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: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. 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 an .sql file with all your queries and your Java project code 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:

Create a class of whatever type you want (Animal, Person, Camera, Cheese, etc.).

- Do not implement the Comparable interface.
- Add a name instance variable so that you can tell the objects apart.
- Add getters, setters and/or a constructor as appropriate.
- Add a toString method that returns the name and object type (like "Pentax Camera").
- 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".
- Create a static list of these objects, adding at least 4 objects to the list.

- In another class, write a method to sort the objects using a Lambda expression using the compare method you created earlier.
- Write a method to sort the objects using a Method Reference to the compare method you created earlier.
- Create a main method to call the sort methods.
- Print the list after sorting (System.out.println).

Create a new class with a main method. Using the list of objects you created in the prior step.

- Create a Stream from the list of objects.
- Turn the Stream of object to a Stream of String (use the map method for this).
- 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.)
- Collect the Stream and return a comma-separated list of names as a single String. Hint: use Collectors.joining(", ") for this.
- Print the resulting String.

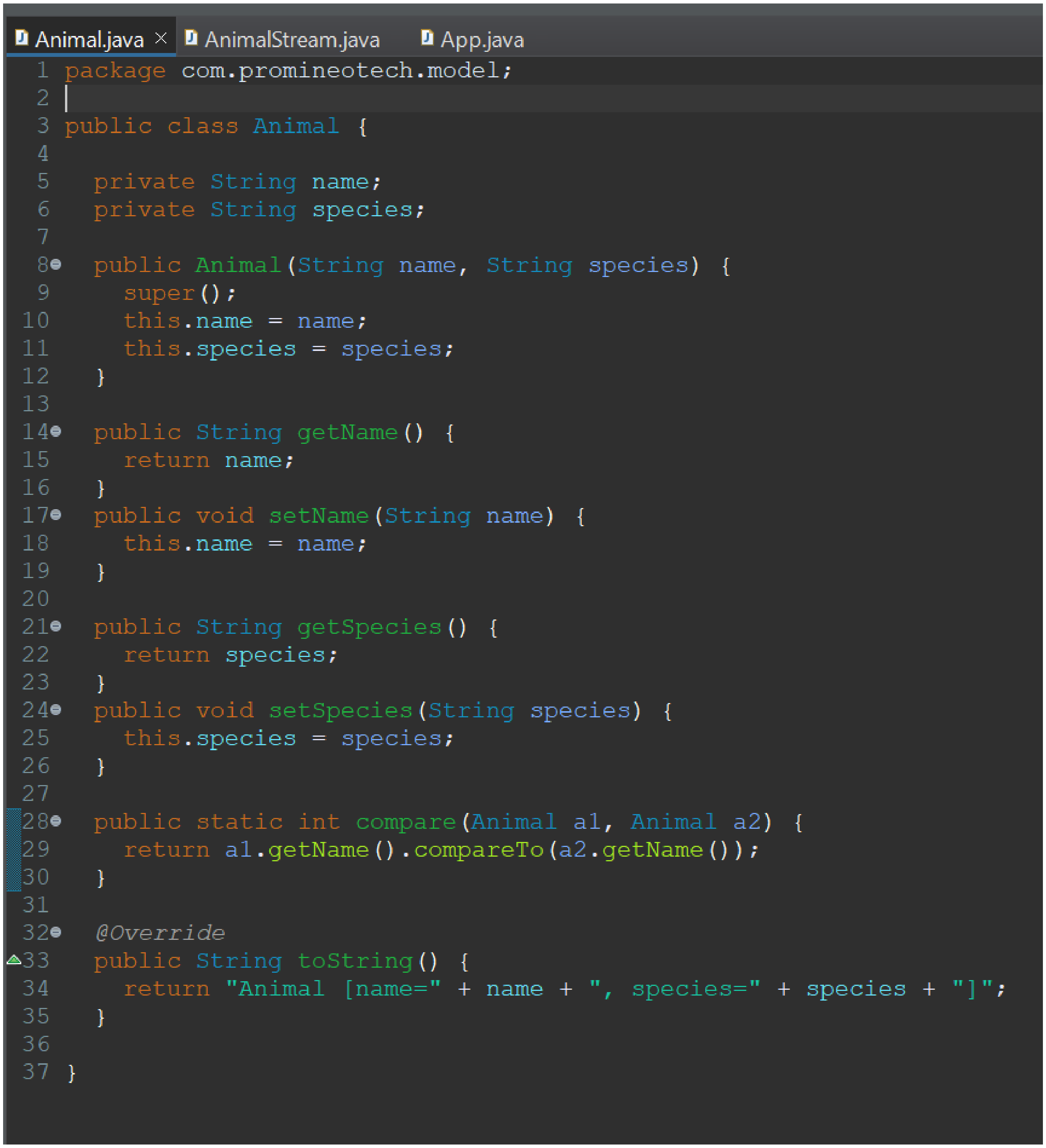
Create a new class with a main method. Create a method (method a) that accepts an Optional of some type of object (Animal, Person, Camera, etc.).

