BAD SMELL EXAMPLES

99年度教育部資訊軟體人才培育推動中心軟體工程推廣分項計畫

編輯委員　國立台北科技大學　劉建宏教授
陳偉凱教授
郭忠義教授

Unresolved warnings

- 仍然可以產生可執行檔，並且可以執行
- 是一種隱性的錯誤
  - 雖然不是語法上的錯誤，但是在執行期間可能會發生無法預期的行為

```
1  public void printSomething() {
2     int size = 3
3     String target = null;
4
5     for(int i = 0; i < size; i++) {
6       System.out.println("i = " + i);
7     }
8
9     System.out.println(target.toString());
10 }
```

```
15  Null pointer access: The variable target can only be null at this location
```

```
i = 0
i = 1
i = 2
Exception in thread "main" java.lang.NullPointerException
        at Examples.main(Examples.java:15)
```


Every dynamic allocated memory is deallocated or there is garbage collection

- 若程式在執行過程中使用大量的 new，而沒有使用 delete 的話，會使記憶體耗盡

```
1  int main() {
2     int size = 10;
3     int result = 0;
4     int array = new int[size];
5
6     // Assign value to the array
7     for(int i = 0; i < size; i++) {
8       array[i] = i;
9     }
10
11    for(int i = 0; i < size; i++) {
12      result += array[i];
13    }
14 }
```

Memory Leak

Long method(1/2)

- 一個長函式
  - 無法完整顯示於一個畫面
  - 行數大於 20 行
- 難理解，需要經常捲動畫面了解上下文
- 難以重複利用
- 難以維護

Long method(2/2)

- 很難有合適的函式名稱
- 使用 *Extract Method* 將之拆解成短函式

```
1  public void createPartControl(Composite parent) {
2     _failnodes = new HashSet<Object>();
3     _comps = new ConcurrentLinkedQueue<IComponent>();
4     _viewer = new TreeViewer(parent, SWT.MULTI | SWT.H_SCROLL);
5     _viewer.setInput(getViewSite());
...   ...
59    _selectionHandler = new SelectionChangHandler();
60    _selectionHandler.setViewer(_viewer);
61 }
```


Feature envy

- 某函式對於某個類別(Class)的興趣高於本身所處的類別
  - 某函式經常需要取得另一個類別的多個成員變數或是呼叫多個成員函式

```
1  public void doSomething() {
2     ClassA a = new ClassA();
3     int x = a.getX();
4     int y = a.getY();
5     int z = a.calculateSomething(x + y, y);
6     a.setZ(z);
7  }
```

```
1  public ClassA() {
2     public void doSomeThing() {
3        z = calculateSomething(x + y, y);
4     }
5  }
```

- 使用*Move Method*將該函式移至另一個類別

Unsuitable naming

- 好的命名令人易讀易懂
  - Class name
  - Member method, Member data
  - Local variable

```
1  public class T() {
2     boolean b = false;
3
4     public int xyz(int x, in
5        int r = 0;
6        r = (x + y) * z / 2;
7        return r;
8     }
9  }
```

```
1  public class Trapezoid() {
2     boolean isIsosceles = false;
3
4     public int calculateArea(int top, int bottom, int height) {
5        int area = 0;
6        area = (top + bottom) * height / 2;
7        return area;
8     }
9  }
```


All assigned variables have proper type consistency or casting (1/2)

- 強制轉型(Casting)是一種危險的語言構件
- 除非你很清楚欲轉換的型別，否則應盡量避免使用
- 因變數型態不一致，可能造成無法預期的執行結果

```
1  void testType() {
2     unsigned short x = 65535;
3     short y = x;
4
5     for(int i = 0; i < y; i++) {
6        Do something
7     }
8  }
```


All assigned variables have proper type consistency or casting (2/2)

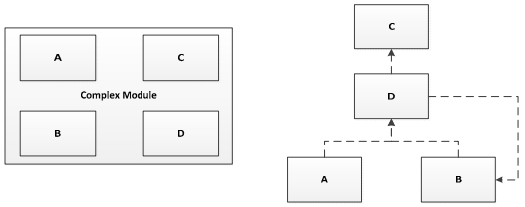
- Upcasting

```
1  class Animal() {}
2
3  class Mammal extends Animal() {}
4
5  class Cat extends Mammal() {}
6
7  class Dog extends Mammal() {}
```

```
1  Mammal m = new Mammal()
2  Cat c = (Cat)m;
```

