|  |  |
| --- | --- |
| Single Responsibility Principle | -A class should have one responsibility and only ONE reason to change.  -Testing: far fewer test cases  -Lower coupling: less functionality will have fewer dependencies  -Organization: easier to search  -Example: printing out Book -> different class |
| Open-Closed Principle | -Objects should be open for extension but closed for modifications  -Means that a class should be extendable without modifying the class itself  -By extending the Guitar class, we can sure that our existing application won’t be affected |
| Liskov Substitution Principle | -Every subclass should be substitutable for their base class  -If A is subtype of B, we should be able to replace B with A without disrupting the behavior of our program  -Class that inherits Car class without an engine (like electric) will throw an error and violate this principle |
| Interface Segregation Principle | -Client should never be forced to implement an interface that it doesn’t use, or clients shouldn’t be forced to depend on methods they do not use  -Bear class, split up the interfaces to only what you want to do (feed, wash) and leave the pet to the crazy person |
| Dependency inversion Principle | -Decoupling of software modules. This way, instead of high level modules depending on low level, both will depend on abstraction  Baeldung: Windows98 class  -don’t use new because we then tightly coupled these 3 class together |
|  |  |
|  |  |
|  |  |