Advanced Programming (CSE201), Quiz -1

Time allocated: 03:20pm – 3:45pm (25 minutes) + 5 minutes for uploading solution

Instructions:

· You must follow all the instructions sent to you earlier over the email.

· Only reasonable and clearly mentioned assumptions (if any) would be accepted.

· For justifications, please be as concise as possible (2-3 sentences only)

Q1) Read the below mentioned problem description and answer below questions:

"*A petshop shop clips the nails of cats and dogs and replaces the gps collar of dogs and tigers. This shop has a groomer responsible for receiving the cats, dogs, and tiger cubs; storing them in two different cages inside the petshop; and for providing the services. These cages are essentially storing cats, dogs, and tiger cubs according to the requested service. One of the cage can store cats and dogs to clip their nails, whereas the other can store tiger cubs and dogs for replacing their gps collar. For clipping the nail, the cat and dog must be feeded. The cats are feeded with milk, whereas dogs are feeded with meat. For replacing the gps collar, the dogs and tiger cubs must be first hypnotized. Hypnotizing the dog means feeding it meat, whereas for tiger cubs it means letting them watch Tom and Jerry cartoons. The groomer receives the cat, dog, and tiger cub throughout the day and then services them together by taking them out of their respective rooms. Any car, dog, and tiger cub can be given for only one type of service.*"

Implement the Object Oriented Implementation of the above program description and identify the class relationships (if any). No need to code the main method. You should only use the concepts taught in Lectures up to interfaces and polymorphism. You must write actual code (no pseudocode). **[7 marks]**

Rubric for Q1: (Total 7 marks)
No need to see the working code. Only see the below points:

1) Classes Shop and Employee
    a) +0.5 marks
2) Data encapsulation being followed inside each classes Shop and Employee
    a) +0.5 marks
3) Two List (or ArrayList) type objects inside Shop class as List<InterfaceA> and List<InterfaceB> (Polymorphism in parameter)
    a) +0.5 marks
4) If Shop instantiates Employee then composition relationship, else association relationship / If employee inherits the shop class
    a) +0.75 marks
5) The employee has association relationship with Shop / If the employee has protected access of shop class variables or has their access through a getter function

        a) +0.5 marks
6) InterfaceA/Abstract_classA with method declaration “public void serviceA()”
    a) +0.5 marks
7) InterfaceB/Abstract_classB with method declaration “public void serviceB()”
    a) +0.5 marks
8) Three things come for service (classes TypeX, TypeY, and TypeZ)
    a) +0.75 marks
9) class TypeX implements/extends InterfaceA/Abstract_classA and provide concrete implementation of serviceA
    a) +0.5 marks
10) Class TypeY implements/extends InterfaceA/Abstract_classA and InterfaceB/Abstract_classB and provide concrete implementations of methods serviceA and serviceB
    a) +1 marks
11) Class TypeZ implements/extends InterfaceB/Abstract_classB and provide concrete implementation of serviceB
    a) +0.5 marks
12) All overridden methods annotated with @Override
    a) +0.5 marks

Q2) Correct the following program. It must follows all principles/coding practices of OOP **[3 marks]**

public class Main{
    public static void main(String[] args){
        /* If some student removes the parameterized constructor in Toads then NO need to pass String type name here Hence, marking scheme as follows for below
            a) If parameterized constructor removed from Toads
               OR
            b) String name passed here correctly
                i) +1 marks
        */
        Amphibians A1 = new ~~Toads();~~ Toads(“abc”)
    OPTION-A
        ~~Amphibians A2 = new Amphibians();~~
        /* Toads on both LHS and RHS: NO MARKS HERE
        Toads A2 = new Toads(“def”);
    OPTION-B
        Amphibians A2 = new Toads(“def”); //only string passed as parameter
        A1.jump(A2);
        A1.eat();
    }

```
}

public interface Amphibians{
        public void eat();
        OPTION-A
        ~~private~~ public void jump(~~Amphibians~~ Toads A); // +1 marks
        OPTION-B
        ~~private~~ public void jump(Amphibians A); // +1 marks
}

public class Toads implements Amphibians {
        private String name;
        public Toads(String n){name = n; }

        public void eat(){
                System.out.println("Toads Eating");
        }
        OPTION-A
        public void jump(~~Amphibians~~ Toads A){ // +1 marks
                System.out.println(this.name+" jumping with " +A.name);
        }
        OPTION-B
        public void jump(Amphibians A){
                System.out.println(this.name+" jumping with " +((Toads)A).name); // +1 mark
        }
}
```