註解

真相只有~~一個~~程式碼，永遠不會是註解。

程式碼會被維護，註解容易隨著時間流逝漸漸變成**不準確的謊言**。

註解是一種用來彌補程式碼表達意圖**失敗**的手段，是一種**必要之惡**。

- **註解無法彌補糟糕的程式碼**

寫註解的動機有很多種，其中一種是「這程式碼寫得不好懂，我該加註解」。

但即使加上註解，原本的程式碼還是不好懂RRR ~~廢物界的霸主，還是廢物~~。

若情況允許，你該重寫它。

- **用程式碼表達你的本意**

```
// 確認註冊的使用者有填寫姓名且成年
if (customer != null && customer.getName() != null
        && customer.getName().trim().length() > 0
        && customer.getAge() > 18) {

}
```

```
if (customer.isNameAndAdultValid()) {

}
```

左圖一個有加註解，一個沒加註解。
你認為讀者喜歡看到哪一種程式碼呢？

- **對意圖的解釋**

```
public int compareAge(Object obj) {
    // do something

    return 1; //某種概念上較大的物件
}
```

將難以聯想的回傳值或是參數翻譯成可讀的解釋。
如果情況允許使用更有表達力的變數名稱會是更好選擇。
如果是不能修改的程式，此種註解就會非常有用。

- **干擾型註解**

```
// 名字
private String name;
// 年紀
private int age;
```

~~我想我應該看得懂這些變數名稱~~

```
/**
 * 回傳年紀
 * @return
 */
public int getAge() {
    return age;
}
```

~~我想我應該也看的出來這個函式回傳什麼~~

請將「製造干擾字詞的誘惑」轉化為「改善程式碼的決心」。

- **當你可以使用函式或變數時就別使用註解**

```
//確認客戶模組裡的第一筆資料主鍵值是否為168
if("168".equals(customerMoudle.getCustomerGroup().get(0).getSeq())){
    // do something
}
```

閱讀註解的時間可能比閱讀程式碼的時間還要多。

```
List<Customer> customerGroup = customerMoudle.getCustomerGroup();
String theFirstCustomerSeq = customerGroup.get(0).getSeq();
if("168".equals(theFirstCustomerSeq)){
    // do something
}
```

重構後，簡單明瞭。

- **出處及署名、被註解起來的程式碼**

請善用原始碼控管工具。

```
// add by KB
```

類似這種註解會隨著時間越久，越與註解中的人無關。

- **函式的標頭**

簡短、只做一件事、具描述能力的命名，會比還得將註解寫在函式標頭更好一點。

真正有益的註解，是你想辦法不寫它的註解。

是程式碼，是事實。