Relationship among classes

Following are the relationships described for these classes also represented in the above UML class diagram in order to make our system function properly and perform the desired tasks of add, delete, search and modify on the employees and cars or make new sales.

Inheritance

Inheritance is a relationship among classes in which on class inherits the attributes and functionality of its parent or super class. It is typically defined as ` Is a` relationship. In our particular case we can say that `Manager` is an `Employee` and also `Sales Man` is an `Employee`. Meaning that the Employee class is the parent class for the Manager class and Sales Man class. We have used this relationship in our diagram to achieve the DRY (do not repeat yourself) principle. Because the Manager and Sales man are both employees and have some same attributes therefore, we used inheritance relationship among these classes

Composition

In a class diagram, a composition relationship is a particular kind of relationship where one class is made up of one or more instances of another class. It is impossible for the component class to exist separately from the container class. So, in our particular case the sales man cannot exist without a manager because for our system we assumed that every sales man must have a manager. Meanwhile, the sale can not be made if that particular car doesn't exist or even if we do not have a sales man so we can not make a sale. Therefore, a composition relationship has been put among these classes. And the multiplicity has also been added in the diagram. From diagram we can see that one sale can only be made by one employee while on the other hand n employee can may have 0 or more sales. Same with the car i.e., a car can may have 0 or more sales but in a single sale there should be exactly one car(assumption). And for sales man and manager we can see in the diagram that one manager can have at least one or more sales man under him but a sales man can exactly have one manager.

Association

Association is the simplest relationship among the classes as it shows some kind of simple association among them. For instance, in our case, we have associated our sales man, manager, sales and car to a driver class which manages all these classes. We have made lists of sales mans, managers, sales and cars in our driver class in order to keep track of these classes. As an instance of driver class has been created within the GUI class which is managing or we can say manipulating all the data using the lists and functions in the driver class. It was done like this because we always would need to manipulate these classes somewhere and if we do it without a driver class, we would have to done a lot more complex coding. So, there driver class was made to achieve the simplicity in our code.