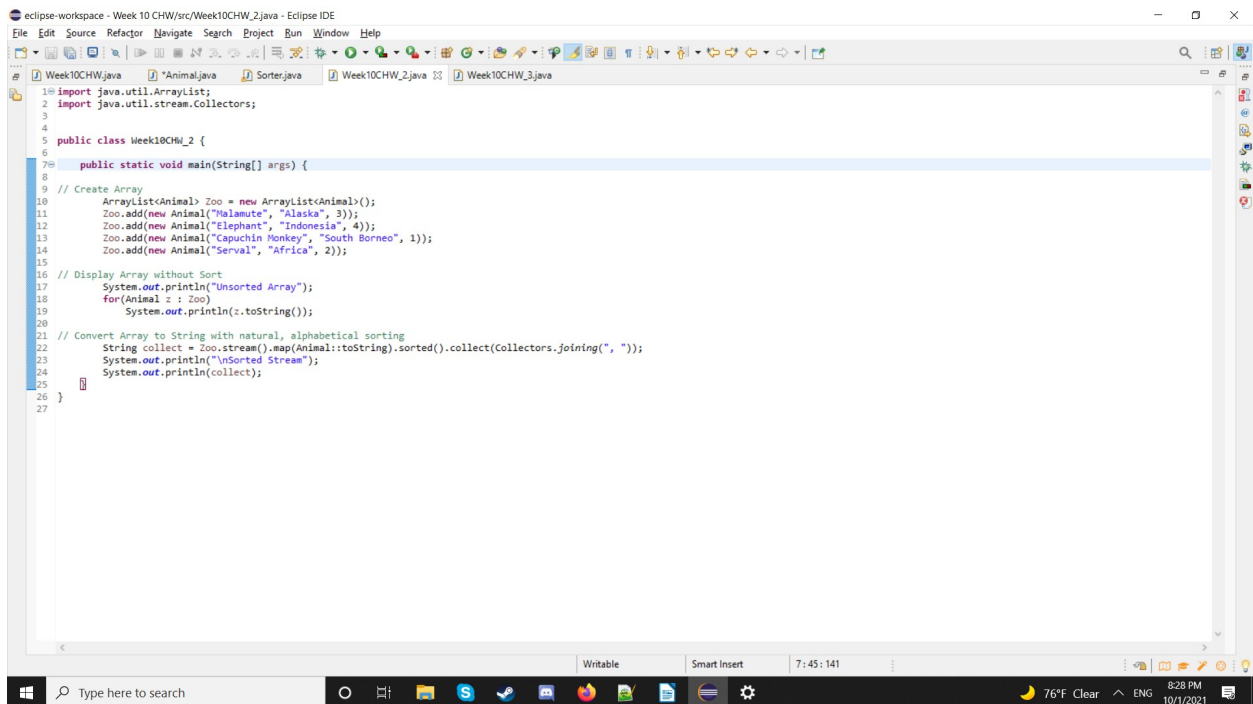
```
<terminated> Week10CHW [Java Application] C:\Users\da3mo\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (Oct 1, 2021, 8:37:08 PM - 8:37:09 PM)
Picked up _JAVA_OPTIONS: -Xmx1g -Xms1g
Sort by Lambda
Unsorted
Alaska Malamute
Indonesia Elephant
South Borneo Capuchin Monkey
Africa Serval

Sorted
South Borneo Capuchin Monkey
Africa Serval
Alaska Malamute
Indonesia Elephant

Sort by Method Reference
Unsorted
Alaska Malamute
Indonesia Elephant
South Borneo Capuchin Monkey
Africa Serval

Sorted
South Borneo Capuchin Monkey
Africa Serval
Alaska Malamute
Indonesia Elephant
```

