抽象

1. 找出key 特徵加以desc
2. 簡化模型以協助思考與運用

就像我能簡單說明,簡單的範例

先想有無的方法可解,再想可行與否

程式不內聚,但意義內聚

藕合抽象,不要藕合實體e.g.Math static 沒有互動的relation

[Factory] 不要在客戶端new 實體

Factory回傳一個抽象

O Factory.getObject(“A”)

X ITest c = new ITest(); //不要在客戶端new 實體

[繼承]

強藉合,父動,子牽連.(動了就要test)

重用是繼承的重點

剛有這功能時,大家很愛用,但用多了,缺點就出來了

Data封裝

變化封裝(封裝類別) : 把value丟進來,我就給你result,(把how算,這種變化封起來)

**奧坎剃刀原則**

2個code都是same result

Effect最小,是比較OK的

較簡單的理論比複雜的理論更好，因為它們更加可檢驗。[[](https://zh.wikipedia.org/wiki/%E5%A5%A5%E5%8D%A1%E5%A7%86%E5%89%83%E5%88%80#cite_note-fn_.28109.29-9)

用較少的東西，同樣可以做好的事情

變動時,變動最小(code變動力

動到原code,就要重test (成本/時間)

所以如果是用加的,就好多了

加一塊,拿掉一塊

具體 : switch case杳表法

抽象 : 16.75/基數,有餘數,then 商+1 (公式就是一種抽象)

X位元運算值

X二元運算

Math.ceilns(value/base value)\*basevalue

<T> 型別參數

強制同型,complier才比較有用(怎麼有用?)

Compare(<T>,Tx,Ty) where T:Icompable<T>