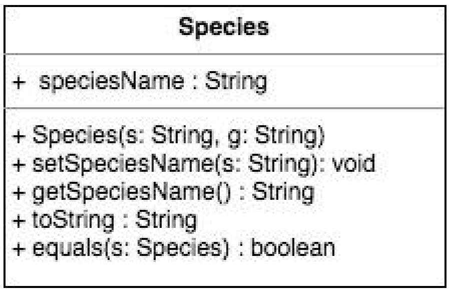
Question 1

a. A parent-child relationship exists between Genus and Species objects (inheritance). Genus is the parent class, while Species is the child class

b. The relationship between Species and Specimen objects is has a relationship

c. 

```
Species
+ speciesName : String
+ Species(s: String, g: String)
+ setSpeciesName(s: String): void
+ getSpeciesName() : String
+ toString : String
+ equals(s: Species) : boolean
```

d. Benefits :

- Has-a relationship is composition relationship which is a productive way of code reuse.
- The programming team gains readability as a result of their efforts. The programming team was able to avoid duplicate methods and variables by using relationships, making it easier to read

e.
(i) The Species class inherits all of the methods from the Genus class, which implies that the toString() function in the Species class does not return an error since it is in the Genus class
(ii) Polymorphism

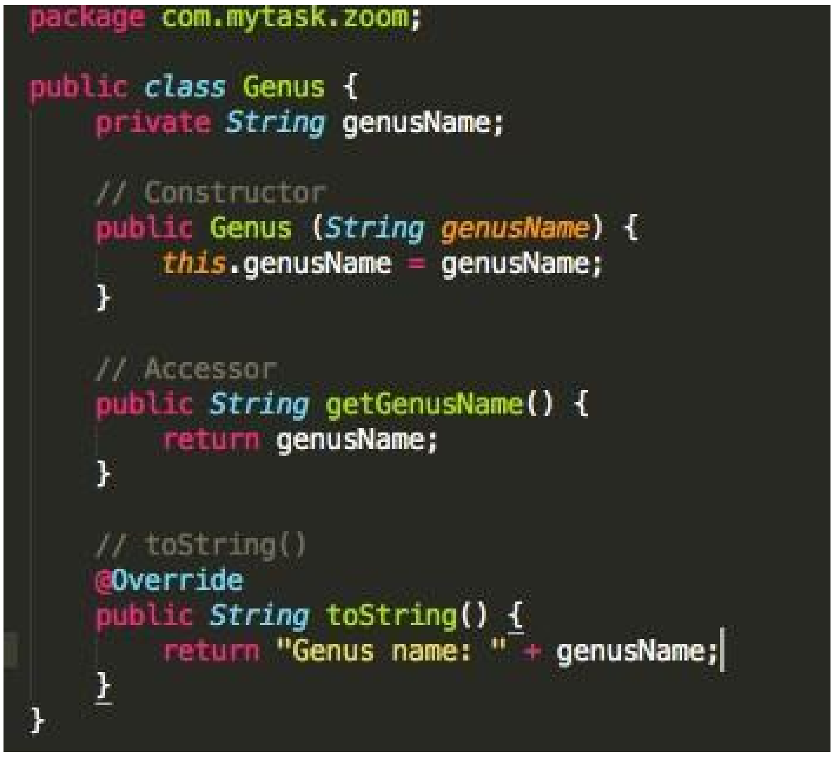
Question 2

a. Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

b. **Benefits** :
- The encapsulated code is more flexible and easy to change with new requirements.
- It prevents the other classes to access the private fields.

c. public String getName()
public int getCage()
public Species getTOA()

d. private String name
private int cageNumber
private Species toa

e. 

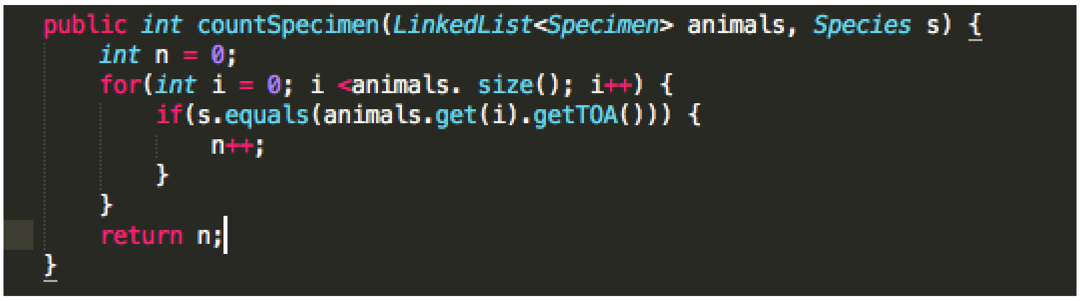
```java
package com.mytask.zoom;

public class Genus {
    private String genusName;

    // Constructor
    public Genus (String genusName) {
        this.genusName = genusName;
    }

    // Accessor
    public String getGenusName() {
        return genusName;
    }

    // toString()
    @Override
    public String toString() {
        return "Genus name: " + genusName;
    }
}
```

f. **Advantage**
The advantage of having the Specimen object as a subclass of the Species object is that Specimen inherits all of the parent class's attributes and behaviors. If Specimen inherits from Species, it indicates there will be no duplicates, or that the attributes and behaviors described in Species will not need to be specified again

Disadvantage
The disadvantage of having the Specimen object as a subclass of the Species object is that data elements in the class are sometimes left unused, which can result in memory waste

Question 3

a. To begin, make a new string private variable for markings. Then, in the constructor, provide a string parameter for marking. Finally, include the marking getter and setter methods.

b. 

