CS261

ASSIGNMENT 5

NAME:

ARCHIT AGRAWAL

ROLL NO. :

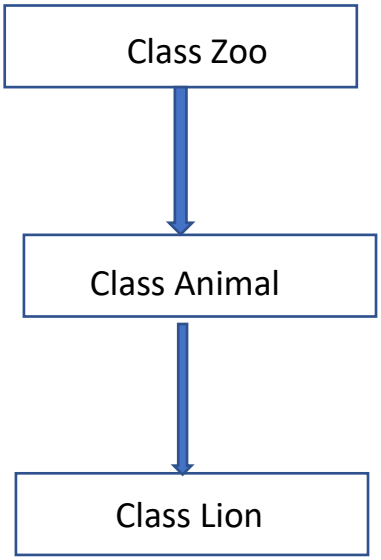
202052307

SECTION:

A

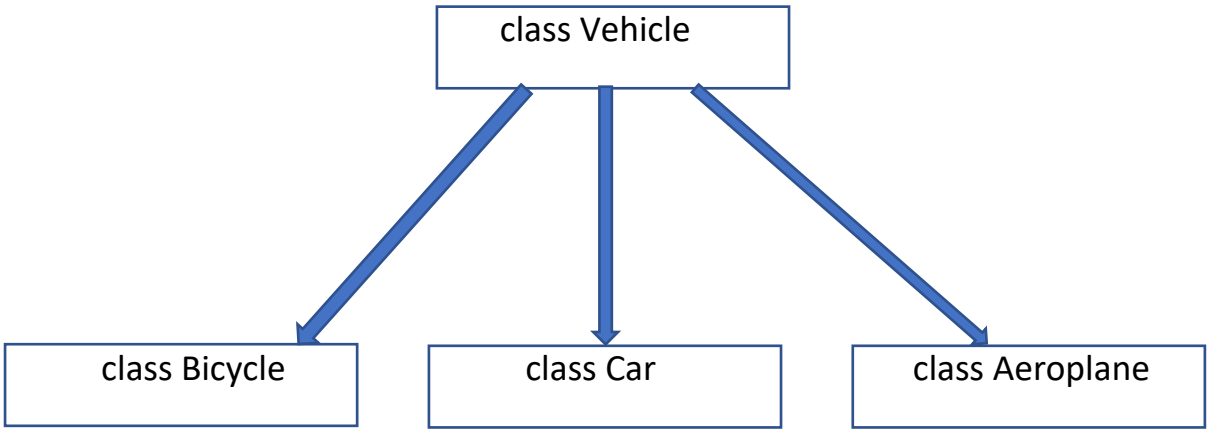
1. **What is the difference between multilevel and hierarchical inheritance. Write a program to implement multilevel inheritance by applying various access controls to its data members and methods.**

In **Multilevel Inheritance**, a derived (child) class will inherit a base (parent) class as well as acts as a base (parent) class for another class.



As shown in the above figure, class Animal is derived from the class Zoo as well as class Animal acts as a parent class for class Lion.

In **Hierarchical Inheritance**, one base class is derived by many child classes.



As the above figure shows, class Bicycle, class Car and class Aeroplane are all derived from same parent class Vehicle.

Approach

We create a class Zoo. It has several data members which stores its name and location as private. We wrote a getter method to get the name and location of zoo and these getter methods are public.

Next, we create class Animals, which have numOfSpecies and typesOfAnimals as data members with private access. It inherits the class Zoo. It also has specie and type instances, since each animal has one.

Next, we created a class named Lion, it inherits the Animals class. In its constructor, the specie and type are set by default.