- 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:

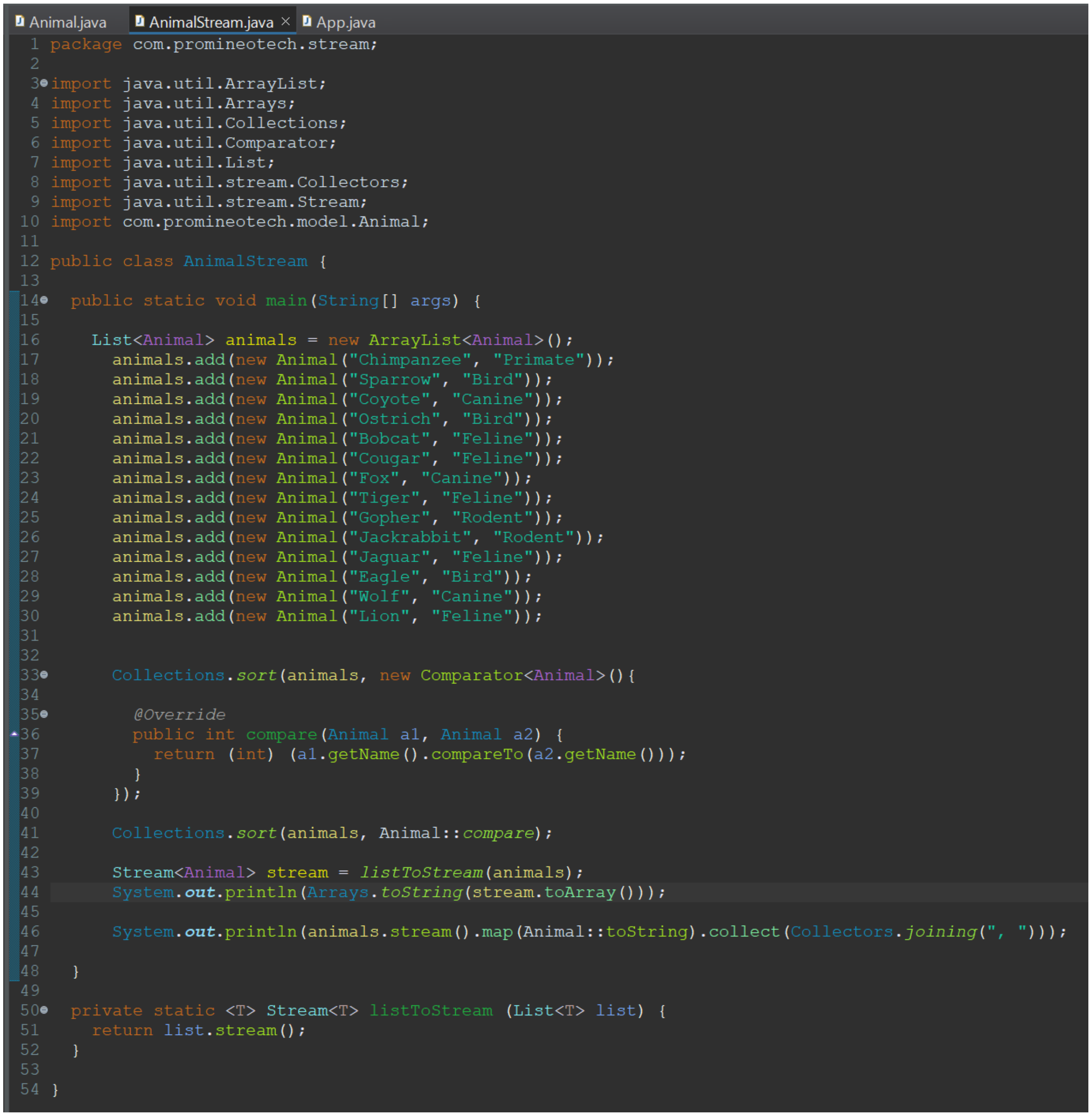
  public Cheese cheesyMethod(Optional<Cheese> optionalCheese) {...}