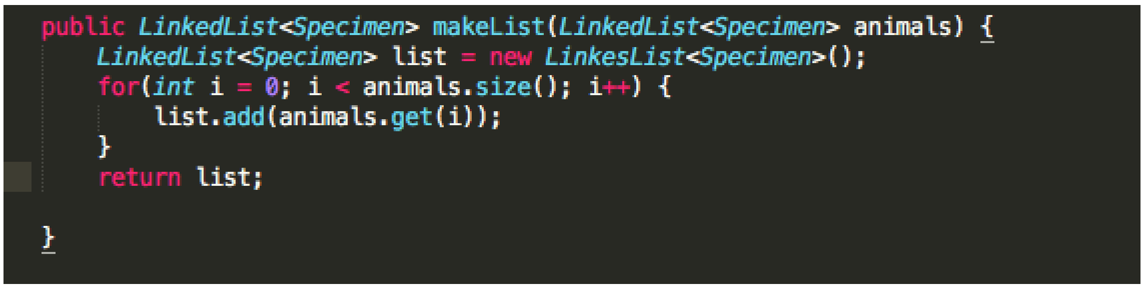
```java
public int countSpecimen(LinkedList<Specimen> animals, Species s) {
    int n = 0;
    for(int i = 0; i <animals. size(); i++) {
        if(s.equals(animals.get(i).getTOA())) {
            n++;
        }
    }
    return n;
}
```

c. Pseudocode

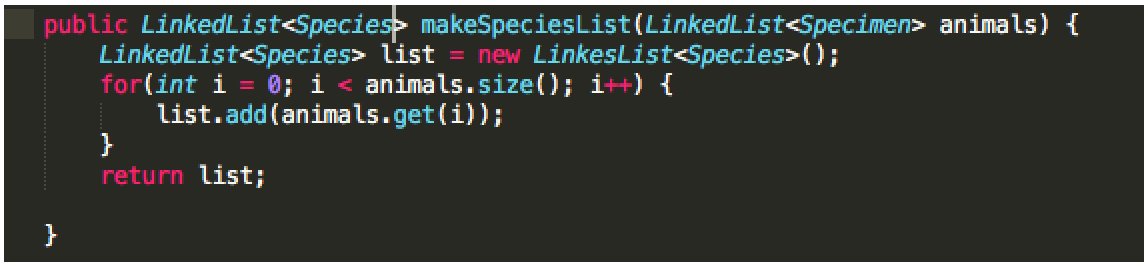
```
listSpecies(Specimen[] Animals) {
        LinkedList<String> list = new LinkedList<String>
        for each animal in Animals {
            if animal species does not exist in the list {
                add the species to the list
        }
    }
}
```

Question 4

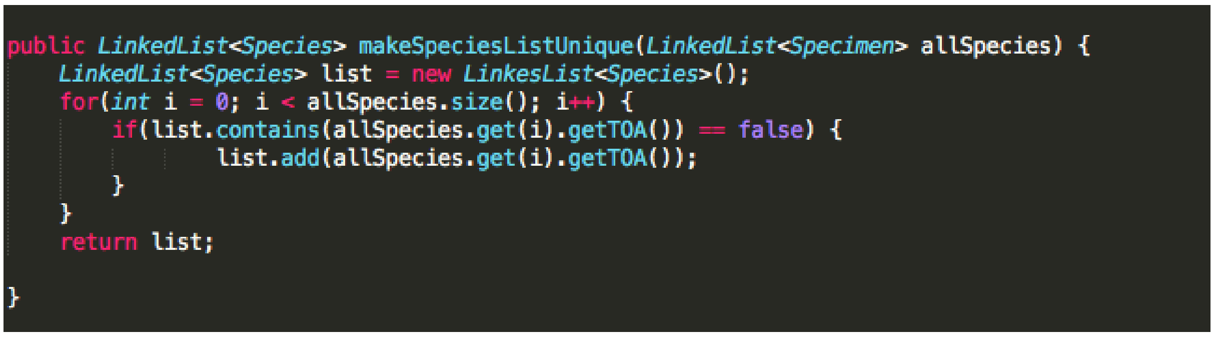
a. Features :
   - Abstract data types display behavior specified by a set of operations that may be used to arrange data. A list is an example of an abstract data type. A list is an example of an abstract data type.
   - A type is exported by abstract data types.

b. 

```java
public LinkedList<Specimen> makeList(LinkedList<Specimen> animals) {
    LinkedList<Specimen> list = new LinkesList<Specimen>();
    for(int i = 0; i < animals.size(); i++) {
        list.add(animals.get(i));
    }
    return list;

}
```

c. 

```java
public LinkedList<Species> makeSpeciesList(LinkedList<Specimen> animals) {
    LinkedList<Species> list = new LinkesList<Species>();
    for(int i = 0; i < animals.size(); i++) {
        list.add(animals.get(i));
    }
    return list;

}
```

d. 

```java
public LinkedList<Species> makeSpeciesListUnique(LinkedList<Specimen> allSpecies) {
    LinkedList<Species> list = new LinkesList<Species>();
    for(int i = 0; i < allSpecies.size(); i++) {
        if(list.contains(allSpecies.get(i).getTOA()) == false) {
            list.add(allSpecies.get(i).getTOA());
        }
    }
    return list;

}
```