**Solid Principals:**

Here we will start taking every single class in the class diagram to see if it follows SOLID Principals or not.

1. Ticket:

as we see in ticket class that it follows open/closed principle as it carries every single detail about user experience and can be extended to more specifications, but it can`t carry any information that is unnecessary for user as car specifications or any other inner information for the garage.

1. Vehicle:

In this class we found that it also follows open/closed principle as it also carries only information and method that belongs to the car not any other information.

1. ParkingLot:

In this class we see that it follows single responsibility principle as it does every single method by itself as we consider this class as the garage itself, so it does take size of how many slots will be created and creates it capture and free the car from it .

|  |  |  |
| --- | --- | --- |
| Method | Follows | Violates |
| getSlot () | 🗸 |  |
| setSlot () | 🗸 |  |
| removeCar () | 🗸 |  |

1. Context:

In this class we see that it follows single responsibility and interface segregation as in this class it manage the interface that user choose to go with it to choose just one interface so we provide more options in many interfaces to not violate the ISP

We also provide SRP as the class goes for every single method for itself

|  |  |  |
| --- | --- | --- |
| Method | Follows | Violates |
| searchSlot () | 🗸 |  |

1. BestFit & FirstCome & TotalCost & Income:

Follows SRP

1. BestFit

|  |  |  |
| --- | --- | --- |
| Method | Follows | Violates |
| searchSlot (Slot [], vehicle) | 🗸 |  |

1. FirstCome

|  |  |  |
| --- | --- | --- |
| Method | Follows | Violates |
| searchSlot (Slot [], vehicle) | 🗸 |  |

1. TotalCost

|  |  |  |
| --- | --- | --- |
| Method | Follows | Violates |
| getTotalCost (int totalcost , int cost) | 🗸 |  |

1. Income

|  |  |  |
| --- | --- | --- |
| Method | Follows | Violates |
| getTotalIncome () | 🗸 |  |
| addFees (int fees) | 🗸 |  |

6. ParkingCtrl:

We use this class as a controller of all the system so it must do exactly what it has from methods, so it follows SRP as show below

|  |  |  |
| --- | --- | --- |
| Method | Follows | Violates |
| Parkin () | 🗸 |  |
| Parkout () | 🗸 |  |
| TotalIncome () | 🗸 |  |
| Slot () | 🗸 |  |
| numberOfVehicles () | 🗸 |  |

7. Screen:

We use this class as a boundary class that user can react with so it follows ISP & OCP as it first opened for extensions for more new features closed for modification and it also has a single interface so user can deal with.

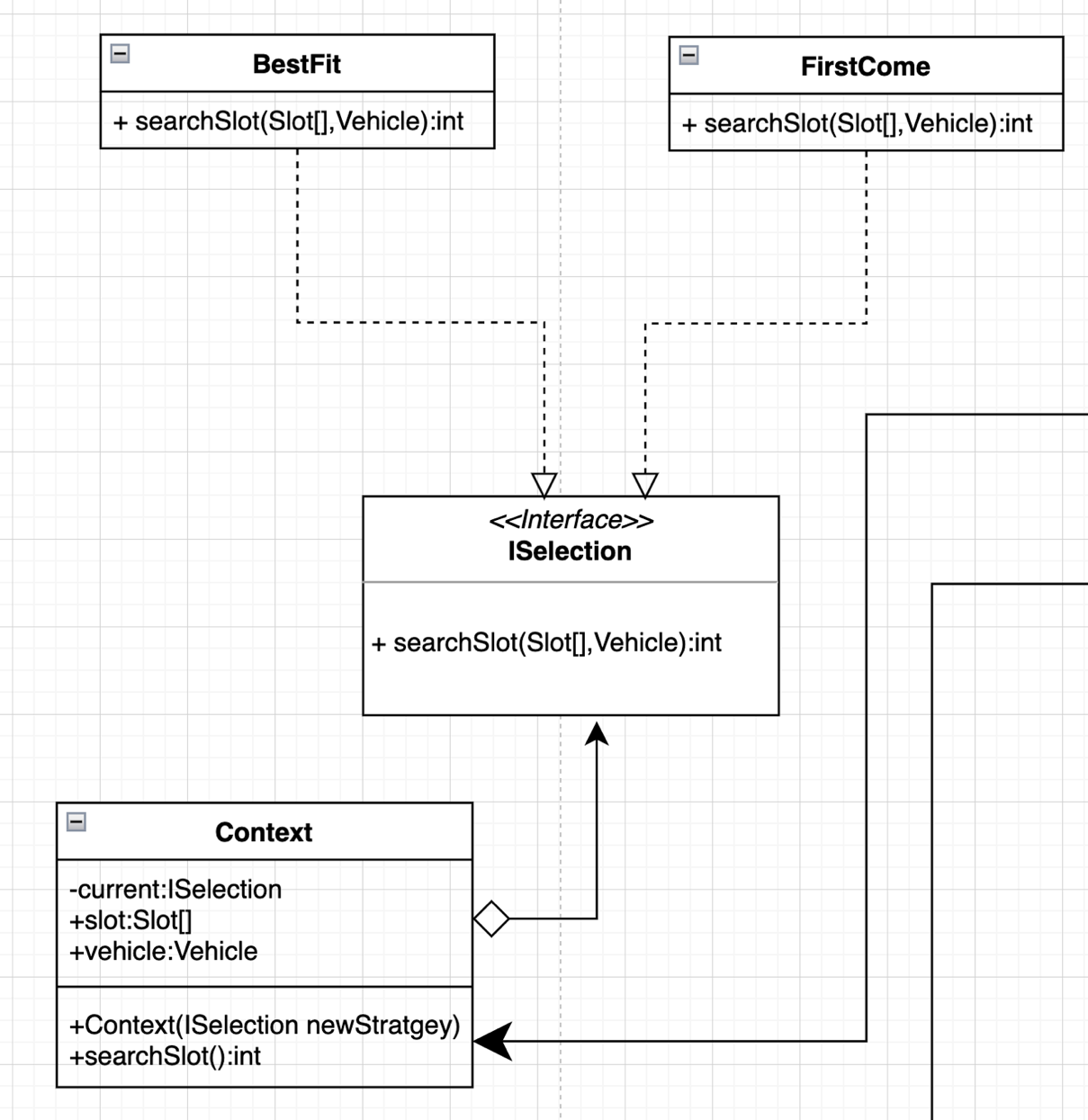
8. Slot:

This class carries infrastructure for each slot itself so it must follows OCP as it is opened for any Extension to carry any information that helps to form the slots and form our garage and it also closed modification as any change in the infrastructure can change the system functionality as total.

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).

Yes it does,

1-Strategy design pattern

Classes included :- Context ,ISelection ,FirstCome ,BestFit

2-Singleton design pattern

Classes Included :- Income