- The method should throw a NoSuchElementException with a custom message if the object is not present.
- 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).
- 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()).
- 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.

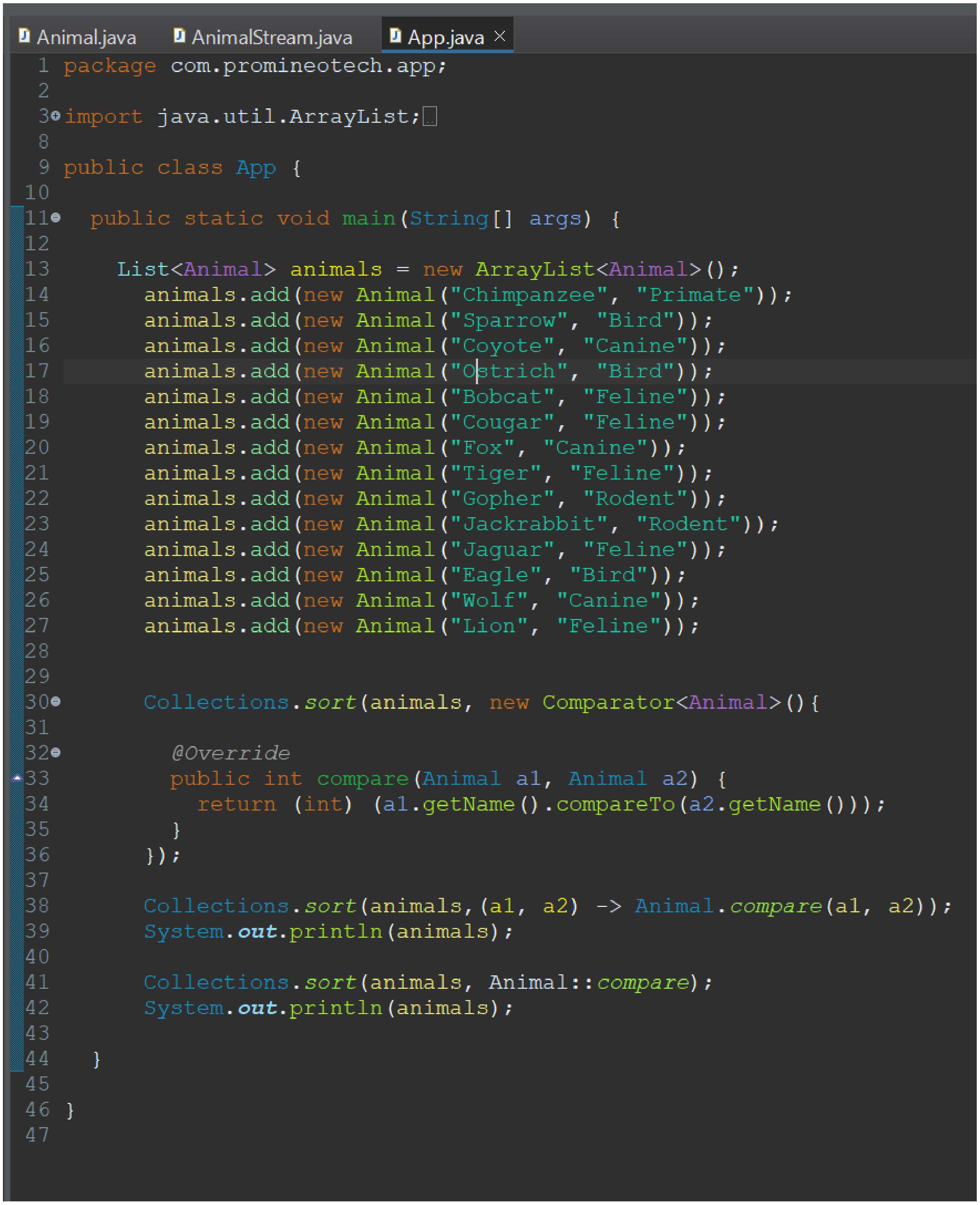
Screenshots of Code:



```java
package com.promineotech.model;

public class Animal {

	private String name;
	private String species;

	public Animal(String name, String species) {
		super();
		this.name = name;
		this.species = species;
	}

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}

	public String getSpecies() {
		return species;
	}
	public void setSpecies(String species) {
		this.species = species;
	}

	public static int compare(Animal a1, Animal a2) {
		return a1.getName().compareTo(a2.getName());
	}

	@Override
	public String toString() {
		return "Animal [name=" + name + ", species=" + species + "]";
	}

}
```



