Question Set 1

A. Genus is a superclass of Species. Species is a child class to Genus.
B. Specimen has a Species object
C. Diagram Below

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

D. 1. The programming team benefits from increased **readability** from implementing OOP inheritance. This is because the code becomes more structured and easy to understand. 2. Furthermore, the team benefits from **reusability;** the team does not have to write the same code for different classes, and is hence also cleaner, making it easier to understand.
E.
i. toString is defined in both classes, however if implemented in the child class, that is always called unless specifically stated to call the super classes toString method. Thus, the Species class toString method is called.

ii. Overriding

Question Set 2

A. Encapsulation is hiding its data members with methods within a class such that its direct access is restricted to private members of that class.
B. 1. Encapsulation provides security as it prevents direct access of private variables from outside of the class. These private members can only be modified through a public interface when accessed from outside of the class.
2. Encapsulation promotes the maintainability of a codebase, as code changes are made independently in one class without affecting others.
C. getName()
D. cageNumber
E. .

```java
public class Genus {
    private String name;

    public Genus(String n) {
        this.name = n;
    }

    // Accessor
    public String getName() {
        return name;
    }

    // toString
    public String toString() {
        return "Name: " + name;
    }
}
```

F. 1. Benefits from code reusability, hence the different Species must not redefine all the code from its.

2. A disadvantage is that the inherited class's members may go unused in a subclass, leading to wasted resources.

Question Set 3

A. Within the superclass, add a private **String** member called *description.* This string should be initialised within the constructor. Then, a setter and getter method must be added. Add public String getDescription() and public void setDescription(String s). These members are passed to the subclasses and to each animal object.

B.

```
public int countSpecimen(Specimen[] animals, Specimen s)
{
    int count = 0;
    for (int i = 0; i < animals.length; i++){
        if(s.equals(animals[i].getTOA())){
            count++;
        }
    }
    return count;
}
```

C.

LinkedList<String> allSpecies = new LinkedList<String>
Loop through each animal in the list
if animal species IS NOT AVAILABLE IN **allSpecies**
Add animal species to allSpecies
end if
end loop
RETURN **allSpecies**

Question Set 4

A. **1.** Behaviours are defined by a set of value and set of operations
2. Abstract data types export a type.

B.

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

C.

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

D.

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