Question Set 1:

1. “Genus” is the parent of its child class “Species”.
2. The relationship between “Species” and “Specimen” is a “has a” relationship.
3. Table

   Description automatically generated with medium confidence
4. It is going to be easy for programmers to maintain the code because all the species has the same fields and because of polymorphism, Species objects can be treated as Genus objects.
   1. Because Species class extends from the Genus class and therefore, it is able to use the methods of the Genus class.
   2. Overriding

Question Set 2:

1. The bundling of data and the methods that are used on the data.
2. 1. Restricts unwanted access to an object by clients

2. Makes the application easier to maintain

1. getCage()
2. private int cageNumber

public class Genus {

private String genusName;

public Genus(String genusName) {

this.genusName = genusName;

}

public void setGenusName(String genusName) {

this.genusName = genusName;

}

public String getGenusName() {

return genusName;

}

@Override

public String toString() {

return "Genus: " + genusName;

}

}

1. Advantage: Since the Specimen class extends from the Species class, the new instances for Specimen could also use the methods of its superclass, Species.

Disadvantage: The Species and Specimen classes are bounded together, meaning a user cannot use either of the classes independently.

Question Set 3:

1. The description of each animal’s marking should be an instance variable in the specimen class. A setter and getter should also be made for the variable. Lastly, the description of the markings could be included in the toString() method.

public void countSpecimens(Specimen[] animals, Species s) {

int count = 0;

for (Specimen animal : animals) {

if (s == animal.getTOA()) {

count++;

}

}

System.out.println(count);

}

1. –

listSpecies (Specimen[] animals) {

Species[] species;

Species name;

for each animal in animals array {

name = animal.getTOA();

if name is in species array {

continue;

}

else {

add name to species array;

}

}

print species array;

Question Set 4:

* + Exports a type.
  + Exports an operation.

public LinkedList<Specimen> makeList(Specimen[] animals) {

LinkedList<Specimen> animals = new LinkedList<Specimen>();

Collections.addAll(animals, animals);

return specimens;

}

public LinkedList<Species> makeSpeciesList(LinkedList<Specimen> animals) {

LinkedList<Species> allSpecies = new LinkedList<Species>();

for (Specimen animal : animals) {

allSpecies.add(animal.getTOA());

}

return allSpecies;

}

public LinkedList<Species> makeSpeciesListUnique (LinkedList<Species> allSpecies) {

LinkedList<Species> species = new LinkedList<Species>();

for (Species toa : allSpecies) {

if (species.isEmpty()) {

species.add(toa);

} else if (species.contains(toa)) {

;

} else {

species.add(toa);

}

}

return species;

}