Assignment8 of Xiaowei Liu

```
>>>>>>>>>>>>>
Animal{name='Pipi', breed='Cat', size=12}
Animal{name='Jack', breed='Dog', size=13}
Animal{name='Rose', breed='Cat2', size=17}
>>>>>>>>>>>>>
Animal{name='Pipi', breed='Cat', size=12}
Animal{name='Rose', breed='Cat2', size=17}
Animal{name='Jack', breed='Dog', size=13}
>>>>>>>>>>>>>
Animal{name='Pipi', breed='Cat', size=12}
Animal{name='Jack', breed='Dog', size=13}
Animal{name='Rose', breed='Cat2', size=17}
>>>>>>>>>>>>>>>>>>>>>>
Serialize Done!
Deserialize Done!
Animal{name='Pipi', breed='Cat', size=12}
Animal{name='Jack', breed='Dog', size=13}
Animal{name='Rose', breed='Cat2', size=17}

Process finished with exit code 0
```

Tester.java

```
// Importing necessary libraries for input-output operations and working with collections
import java.io.*;
import java.util.*;

// The main class of your program
public class Tester {

    // The main method where the program execution begins
    public static void main(String[] args) throws IOException, ClassNotFoundException {

        // Creating instances of Cat and Dog objects
        Cat cat = new Cat("Pipi", "Cat", 12);
        Dog dog = new Dog("Jack", "Dog", 13);
        Cat cat1 = new Cat("Rose", "Cat2", 17);
```

```
        // Creating a list to store Animal objects
        List<Animal> list = new LinkedList<Animal>();
        list.add(cat);
        list.add(dog);
        list.add(cat1);

        // Printing information about each animal in the list
        System.out.println(">>>>>>>>>>>>>");
        for (Animal animal : list) {
            System.out.println(animal.toString());
        }

        // Sorting the list based on breed using BreedComparator
        System.out.println(">>>>>>>>>>>>>");
        Collections.sort(list, new BreedComparator());
        for (Animal animal : list) {
            System.out.println(animal.toString());
        }

        // Sorting the list based on size using SizeComparator
        System.out.println(">>>>>>>>>>>>>");
        Collections.sort(list, new SizeComparator());
        for (Animal animal : list) {
            System.out.println(animal.toString());
        }

        // Serializing the list of animals to a file
        System.out.println(">>>>>>>>>>>>>>>>>>>>>>>");
        Serialize(list);

        // Deserializing and printing the animals from the file
        Deserialize();
    }

    // Comparator for sorting animals based on breed
    public static class BreedComparator implements Comparator<Animal> {
        @Override
        public int compare(Animal animal1, Animal animal2) {
            return animal1.getBreed().compareTo(animal2.getBreed());
        }
    }

    // Comparator for sorting animals based on size
    public static class SizeComparator implements Comparator<Animal> {
        @Override
        public int compare(Animal animal1, Animal animal2) {
            return Integer.compare(animal1.getSize(), animal2.getSize());
        }
    }

    // Method to deserialize the list of animals from a file
    public static void Deserialize() throws IOException, ClassNotFoundException {

        // Initialize an ObjectInputStream to read from the file
        ObjectInputStream in = new ObjectInputStream(new FileInputStream("list.txt"));

        // Read the object from the file
```

```
        Object obj = in.readObject();
        System.out.println("Deserialize Done!");

        // Print information about each animal in the deserialized list
        for (Animal animal : (List<Animal>) obj) {
            System.out.println(animal.toString());
        }
    }

    // Method to serialize and write the list of animals to a file
    public static void Serialize(List<Animal> list) throws IOException {

        // Initialize an ObjectOutputStream to write to the file
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("list.txt"));

        // Write the list of animals to the file
        out.writeObject(list);
        System.out.println("Serialize Done!");
    }
}
```

Animal.java

```
import java.io.Serializable;

// This is a class representing an Animal, which implements Comparable and Serializable interfaces.
class Animal implements Comparable<Animal>, Serializable {
    // Attributes of an Animal
    private String name;
    private String breed;
    private int size;

    // Default constructor for Animal
    public Animal() {
    }

    // Parameterized constructor for Animal, allowing initialization with specific name, breed, and size
    public Animal(String name, String breed, int size) {
        this.name = name;
        this.breed = breed;
        this.size = size;
    }

    // Method to compare animals based on their names (implements Comparable interface)
    public int compareTo(Animal animal) {
        return this.name.compareTo(animal.name);
    }

    // Getter method for retrieving the name of the animal
    public String getName() {
        return name;
    }

    // Setter method for setting the name of the animal
    public void setName(String name) {
        this.name = name;
```

```
    }

    // Getter method for retrieving the breed of the animal
    public String getBreed() {
        return breed;
    }

    // Setter method for setting the breed of the animal
    public void setBreed(String breed) {
        this.breed = breed;
    }

    // Getter method for retrieving the size of the animal
    public int getSize() {
        return size;
    }

    // Setter method for setting the size of the animal
    public void setSize(int size) {
        this.size = size;
    }

    // toString method for providing a string representation of the Animal object
    @Override
    public String toString() {
        return "Animal{" +
                "name='" + name + '\'' +
                ", breed='" + breed + '\'' +
                ", size=" + size +
                '}';
    }
}

// This is a subclass of Animal representing a Cat.
class Cat extends Animal {
    // Constructor for Cat, utilizing the constructor of the superclass (Animal)
    public Cat(String name, String breed, int size) {
        super(name, breed, size);
    }

    // Method representing a sound that a cat makes (not implemented here, just a placeholder)
    public String sound() {
        return "";  // Placeholder for the sound method
    }
}

// This is a subclass of Animal representing a Dog.
class Dog extends Animal {
    // Constructor for Dog, utilizing the constructor of the superclass (Animal)
    public Dog(String name, String breed, int size) {
        super(name, breed, size);
    }

    // Method representing a sound that a dog makes (not implemented here, just a placeholder)
    public String sound() {
        return "";  // Placeholder for the sound method
```

```
    }
}
```