1. Design concepts

* Coupling:
* Data coupling: use exactly all information in data passing in arguments. E.g: pay(Interbank request)
* Stamp coupling: only use part of argument data. E.g: displayBikeList using List<Bike> with full data but only use Bike Type and battery
* Control coupling: data passed decided one module. E.g: getStringBatteryStatus (type of bike decide if the information of bike battery is n/a of calling bike.getBatteryStatus))
* Common coupling: two or more functions share global data. An example of global coupling would be global information status regarding an operation, with the multiple modules reading and writing to that location. E.g: Order object singleton
* Cohesion:
* Sequential: output is used for input
* Functional: All processing elements of a module are essential to the performance of a single and well-defined task. E.g: entity define object, dao class to communicate with database
* Logical: service classes, operate service on that type of data
* Temporal: screen handler classes, related by the time the screen is on, and controller class, which is used by screen handler

1. Design principles

* Single responsibility principle
* Open-closed principle: open for extension, closed for modification. E.g: if we want to change rent bike fee calculate method, we just need to create new class and implement PaymentServiceInterface and passing it as an argument (also using strategy design pattern help)
* Liskov substitution principle: object can be replaced by instance of their subtype. E.g: no extends class, interface implemented can be replaced by class implements it
* Dependency inversion principle: Modules communicate with others by interface. E.g: Service and Controller, Service and ObjectDAO interface

1. Design patterns

* Strategy: create service interfaces in case of changing methods
* Singleton: class Order
* Data access object: class only for communicate with database. This way, the service remains completely in dark about how the low-level operations to access the database is done. This is known as the principle of **Separation of Logic**.