5S哲学

整理（Seiri）

整顿（Seiton）

清楚（Seiso）

清洁（Seiketsu）

身美（Shitsuke）

1. ：整洁代码

为什么要整洁代码

糟糕的代码究竟有什么危害

混乱的代价

简单代码，依其重要顺序：

能通过所有测试

没有重复代码

体现系统中的全部设计理念

包括尽量少的实体，比如类、方法、函数等

1. ：有意义的命名

取个好名字的的规则

名副其实

（如果名称需要注释来补充，那就不算是名副其实）

避免误导

做有意义的区分

废话都是冗余

例子（variable一词永远都不应该出现变量名中，Table一词永远不当出现在表名中）

犯错的代码

getActivityAccount（）

getActivityAccounts（）

getActivityAccountInfo（）

如果缺少明确的约定，God knows调用哪个函数。要区分名称，就要以读者能够鉴别不同之处的方式来区分

2.5使用读得出来的名称

如果名称读不出来，那讨论的时候就会像个傻鸟

2.6使用可搜索的名称

单字母变量和数字常量有一个问题，就是很难在一大篇文字中找出来

单字母应该仅用于短方法中的本地变量，名称长短应与其作用域对应

2.7避免使用编码

2.7.1匈牙利语标记法

2.7.2成员前缀

也不必使用m\_前缀来标明成员变量。应当把类和函数做得足够小，消除对成员前缀的需求

2.7.3接口与实现

2.8避免思维映射

2.9类名

类名和对象名应该是名词或名词短语，类名不应该是动词

2.10方法名

方法名应该是动词或动词短语

属性访问器、修改器和断言应该根据其值命名

重载构造器时，使用描述了参数的静态工厂方法名

2.11别扮可爱

不要使用哪些与文化紧密相关的词

言到意到。意到言到。

2.12每个概念对应一个词

给每个抽象概念选一个词，并且一以贯之

例如，同一堆代码中有controller，又有manager，还有driver就会令人感到困惑

2.13别用双关语

避免将单一单词用于不同的目的。

2.14使用解决方案领域的名称

记住，只有程序员才会读你的代码。尽量用那些CS术语、算法名、模式名、数学术语

2.15使用源自所涉问题领域的名称

如果不能用程序员所熟悉的术语来给手头工作命名，就采用所涉问题领域的名称吧。

分离解决方案领域和问题领域的概念

2.16添加有意义的语境

很少有名称能够自我说明的。你需要用有良好命名的类、函数或命名空间来放置名称，给读者提供语境。如果没能这么做，那么给名称添加前缀就是最后一招了。

2.17不要添加没用的语境

1. 函数

3.1短小

3.2只做一件事

函数应该做好一件事，只做一件事

3.3每个函数一个抽象层级

自顶向下读代码，向下原则

3.4switch语句

一般利用多态来实现

3.5使用描述性的名称

如果每个例程都让你感觉深合己意，那就是整洁代码

函数越短小，功能越集中，就越便于取个好名字

命名方式要保持一致，使用与模块名一脉相承的短语、名词和动词给函数命名

3.6函数参数

参数个数越少越少，最好不要超过3

测试上，如果多于两个参数，则其中覆盖组合过多，令人止步

3.6.1一元函数的普遍形式

3.6.2标识函数

向函数传入布尔值违背了函数只做一件事的原则，宣布函数不止做一件事，如果为true做这件事，false做另一件事

3.6.5参数对象

如果函数看来需要两个、三个或者三个以上参数，说明这其中一些参数可能需要封装成一个类

3.6.7动词和关键字

给函数取个好名字，能较好地解释函数的意图。对一元函数来说，函数和参数应当形成一种非常良好的动词名词对的形式

3.7无副作用

副作用是一种谎言。函数可能会做其它被藏起来的事。

3.8分隔指令与询问