**Batch - 29 Has A Relation Questions (Lab Practice)**

Ques - 1

-----------

Ques - 1

-----------------

Question:

Imagine you are designing a Java system to model the relationship between a Car and its underlying Engine components to illustrate the concept of composition. The Car class represents an automobile, and the Engine class encapsulates the essential features of the engine powering the car.

Instructions:

Create a Car class with private attributes for the car's make, model, and year of manufacture. Implement a parameterized constructor to set these attributes during the car's instantiation.

Develop an Engine class with private attributes for the engine's capacity, fuel type, and horsepower. Implement a parameterized constructor for the Engine class.

Establish a HAS-A(composition) relationship between the Car and Engine classes by including an instance of the Engine class as a private attribute within the Car class. Ensure that a Car object is composed of an Engine object.

Write a method in the Car class that allows for starting the car, simulating the activation of the underlying engine. This method should internally invoke a method in the Engine class to initiate the engine startup.

In the main method of a separate class, create an instance of a Car with specific make, model, and year attributes, along with an associated Engine instance that defines the engine's characteristics. Demonstrate the composition relationship by invoking the car's start method and observe the coordination with the underlying engine.

Note: Follow Java naming conventions, employ proper encapsulation, and provide necessary accessor methods.

**class** Car//automobile

{

String brand;

String model;

**int** yearOfManufacture;

**private** Engine e;

**public** Car(String brand, String model, **int** yearOfManufacture,Engine e) {

**super**();

**this**.brand = brand;

**this**.model = model;

**this**.yearOfManufacture = yearOfManufacture;

**this**.e=e;

}

**void** starting()

{

**if**(e==**null**)

{

System.***err***.println("Engine not started");

}

**else**

{

System.***out***.println("Engine start");

}

}

@Override

**public** String toString() {

**return** "Car [brand=" + brand + ", model=" + model + ", yearOfManufacture=" + yearOfManufacture + ", e=" + e

+ "]";

}

}

**class** Engine

{

**int** capacity;

String fuelType;

**int** horsepower;

**public** Engine(**int** capacity, String fuelType, **int** horsepower)

{

**super**();

**this**.capacity = capacity;

**this**.fuelType = fuelType;

**this**.horsepower = horsepower;

}

@Override

**public** String toString() {

**return** "Engine [capacity=" + capacity + ", fuelType=" + fuelType + ", horsepower=" + horsepower + "]";

}

}

**public** **class** MyProgram

{

**public** **static** **void** main(String[] args)

{

Engine e=**new** Engine(25,"petrole",10000);

Car c=**new** Car("Audi","A8",2024,e);

c.starting();

System.***out***.println(c);

Car c1=**new** Car("LAmbo","HArican",2016,**null**);

c1.starting();

System.***out***.println(c1);

}

}

**OUTput:-**

Engine start

Engine not started

Car [brand=Audi, model=A8, yearOfManufacture=2024, e=Engine [capacity=25, fuelType=petrole, horsepower=10000]]

Car [brand=LAmbo, model=HArican, yearOfManufacture=2016, e=null]

Ques - 2

--------------

Ques - 1

---------------

Question:

As a Java developer working on an e-commerce platform, you are tasked with designing a system that models the relationship between a Customer and their associated ShoppingCart. The objective is to illustrate the concept of HAS-A, showcasing how a Customer is linked to a ShoppingCart that encapsulates the items they intend to purchase.

Instructions:

Create a Customer class with private attributes for the customer's name, email, and unique customer ID. Implement a parameterized constructor to set these attributes during the instantiation of a Customer object.

Develop a ShoppingCart class with private attributes for the cart ID and a list to store the items that the customer wishes to purchase. Implement a parameterized constructor for the ShoppingCart class that initializes the cart ID.

Establish a HAS-A relationship between the Customer and ShoppingCart classes by including an instance of the ShoppingCart class as a private attribute within the Customer class. Ensure that a Customer object is associated with a specific ShoppingCart object, emphasizing the aggregation nature of this relationship.

Implement a method in the ShoppingCart class to add items to the cart. This method should take parameters representing the items to be added and manage the list of items within the cart.

In the main method of a separate class, create an instance of a Customer with a unique ID, name, and email. Also, instantiate a ShoppingCart associated with this customer. Demonstrate the aggregation relationship by adding various items to the customer's shopping cart and displaying the contents of the cart.

Note: Follow Java naming conventions, employ proper encapsulation, and provide necessary accessor methods.