Assignment Report for

Course and Section	CSC.215
Assignment Name	Assignment 05
Due Date and Time	November 1st @ 11:55 PM
First Name and Last Name	AJ Arce
SFSU Email Account	Aarce3@sfsu.edu
First Name and Last Name of Teammate	[First Name] [Last Name]
SFSU Email Account of Teammate	[SFSU Email Account]

PART A

Question Description and Analysis:

This part of the assignment asks that I cover one of the 8 Class Design Guidelines and explain all its bullet points in one detail. I need to use at least one-half of a page to explain the guideline. Then I have to write code to demonstrate my application of the guideline in designing a class. Then with the code as an example, to explain how I applied the guideline inside of it.

Answer:

The guideline I chose was guideline 7, Inheritance vs Aggregation. There are two key concepts used to define relationships between classes. Inheritance represents an “IS-A” relationship, this is where one class extends another, inheriting its properties and behaviors. For example, if a **Student** class extends an **Human** class, it implies that a **Student** IS-A **Human,** inheriting methods and attributes from the **Human** class while possible adding some of its own. This promotes reuse of code and supports polymorphism, which allows subclasses to be used interchangeably with their parent classes. Then we have aggregation which represents a “HAS-A” relationship, which is when one class contains another class as a component. For example, a **Car** class might have an attribute of type **engine** indicating that a **Car** has a **steeringWheel**. Aggregation emphasizes composition, which is when objects are combines to build more intricate types.

Example:

```
package zooAnimals;


class Animal {
   private String nameOfAnimal;
   private int ageOfAnimal;

   public Animal(){
       nameOfAnimal = "N/A";
       ageOfAnimal = 0;
   }

   public Animal(String name, int age){
       this.nameOfAnimal = name;
       this.ageOfAnimal = age;
   }

   public void makeSound() {
       System.out.println("Animal makes a sound");
   }

   public String getName() {
       return nameOfAnimal;
   }
   public int getAge(){
       return ageOfAnimal;
   }
```

```java
}
package zooAnimals;

class Dog  extends Animal{
    private String breed;
    public Dog(String name, int age, String breed){
        super(name, age);
        this.breed = breed;
    }

    @Override
    public void makeSound() {
        System.out.println(getName() + " barks at you!");
    }

    public void fetch(){
        System.out.println("Dog fetches ball");
    }

    public String getBreed(){
        return breed;
    }
}



package zooAnimals;

public class Main {
    public static void main(String[] args) {
        Dog bullDog = new Dog("Honey", 6, "English Bulldog");
        bullDog.makeSound();
    }

}

package carDealership;


class Engine {

    private String type;

    private int horsepower;


    public Engine(){

        this.type = "Unknown";

        this.horsepower = 0000;

    }


    public Engine(String type, int horsepower) {

        this.type = type;

        this.horsepower = horsepower;

    }


    public void startEngine() {

        System.out.println("Engine starts with " + horsepower + " horsepower.");

    }


    public String getType() {

        return type;

    }


    public int getHorsepower() {

        return horsepower;

    }



}

package carDealership;
```

```java
class Car {
   private String model;
   private Engine engine;

   public Car(String model, Engine engine) {
       this.model = model;
       this.engine = engine;
   }

   public String getModel() {
       return model;
   }
   public Engine getEngine() {
       return engine;
   }

   public void startCar() {
       System.out.println("Starting the car: " + model);
       engine.startEngine();
   }
}
package carDealership;

public class Main {
   public static void main(String[] args) {
       Engine engine = new Engine("V8", 400);
       Car car = new Car("Porsche", engine);
       car.startCar();
   }
}
```

In the code above, the guideline of using inheritance and aggregation is shown through the Dog, Animal, Car, and Engine classes. The Dog class extends the Animal class which shows the IS-A relationship between the two classes. A dog class is a type of Animal class. This allows the Dog class to inherit the data fields and methods of the Animal class such as nameOfAnimal, ageOfAnimal, and makeSound(). The dog class also adds its own attribute, breed, and its own behavior, fetch(). It also overrides the makeSound() method to provide a dog-specific behavior, barking. The constructor in Dog uses super(name, age) to call the superclass constructor which makes sure that the inherited fields are correctly initialized.

The aggregation relationship is shown through the Car and Engine classes. The Car class contains an Engine object which demonstrates a HAS-A relationship where a Car has an Engine. This code allows the Engine class to be reused independently of the Car. The Car constructor initializes the Engine instance through its parameters, which chows how aggregation is use to build complex objects from more simple ones. The Car class also interacts with the Engine class through method calls like engine.start(), this shows how objects work together in an aggregated relationship.

PART B

Question Description and Analysis:

This part of the assignment asks that I create a program to demonstrate my understanding of the HAS-A relationship between two classes. My program has to consist of at least 5 classes. A driver class with the main method and 4 user-defined classes. The program must be organized and the main method in the driver class must consist of 1 statement. It also asks that I create a diagram of the class relationships among the classes I created. I must explain my understanding of the problem that I'm asked. I'm also asked to analyze my results of the program and explain the analysis.

