1. **SOILD固态石墨 [CRP-SRP-OCP-ISP-LSP-LOD-DIP]**

**合成复用原则**(组合/聚合复用原则)

+ Composite/Aggregate Reuse Principle

+ **尽量使用对象组合/聚合，而不是继承来达到复用的目的**

**单一职责原则**

+ Single Responsibility Principle

+ 应该只有一个原因能引起类的变化，接口一定要做到单一职责，JS中，一个对象(方法)只做一件事。**一个类负责一项职责**

**开闭原则**(最基础的一个原则)

+ Open-Closed Principle

+ 软件实体是可扩展的**(对扩展开放)**，但是不可修改**(对修改关闭)**

+ 尽量在不修改原有代码的情况下进行扩展。

**接口隔离原则**

+ Interface Segregation Principle

+ 建立多个专门的接口，不要建立一个臃肿庞大的总接口。**细化接口，接口中方法尽量少。**与单一职责原则的区别： SRP注重职责，业务逻辑上的划分；接口隔离原则注重方法数量

+ 为接口定义了规范

\* 接口要小，且不违背SRP

\* 接口高内聚

\* 定制服务

\* 接口设计有限度：粒度越小，越灵活，同时也更复杂

**里氏替换原则**(里氏代换原则是实现开闭原则的重要方式之一)

+ Liskov Substitution Principle

+ 所有引用基类的地方必须能透明地使用其子类的对象，**是继承与派生的规则**

+ 为继承/派生定义了规范

\* 子类必须完全实现父类的方法

\* 子类可以有自己的个性

\* 覆盖或实现父类的方法时输入参数可以被放大(前置条件可以宽松)

\* 覆盖或实现父类的方法是输出结构可以被缩小(后置条件可以苛刻)

**迪米特法则**(最少知识原则)

+ Law of Demeter/LeastKnowledge Principle

+ 一个软件实体尽可能少地与其他实体发生相互作用，**实现低耦合高内聚**

\* 软件实体：包括对象、系统、类、模块、函数、变量等

+ 系统设计时可以通过引入一个合理的第三者来降低现有对象之间的耦合度

+ 一个对象的朋友可以是： \* this\以参数形式传入到当前对象方法中的对象\成员对象\成员对象为集合时，集合中的元素\ this所创建的对象

**依赖倒置原则**

+ Dependency Inversion Principle

+ 高层模块不依赖底层模块，两者都依赖其抽象，抽象不依赖于细节，细节依赖于抽象

**针对接口编程，而不是针对实现编程**

+ 开闭原则是目标，里氏代换原则是基础，依赖倒转原则是手段，它们相辅相成，相互补充，目标一致