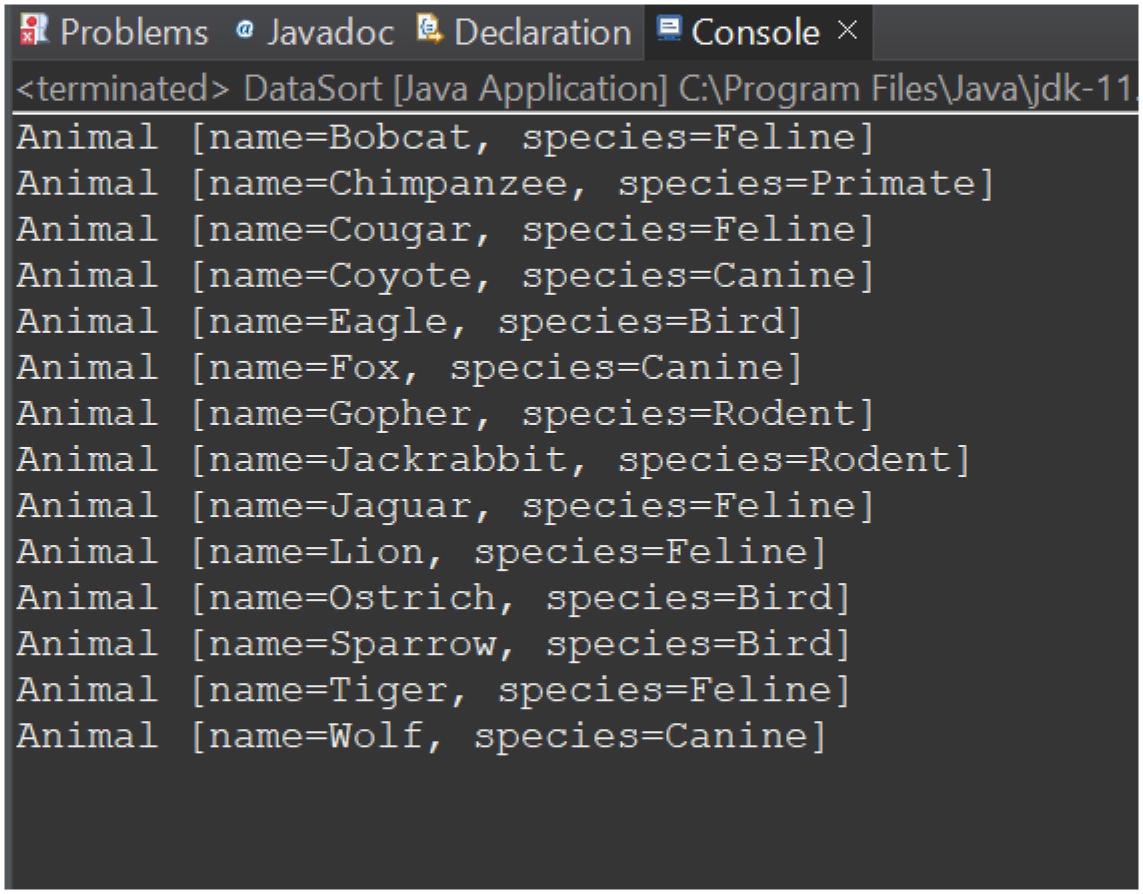
```java
package com.promineotech.stream;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;
import com.promineotech.model.Animal;

public class AnimalStream {

  public static void main(String[] args) {

    List<Animal> animals = new ArrayList<Animal>();
      animals.add(new Animal("Chimpanzee", "Primate"));
      animals.add(new Animal("Sparrow", "Bird"));
      animals.add(new Animal("Coyote", "Canine"));
      animals.add(new Animal("Ostrich", "Bird"));
      animals.add(new Animal("Bobcat", "Feline"));
      animals.add(new Animal("Cougar", "Feline"));
      animals.add(new Animal("Fox", "Canine"));
      animals.add(new Animal("Tiger", "Feline"));
      animals.add(new Animal("Gopher", "Rodent"));
      animals.add(new Animal("Jackrabbit", "Rodent"));
      animals.add(new Animal("Jaguar", "Feline"));
      animals.add(new Animal("Eagle", "Bird"));
      animals.add(new Animal("Wolf", "Canine"));
      animals.add(new Animal("Lion", "Feline"));


      Collections.sort(animals, new Comparator<Animal>(){

        @Override
        public int compare(Animal a1, Animal a2) {
          return (int) (a1.getName().compareTo(a2.getName()));
        }
      });

      Collections.sort(animals, Animal::compare);

      Stream<Animal> stream = listToStream(animals);
      System.out.println(Arrays.toString(stream.toArray()));

      System.out.println(animals.stream().map(Animal::toString).collect(Collectors.joining(", ")));

  }

  private static <T> Stream<T> listToStream (List<T> list) {
    return list.stream();
  }

}
```



