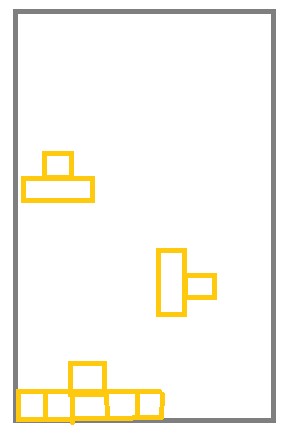
单一职责原则：

1. 就一个类而言，应该仅有一个引起它变化的原因。
2. 大多数时候，一件产品单一些，职责单一一些，或许是更好的选择。
3. 如果一个类承担的职责过多，就等于把这些职责耦合在一起，一个职责的变化可能会削弱或者抑制这个类完成其它职责的能力，这种耦合会导致脆弱的设计，当变化方生时，设计会遭受到意想不到的破坏。
4. 软件设计真正要做的许多内容，就是发现职责并把那些职责相互分离。



开放-封闭原则

1. 即软件实体（类、模块、函数等等）应该可以扩展，但是不可修改。 //HK
2. “对扩展是开放的” ， “对修改是封闭的”//JSQ
3. 设计软件要容易维护又不容易出问题的最好的方法，就是多扩展、少修改。
4. 等到变化时立即采取行动。
5. 在我们最初编写代码时，假设变化是不会发生的。当变化发生时，我们就创建抽象隔离以后发生的同类变化。
6. 面对需求，对程序的改动是通过增加代码进行的，而不是更改现有代码。

迪米特法则：

1. 如果两个类不必彼此直接通信，那么这两个类就不应当发生直接的相互作用，如果其中一个类需要调用另一个类的某一个方法的话，可以通过第三者转发这个调用。
2. 在类的结构设计上，每一个类都应当尽量降低成员的访问权限。
3. 迪米特法则强调了类之间的松耦合。类之间的耦合越弱，越有利于复用，一个处在弱耦合的类被修改，不会对有关联的类造成波及。

合成/聚合复用原则：

1. 尽量使用合成/聚合，尽量不要使用继承。
2. 优先使用对象的合成/聚合将有助于你保持每个类被封装，并被集中在单个任务上。这样类和类的继承层次就会保持较小的规模，并且不太可能增长为不可控制的庞然大物。