Are any modules excessively complex and should be restructed

- 太過複雜的模組，可能使得該模組與其他模組之間產生不必要的 Coupling
  - 應將過於龐大的模組拆解成數個小模組



Loop termination conditions are obvious and invariably achievable

- 迴圈的設計步驟
  - 設定初始條件
  - 執行過程中，逐步修改初始條件
  - 謹慎定義結束條件

```
1  for(int i = 1; (i % 2) ? ((i + 100) < 200) : ((i* 30) < 50); i++) {
2     Do something
3  }
4
5  for(int i = 0; i < 100; i++) {
6     Do something
7     i = i * 5;
8  }
9
10 int i = 0;
11 while(i < 10) {
12    Do something
13 }
```

```
1  for(int i = 1; i < 10; i++) {
2     Do something
3  }
4
5  for(int i = 0; i < 100; i++) {
6     Do something
7  }
8
9
10 int i = 0;
11 while(i < 10) {
12    Do something
13    i++;
14 }
```


Parentheses are used to avoid ambiguity

- 適當的運用括號
  - 增加可讀性
  - 可避免邏輯上的錯誤

```
1  public int trapezoidArea(int top, int bottom, int height) {
2     int area = top + bottom * height / 2;
3     return area;
4  }
5
6  if (isOK && getX() *
7     Do something
8  }
```

```
1  public int trapezoidArea(int top, int bottom, int height) {
2     int area = (top + bottom) * height / 2;
3     return area;
4  }
5
6  if ((isOK) && (getX() * getY() == 2000) && (!isFinished)) {
7     Do something
8  }
```

Lack of comments(1/2)

- 良好的註解可提高可讀性
- 註解撰寫時機
  - 需要產生 API 文件(Java Doc 式的註解)
  - 複雜的演算法
  - 輔助設計與思考

Lack of comments(2/2)

```
1  public RSSIMapCollection() {
2    _maps = new Hashtable<String, RSSIMap>();
3    _listeners = new Vector<RSSIMapCollectionEventListener>();
4    _stabilizes = new SelectionProperty(STABILIZES_LABEL);
5    _stabilizes.addElement(Stabilize.NONE);
6    _stabilizes.addElement(Stabilize.THRESHOLD);
7    _stabilizes.addEle
8    _stabilizes.addEle
9    _stabilizes.setSel
10 }
```

```
1  public RSSIMapCollection() {
2    _maps = new Hashtable<String, RSSIMap>();
3    _listeners = new Vector<RSSIMapCollectionEventListener>();
4
5    // Initialize a selection property for multiple stabilizations
6    _stabilizes = new SelectionProperty(STABILIZES_LABEL);
7    _stabilizes.addElement(Stabilize.NONE);
8    _stabilizes.addElement(Stabilize.THRESHOLD);
9    _stabilizes.addElement(Stabilize.AVERAGE);
10   _stabilizes.addElement(Stabilize.WIEGHTED);
11   _stabilizes.setSelectedItem(Stabilize.THRESHOLD);
12 }
```


Fat View (1/2)

- View除了呈現畫面功能的程式碼以外，不應該含有

```
1    // codes that create menus, buttons, and connects signals to slots (omitted)

32   MainWindown::loadMindMap() {
33     /** open dialogue box that lets
34        * file using ifstream. Also, assu
35        * varaibles that we need to re
36        */
37     ...
45     while (fin.eof()) { // fin is a ifstre
46       fin >> line;
47       if (line == "//ROOTNODE") {
48         fin >> nodeId >> nodeDescription;
49         newRoot = new AbstractNode(nodeId, nodeDescription);
50         fin >> coordinateX >> coordinateY >> width >> height;
51         newRoot->setX(coordinateX);
52         ... // more bussiness logic
100  }
```

```
// ROOTNODE
0 MindMind_Topic
50 50 40 60
// NODE
1 10 Node_Description
150 0 40 60
...
```

read the text

Sample Input File

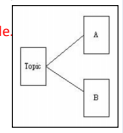
Fat View (2/2)

