# First question:

Does your class diagram respect or violate SOLID principles? Justify your answer

# Answer:

Yes, it almost respect all the SOLID principles.

Single responsibility:

Each class has only one reason to change

Examples: Parking class , Task class

Open-Closed:

Each class is open for extension but closed for modification.

Liskov Substitution:

Each child class can be used instead of its parent class

Examples: we can use "parkingout" class instead of "ParkingController" class

Interfaces Segregation:

Each class depend on code that it needs

Examples: the "Entering Details" class only has the "execute task" function

Dependency inversion:

Bad news

Some classes violate this principle(parking-vehicle owner-vehicle)

(garage owner- task)

# Second question:

Does your class diagram contain any design pattern(s), if yes name it and list the names of the classes involved in such pattern(s).

# Answer:

Yes.

Pattern: **Strategy** Design Pattern.

List of Classes:

Group one:

-TaskController(Interface-parent).

-Entering Details.

-Displaying\_Slots.

-Displaying\_Total Income.

-Displaying numofVehicles.

Group two:

-ParkingController(Interface-parent).

-ParkingOut.

-BestFit.

-FirstCome.