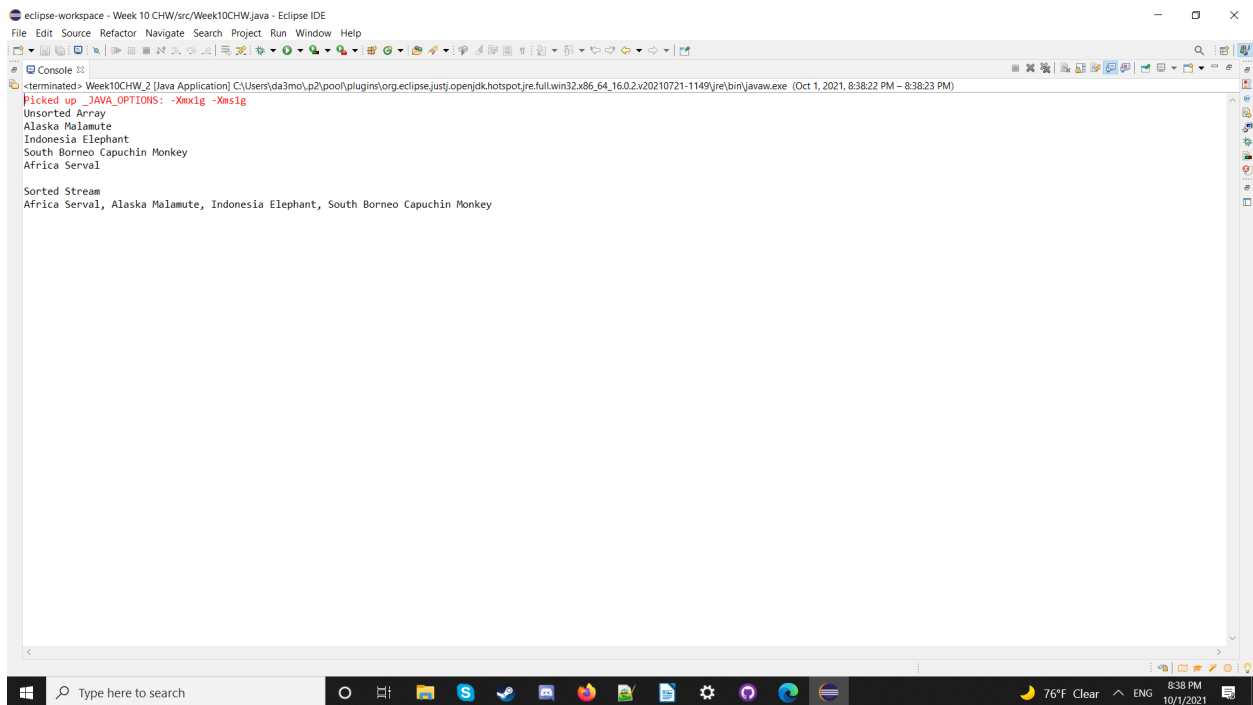
Week10CHW_2 (Coding Assignment #2)



The screenshot shows the Eclipse IDE interface with the Week10CHW_2.java file open. The code is as follows:

```
1 import java.util.ArrayList;
2 import java.util.stream.Collectors;
3
4
5 public class Week10CHW_2 {
6
7     public static void main(String[] args) {
8
9         // Create Array
10         ArrayList<Animal> Zoo = new ArrayList<Animal>();
11         Zoo.add(new Animal("Malamute", "Alaska", 3));
12         Zoo.add(new Animal("Elephant", "Indonesia", 4));
13         Zoo.add(new Animal("Capuchin Monkey", "South Borneo", 1));
14         Zoo.add(new Animal("Serval", "Africa", 2));
15
16         // Display Array without Sort
17         System.out.println("Unsorted Array");
18         for(Animal z : Zoo)
19             System.out.println(z.toString());
20
21         // Convert Array to String with natural, alphabetical sorting
22         String collect = Zoo.stream().map(Animal::toString).sorted().collect(Collectors.joining(", "));
23         System.out.println("Unsorted Stream");
24         System.out.println(collect);
25     }
26 }
27
```