Answer:

1. The problem requires creating a Java program that is able to demonstrate my understanding of the HAS-A relationship between classes. The program muse use at least five classes, one driver class and four user-defined data type classes. One class must contain properties of the other three classes. The program should show this relationship, be organized into five separate .java files, and the main method should only contain one method call. I plan to create a primary class that aggregates the other three classes. Each class will have attributes and methods relevant to its role. The solution will be structured with each class in its own file, and I will pay attention to keeping the relationships and interactions clear and logical. The program's output and a simple diagram will illustrate the correct use of the HAS a relationship.

3. The program compiles successfully and meets the client's requirements by demonstrating the HAS-A relationship using the Traveler class. This class aggregates instances of Luggage, Passport, and Ticket. The toString method provides a clear and detailed representation of a traveler's details. It also makes sure that the information is properly displayed if all the components were set correctly. This works well because each class is structured, encapsulated, and provides appropriate methods for accessing and modifying its fields. One potential issue is the lack of validation in the setter methods of the Traveler class. This could lead to incorrect data being set like negative ages or dates that don't make sense. In order to solve this issue, I plan to

implement validation checks in the setters of all classes to ensure data integrity. I'm also considering using Java's data classes for better handling of dates and to make the code more robust-future proof. These enhancements will increase the program's reliability and maintainability as well as meets the requirements.

Screenshots of Outputs and Explanation:

These screenshots show what I accomplished…

PART C

Question Description and Analysis:

This part of the assignment asks that I show my understanding of class relationship inheritance IS-A through a java program. This program must consist of at least 4 classes: a driver class that contains the main method and 3 user-defined data type classes that are related by the 3-level class relationship inheritance IS-A. I'm supposed to analyze the problem and explain my understanding of the problem and how I plan to solve the problem. After creating the program it asks that I analyze the result and figure out what I could have done better, what could be improved and how.

Answer:

1. The problem asks that I create a program that demonstrates the relationship inheritance between classes. Specifically, the IS-A relationship. I must use a 3-level inheritance

structure within my program. The client wants the program to have at least four classes that illustrate how inheritance works between classes. The program should also include a driver class where the main method contains only one statement, which calls a method to run the program. Key details for this program include ensuring that each subclass appropriately inherits attributes and behaviors from its parent class while making its own specific attributes and methods. The solution involves organizing classes in a logical manner. This helps maintain a clear inheritance chain and makes sure that all methods and attributes are correctly accessed or even overridden when needed. When coding, attention should be directed towards the encapsulation of attributes, the correct use of the super keyword, and making sure that the driver class is able to showcase the program properly.

3. The program is able to compile successfully and I believe it meets all of the client's requirements by demonstrating a 3-layer inheritance structure as well as proper usage of the IS-A relationship throughout the classes. The implementation shows the inheritance of attributes and methods properly, and the driver class is able to demonstrate the functionality of the code with a single method call within the main method. What works well is the structure of the class hierarchy and the encapsulation of attributes. This makes sure that each subclass is able to inherit and extend from its superclass properly. There are a lot of improvements that could be made to the program. I believe error handling and adding in more validation checks, especially for the user inputs, should be a great addition to prevalent unrealistic values. Additionally, expanding the features within the LuxuryElectricCar class could be nice as well. In the future, the program

could possible integrate an interface or abstract class to allow for even more flexibility within the design of the code.

Screenshots of Outputs and Explanation:

These screenshots show what I accomplished…

PART D

Question Description and Analysis:

This part of the assignment asks that I create a java program to show that I understand both IS-A and HAS-A relationships between classes. I must create this program with at east 15 classes. 1 driver class, 3 user-defined data type classes that are related by the 3-layer class relationship, and 11 user-defined data type classes that are used to show the HAS-A relationships, both aggregation and composition. Then I must analyze the problem and show I understand then another analysis about the result and how it could be improved.

Answer:

1. This problem asks that I create a Java program that shows my understanding of both IS-A and HAS-A relationships using at least 15 classes. The client expects a 3-layer inheritance structure for IS-A relationships and a mix of both composition and aggregation for HAS-A relationships. The program also requires a driver class whose main method calls a single demonstration method. In order to meet these requirements, I plan to design a system, like a University or Hotel management system, that is able to incorporate these relationships easily. I plan to create a base class with multiple levels of subclasses for IS-A, and additional associated classes to demonstrate HAS-A. The key

elements for this program include organized class hierarchy, encapsulation, and proper constructor calling. My focus when coding will be on the readability, class structure, and making sure that the relationships are clear. This shows that I have a pretty solid grasp of object-oriented principles.

3. The program is able to compile and run properly as well as meet the client's requirements by creating a program that is able to incorporate both IS-A and HAS-A relationships. The design uses inheritance to represent different types of people inside of the hotel, while the HAS-A relationships are used to associate different classes. What works well with the program is the structure and organization of the classes. Each class serves a purpose and methods like bookRoom() and scheduleEvent() help simulate real-world hotel operations. With that being said, there is still a bunch of areas that could be built upon. For example, some validation methods or error handling could be worked on a bit more, especially when it comes to the reservations, booking, or the spa availability. There could definitely be some more preventions to logical errors or inconsistencies. The program also lacks some data storage but I still have yet to learn how to add that to a program. In the future when I learn how to do that I could comeback to the program and integrate a database for storing all of the information about the hotel. Such improvements could make the program more efficient and scalable.