Inheritance Assignment 2

Question set 1

1. The relationship between the Genus and Species objects is inheritance which the Species subclass inherits from the Genus superclass.
2. The relationship between the Specimen and Species objects is Aggregation which the specimen class has a member variable that is a species object Species and specimen are associated. They "has a relationship”.
3. UML for species class

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

1. A. avoids programmer to rewrite the same code multiple times.

B. make programmers program to be more efficient because functionality inherited from a parent class does not need to be re-tested in the new class.

1. (i). it does not cause an error because the Species subclass overrides the toString() method from the genus superclass

(ii). Overriding or polymorphism.

Question set 2

1. Encapsulation refers to the practice of hiding the structure and representation of data within a class and making it accessible outside the class via accessor functions.
2. A. **Data Hiding –** It can provide the programmer to hide the inner classes and the user to give access only to the desired codes. It allows the programmer to not allow the user to know how variables and data store.

B. **Reusability –** It allows the user to a programmer to use the existing code again and again in an effective way.

1. private String name; (variable)

public String getName() (accessor method)

1. private String name
2. Code:

public class Genus {  
 private String genusName;  
  
 public Genus(){}  
  
 public Genus(String g)  
 {  
 this.genusName = g;  
 }  
  
 public String getGenusName() {  
 return genusName;  
 }  
  
 public void setGenusName(String genusName) {  
 this.genusName = genusName;  
 }  
  
 @Override  
 public String toString() {  
 return "Genus{" +  
 "genusName='" + genusName + '\'' +  
 '}';  
 }  
}

1. Advantage:

Advantage of having the specimen object as a subclass of the species object is so the subclass (child/ specimen) can use attributes or method which the superclass (parent/ species). From that, it avoids programmer to write the same code or function over and over.

Disadvantage:

Disadvantage of having the specimen object as a subclass of the species object is it can cause logical inconsistent which means data in the Species class (parent) may not be consistent across the associated Specimen class(child)

Question set 3

1. The changes needed are:
2. Create new instance variable in specimen class called description of markings.
3. Create setter and getter for the new instance variable.
4. Add the new instance variable description into the toString() method.
5. Code:
6. public void countSpecimen( Specimen [] animals, Species s)  
   {  
    int counterAnimals = 0;  
    for (int i = 0; i < animals.length; i++)  
    if (s.equals(animals[i].getTOA())) {  
    counterAnimals++;  
    }  
    System.*out*.println(counterAnimals);  
   }

* First, we define a function with array list type named listSpecies which contains parameter array of animals from specimen class.
* Then I create a new arraylist named listOfSpecies
* Then, I loop through all the elements inside animals array. Inside the loop, I check whether each element in animals already contain the type of animal or not in the arraylist listOfSpecies.
* If yes, I continue to loop over and check
* If no, I add it inside the arraylist listOfSpecies.
* Lastly, I return the arraylist

Question set 4

1. Java abstract data type(ADT) in a data structure is a type of data type whose behaviour is defined by a set of operations and values.In Java abstract data type, we can only know what operations are to be performed and not how to perform them i.e. it does not tell how algorithms are to be implemented or how the data will be organized in the memory. This type is thus called abstract as it does not give the view of implementation.
2. public LinkedList<Specimen> makeList(Specimen [] animals)  
   {  
    LinkedList<Specimen> listSpecimen= new LinkedList<>();  
    for (int i = 0; i<animals.length; i++)  
    {  
    listSpecimen.add(animals[i]);  
    }  
    return listSpecimen;  
   }
3. public LinkedList<Species> makeSpeciesList( LinkedList<Specimen> animals)  
   {  
    LinkedList<Species> listSpecies = new LinkedList<>();  
     
    for (Specimen animal : animals) {  
    listSpecies.add(animal.getTOA());  
    }  
    return listSpecies;  
     
   }
4. public LinkedList<Species> makeSpeciesListUnique ( LinkedList<Species> allSpecies)  
   {  
    LinkedList<Species> listUniqueSpecies = new LinkedList<>();  
    for (Species species : allSpecies) {  
    if (listUniqueSpecies.contains(species))  
    {  
    continue;  
    }  
    else  
    {  
    listUniqueSpecies.add(species);  
    }  
     
    }  
     
    return listUniqueSpecies;  
   }