#2 Results

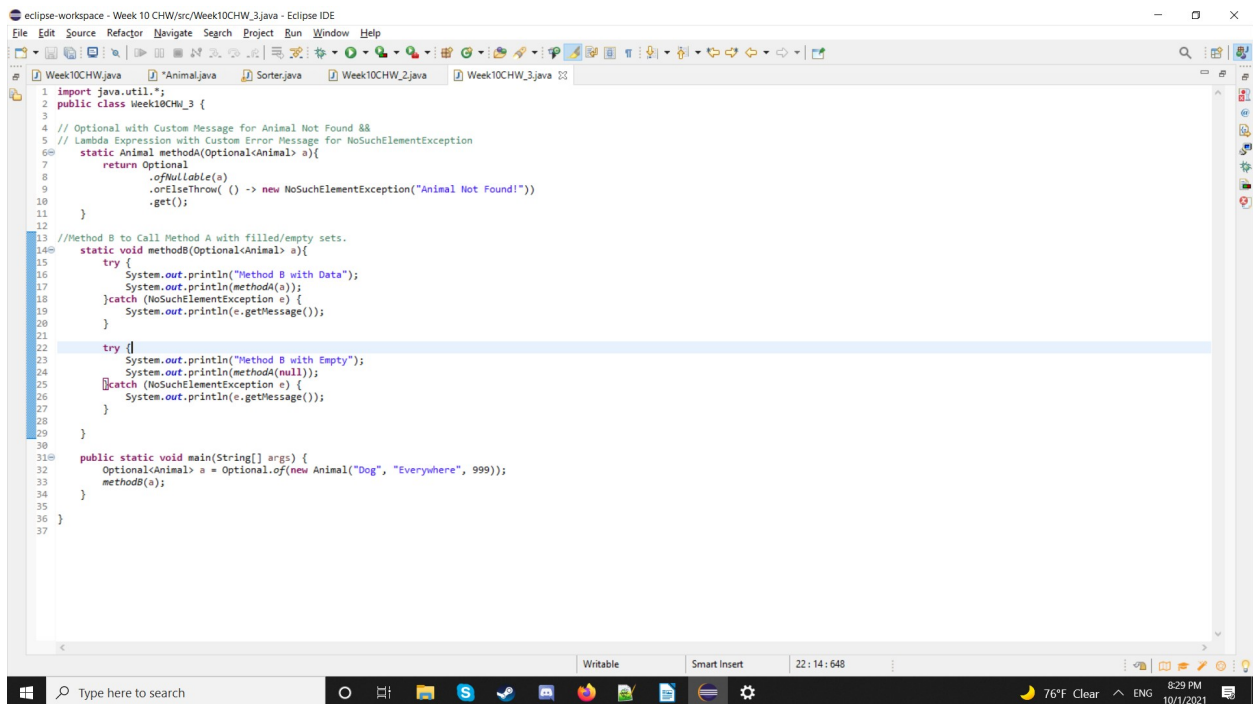


```
eclipse-workspace - Week 10 CHW/src/Week10CHW.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

<terminated> Week10CHW_2 [Java Application] C:\Users\da3mo\p2\pool\plugins\org.eclipse.justi.openjdk hotspot.jre.full\win32\x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (Oct 1, 2021, 8:38:22 PM - 8:38:23 PM)
Picked up _JAVA_OPTIONS: -Xmx1g -Xms1g
Unsorted Array
Alaska Malamute
Indonesia Elephant
South Borneo Capuchin Monkey
Africa Serval

Sorted Stream
Africa Serval, Alaska Malamute, Indonesia Elephant, South Borneo Capuchin Monkey
```