Animal.java AnimalStream.java App.java

```java
package com.promineotech.app;

import java.util.ArrayList;

public class App {

  public static void main(String[] args) {

    List<Animal> animals = new ArrayList<Animal>();
      animals.add(new Animal("Chimpanzee", "Primate"));
      animals.add(new Animal("Sparrow", "Bird"));
      animals.add(new Animal("Coyote", "Canine"));
      animals.add(new Animal("Ostrich", "Bird"));
      animals.add(new Animal("Bobcat", "Feline"));
      animals.add(new Animal("Cougar", "Feline"));
      animals.add(new Animal("Fox", "Canine"));
      animals.add(new Animal("Tiger", "Feline"));
      animals.add(new Animal("Gopher", "Rodent"));
      animals.add(new Animal("Jackrabbit", "Rodent"));
      animals.add(new Animal("Jaguar", "Feline"));
      animals.add(new Animal("Eagle", "Bird"));
      animals.add(new Animal("Wolf", "Canine"));
      animals.add(new Animal("Lion", "Feline"));


      Collections.sort(animals, new Comparator<Animal>(){

        @Override
        public int compare(Animal a1, Animal a2) {
          return (int) (a1.getName().compareTo(a2.getName()));
        }
      });

      Collections.sort(animals,(a1, a2) -> Animal.compare(a1, a2));
      System.out.println(animals);

      Collections.sort(animals, Animal::compare);
      System.out.println(animals);

  }

}
```

Screenshots of Running Application:



```
<terminated> DataSort [Java Application] C:\Program Files\Java\jdk-11
Animal [name=Bobcat, species=Feline]
Animal [name=Chimpanzee, species=Primate]
Animal [name=Cougar, species=Feline]
Animal [name=Coyote, species=Canine]
Animal [name=Eagle, species=Bird]
Animal [name=Fox, species=Canine]
Animal [name=Gopher, species=Rodent]
Animal [name=Jackrabbit, species=Rodent]
Animal [name=Jaguar, species=Feline]
Animal [name=Lion, species=Feline]
Animal [name=Ostrich, species=Bird]
Animal [name=Sparrow, species=Bird]
Animal [name=Tiger, species=Feline]
Animal [name=Wolf, species=Canine]
```

URL to GitHub Repository:

https://github.com/jprengaman/BESD-Week11