Assignment 2 – Inheritance

Question Set 1

a. Species extends Genus

b. Specimen class has a member variable that is of the Species object

c. UML Diagram:

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

d. They show the actual relationships between the hierarchy of the real world. The use of inheritance can make the code more efficient as it prevent the programmers from repeating the codes when coding for objects.

e. i. Because the compiler was able to run the correct method as the two methods are both related to the Species Class.
ii. Overriding

Question Set 2

a. Encapsulation :

It is also known as data/information hiding. It is a mechanism of wrapping the variables and methods together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can only be accessed through the methods of their current class.

b. Benefits of Encapsulation:

a. protects an object from unwanted access
b. Increases usability

c. Accessor method: public int getCage()

d. Instance Variable: private int cageNumber

e. Code:

```
public class Genus
{
        private String genus;

        public Genus(String genus)
        {
                this.genus = genus;
        }
```

```
        public String getGenus()
        {
                return genus;
        }

        public void setGenus(String genus)
        {
                 this.genusName = genus;
        }

        @Override
        public String toString()
        {
                return "Genus{" + "genus='" + genus + '\'' + '}';
        }
}
```

f. Advantage: All the methods found in the Species class will be inherited by the Specimen class, meaning that programmers will not have to repeat the same codes and hence makes it more efficient

Disadvantage: Some methods in the Species class will not match up to that of Specimen class, however the Specimen class still needs to carry those methods.

Question Set 3

a. Changes to be made :

I. We have to make an instance variable within the Specimen class for the description of each individual' s markings.
II. We have to create getter and setter methods for the markings.

b. Code:

```
public void countSpecimens( Specimen[] animals, Species s )
{
        int sCount = 0;
        int i;
        for (i=0; i< animals.length; i++)
        {
                if (s.equals(animals[i].getTOA())
                 {
                sCount ++;
                }
         }

        System.out.println( sCount );
}
```

b. Pseudocode:

Question Set 4

a. **Features of ADT:**
   a. Interface
   b. Memory
   c. Functions/ Methods (Public and Private)

b. **Code:**

```
LinkedList makeList( Specimen [] animals )
{
        LinkedList specimenList = new LinkedList();
        for (int i=0; i<animals.length; i++ )
        {
        specimenList.addHead( animals[i] );
        }
        return specimenList;
}
```

c. **Code:**

```
public LinkedList makeSpeciesList( LinkedList animals )
{
    LinkedList sList = new LinkedList();
    Specimen indivSpecimen = (Specimen) animals.getHead();
    while (indivSpecimen != null)
    {
        sList.addHead( indivSpecimen.getTOA() );
        indivSpecimen = (Specimen) animals.getNext();
    }
    return sList;
}
```

d. **Code:**

```
public void makeSpeciesListUnique( LinkedList allSpecies )
{
    boolean found;
    LinkedList unique = new LinkedList();
    Species current;
    Species speciesType = (Species) allSpecies.getHead();
    while (speciesType != null )
    {
        found = false;
        current = (Species) unique.getHead();
        while (current != null)
        {
            if (current == speciesType) found = true;
            current = (Species) unique.getNext();
        }
        If ( !found ) unique.addHead( speciesType );
        speciesType = (Species) allSpecies.getNext();
    }
    allSpecies = unique;
```

```
}
```