Code

```
import java.util.*;

class Zoo{

    //these two data members are set to private so as to
    //prevent any outsider to change the name and address of zoo
    //and also we don't need these in the sub-classes
    private String zooName = "WildLands Animals Zoo";
    private String zooLocation = "Indiranagar, Bengaluru, Karnataka";

    //to get the name and location of Zoo
    //we will form a getter method
    //and set them to public so they can be accessed from anywhere

    public String getName(){
        return zooName;
    }

    public String getLocation(){
        return zooLocation;
    }


}
```

```
class Animals extends Zoo{

    //all different species of animals in the zoo are listed here
    private String[] typesOfAnimals = {"Panthera Leo", "Rufous
Hummingbird", "Masai Giraffe", "Blue Mormon Butterfly"};
    protected String specie;
    protected String type;
    //numOfSpecies is private because it stores different number of
animal species
    private int numOfSpecies = 1835780735;

}

class Lion extends Animals{


    public Lion(){
        this.specie = "Panthera Leo";
        this.type = "Mammal";
    }

    public float weight;

    //this is public so that it can be accessed fron amywhere
    public float getWeight(){
        return weight;
    }

    public void setWeight(float kg){
        this.weight = kg;
    }

}
public class Driver{

    public static void main(String[] args){

        Lion lion1 = new Lion();
        System.out.println(lion1.specie);
    }
}
```

Output

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) {
Panthera Leo
PS C:\Users\Archit\Desktop\cprog>
```

2. Explain different access modifiers with examples. What is the output of following program:

```
class X{
    protected int i = 1221;

    void methodOfX(){
        System.out.println(i);
    }
}

public class MainClass{


    public static void main(String []args){
        X x = new X();
        System.out.println(x,i);
        x.methodOfX();
    }
}
```

Access Modifiers in JAVA are used to restrict the accessibility or scope of a class, constructor, variable, method or data member. There are four different access modifiers available in JAVA:-

- Default
- Private
- Public
- Protected

(a) Default: When no access modifier is specified for a class, method or data member, it is said to have default access by default.

Can be Accessed in:

➔ same class

➔ same package sub-class

➔ same package non sub-class (classes that aren't related)

Can not be Accessed in:

➔ different package sub-class

➔ different package non sub-class

(b) Private: The private access modifier is specified using **private** keyword.

***Can be Accessed in*:**

➔ same class

***Can not be Accessed in*:**

➔ same package sub-class
➔ same package non sub-class (classes that aren't related)
➔ different package sub-class
➔ different package non sub-class

Refer to the code of Q1. If I want to access numOfSpecies using object of Lion class, following error throws up.

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) {
Driver.java:63: error: numOfSpecies has private access in Animals
        System.out.println(lion1.numOfSpecies);
                                 ^
1 error
PS C:\Users\Archit\Desktop\cprog>
```

(c) Protected: The protected access modifier is accessed using the keyword **protected.**

***Can be Accessed in*:**

➔ same class
➔ same package sub-class
➔ same package non sub-class (classes that aren't related)
➔ different package sub-class

***Can not be Accessed in*:**

➔ different package non sub-class

(d) Public: The public access modifier is accessed using the keyword **public.**

***Can be Accessed in*:**

➔ same class
➔ same package sub-class
➔ same package non sub-class (classes that aren't related)

➔ different package sub-class
➔ different package non sub-class

These have the widest scope and can be accessed from anywhere in the program. There is no restriction on the scope of public access members.

Refer to code of Q.1

```
public class Driver{

    public static void main(String[] args){

        Lion lion1 = new Lion();
        System.out.println(lion1.getName());
        System.out.println(lion1.getLocation());
    }
}
```

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?
WildLands Animals Zoo
Indiranagar, Bengaluru, Karnataka
PS C:\Users\Archit\Desktop\cprog>
```

This is because getName() and getLocation() has public access.

The output of the given program will be:

1221

1221

The given code will throw an error actually because it has (x,i) in line 13, where it has asked to print. If we change it to “x.i”, then the output will be same as above.

3. Correct the code for overloading methods:

```
public class Figure{

    public String draw(String s){
        return "Figure Drawn";
    }
```

```
    public void draw(String s) {}
    public void draw(double f) {}
}
```

The given code has defined the **draw** method with the same parameter **"String s"** twice. If a method is defined with same parameters twice, then the compiler will get confused which method's body it should enter in and hence a compile time error occurs.

The corrected code for method overloading is given below:

```
public class Figure{

    public String draw(String s){
        return "Figure Drawn";
    }

    //public void draw(String s) {}
    public void draw(double f) {}
}
```

Now, we have two different **draw** methods with different parameters, one with **"String s"** parameter and the other with **"double f"** parameter. Now, since the signature is different, the compiler will be able to decide that in which method's body it should enter and the code will execute without any errors.

4. **Create a class diagram for Q1 and Q2 of this assignment.**