|  |  |
| --- | --- |
| OOPs Design Principles | |
| Minimize Coupling (Always Loose Coupling) | |
| Coupling between modules/components is their degree of mutual interdependence; lower coupling is better. In other words, coupling is the probability that code unit “B” will “break” after an unknown change to code unit “A”. | |
| TIGHT COUPLING | in general, Tight coupling means the two classes often change together. In other words, if A knows more than it should about the way in which B was implemented, then A and B are tightly coupled. |
| LOOSE COUPLING | In simple words, loose coupling means they are mostly independent. If the only knowledge that class A has about class B, is what class B has exposed through its interface, then class A and class B are said to be loosely coupled. In order to overcome from the problems of tight coupling between objects, spring framework uses dependency injection mechanism with the help of POJO/POJI model and through dependency injection it’s possible to achieve loose coupling. |
| WHICH IS BETTER ? | In general, Tight Coupling is bad in but most of the time, because it reduces flexibility and re-usability of code, it makes changes much more difficult, it impedes test ability etc. loose coupling is a better choice because A loosely coupled will help you when your application need to change or grow. If you design with loosely coupled architecture, only a few parts of the application should be affected when requirements change.   * Tight coupling is not good at the test-ability. But loose coupling improves the test ability. * Tight coupling does not provide the concept of interface. But loose coupling helps us follow the GOF principle of program to interfaces, not implementations. * In Tight coupling, it is not easy to swap the codes between two classes. But it’s much easier to swap other pieces of code/modules/objects/components in loose coupling. * Tight coupling does not have the changing capability. But loose coupling is highly changeable. |