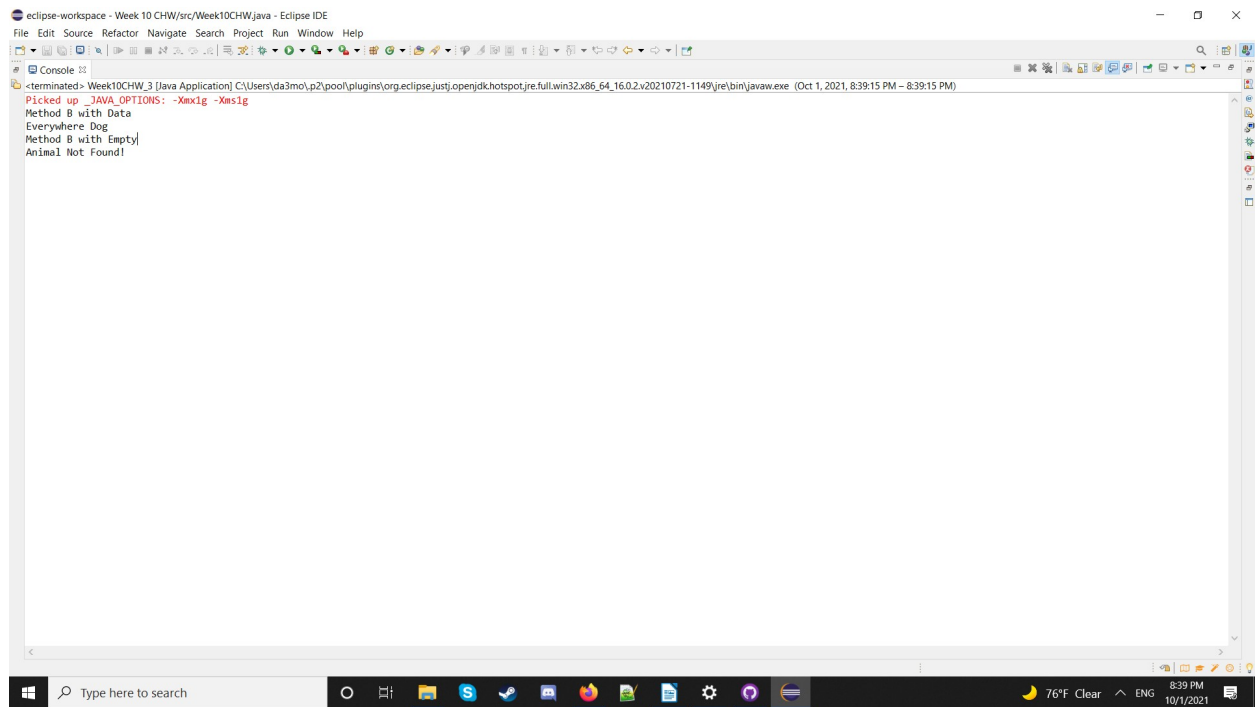
Week10CHW_3 (Coding Assignment #3)



```
eclipse-workspace - Week 10 CHW/src/Week10CHW_3.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Week10CHW.java *Animal.java Sorter.java Week10CHW_2.java Week10CHW_3.java
1 import java.util.*;
2 public class Week10CHW_3 {
3
4 // Optional with Custom Message for Animal Not Found &&
5 // Lambda Expression with Custom Error Message for NoSuchElementException
6 static Animal methodA(Optional<Animal> a){
7     return Optional
8         .ofNullable(a)
9         .orElseThrow(() -> new NoSuchElementException("Animal Not Found!"))
10        .get();
11 }
12
13 //Method B to Call Method A with filled/empty sets.
14 static void methodB(Optional<Animal> a){
15     try {
16         System.out.println("Method B with Data");
17         System.out.println(methodA(a));
18     }catch (NoSuchElementException e) {
19         System.out.println(e.getMessage());
20     }
21
22     try {}
23     {
24         System.out.println("Method B with Empty");
25         System.out.println(methodA(null));
26     }catch (NoSuchElementException e) {
27         System.out.println(e.getMessage());
28     }
29 }
30
31 public static void main(String[] args) {
32     Optional<Animal> a = Optional.of(new Animal("Dog", "Everywhere", 999));
33     methodB(a);
34 }
35 }
36
37 }
```

#3 Results



The screenshot shows the Eclipse IDE interface. The title bar reads "eclipse-workspace - Week 10 CHW/src/Week10CHW.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations and development tools. The console window at the bottom displays the following output:

```
<terminated> Week10CHW_3 [Java Application] C:\Users\da3mo\p2\pool\plugins\org.eclipse.justi.openjdk hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (Oct 1, 2021, 8:39:15 PM)
Picked up _JAVA_OPTIONS: -Xmx1g -Xms1g
Method B with Data
Everywhere Dog
Method B with Empty
Animal Not Found!
```

The Windows taskbar at the bottom shows the search bar, task view button, and several application icons. The system tray on the right indicates a temperature of 76°F, clear weather, and the date/time as 8:39 PM on 10/1/2021.

URL to GitHub Repository:

<https://github.com/JDu8Du8/CSN-BE-Week10-CHW>