- View除了呈現畫面功能的程式碼以外，不應該含有

```
1    // codes that create menus, buttons, and connects signals to slots (omitted)

32   void MindMap::loadMindMap(string filePath) {
33     while (fin.eof()) {
34       fin >> line;
35       if (line == "//ROOTNODE") {
36         fin >> nodeId >> nodeDescription;
37         newRoot = new AbstractNode(nodeId, nodeDescription);
38         fin >> coordinateX >> coordinateY >> width >> height;
39         newRoot->setX(coordinateX);
40         ... // set coordinateY, width, and height for root node.
41       } else if (line == //NODE) {
50         fin >> parentId >> nodeId >> nodeDescription;
51         ... // more bussiness logic
     }
     ... // more methods
```

MindMap object is now responsible for loading exisitng mind map.



Files are checked for existence before attempting to access them

- 開啟檔案之後沒有測試檔案是否正確載入就進行操作。

```
1    ... // include necessary header files.
5  using namespace s
6  int main () {
7    ifstream inputFil
8    inputFileStream.
9    char output[100
10   while (!inputFile
11     inputFileStrea
12     ... // process
16   }
17   inputFileStream.
18 }
```

```
1    ... // include necessary header files.
5  using namespace std;
6  int main () {
7    ifstream inputFileStream;
8    inputFileStream.open("MyText.txt");
9    char output[100];
10   if (inputFileStream.is_open()) {
11     while (!inputFileStream.eof()) {
12       inputFileStream >> output;
13       ... // process read-in data
16     }
17   } else {
18     ... // error-handling code
20   }
22 }
```

Check if file has been opened successfully.

Each class have appropriate constructors and destructors

- 不宣告Constructor以及Destructor而使用編譯器產生的

```
1  Class Student {
2  public:
3    ~Student () {
4      delete _fullName; // release source
5    }
6    Student (int id, char *fullName) {
7      _id = id;
8      int length;
9      _fullName = new char [length + 1]; // allocate memory space
10     strcpy(_fullName, fullName);
11   }
12   ...
20 private:
21   int _id;
22   char* _fullName;
23 }
```

Now we have Constructor and Destructor

Duplicated Code (1/2)

- 一段段重複的程式碼在一個或多個類別或方法中出現。

```
public class ClassAReport {
  ...
  public int calculateAverage(List<Integer> scores) {
    int sum, average = 0;
    for (int i = 0; i < scores
      sum += scores.get(i
    }
    average = sum / score
    retrun average;
  }
  ...
}
```

```
public class ClassBReport {
  ...
  public int calculateAverage(List<Integer> scores) {
    int sum, average = 0;
    for (int i = 0; i < scores.size(); i++) {
      sum += scores.get(i);
    }
    average = sum / scores.size();
    retrun average;
  }
  ...
}
```

This piece of code occurs more than once!

Duplicated Code (2/2)

- 重複的程式碼應當抽出來改寫成類別或是副程式讓其他程式得以共享。

This class is responsible for calculating average.

```
public class AverageCalculator {
  public int calculateAv
    int sum, average = 0
    for (int i = 0; i < sco
      sum += scores.ge
    }
    average = sum / sc
    retrun average;
  }
  ...
}
```

```
public class ClassAReport {
  ...
  public int calculateAverage(List<Integer> scores) {
    AverageCalcular ac = new AverageCalculator();
    return ac.calulateAverage(scores);
  }
}
```

Let ac do the math.

```
public class ClassBReport {
  ...
  public int calculateAverage(List<Integer> scores) {
    AverageCalcular ac = new AverageCalculator();
    return ac.calulateAverage(scores);
  }
}
```


All methods have appropriate access modifiers and return types (1/2)

- 副程式都沒有加上適當的access modifier，使其他類別可以無限制的存取該類別應當保護的資料或是副程式。
  - 應按照設計原則 (如Information Hiding)替成員涵式程式以及成員變數冠上適當的access modifier.

```
Class Account {
public:
  string _password;
  string getPassword();
  ...
};
```

```
Class Account {
public:
  string getPassword();
  ...
private:
  string _password;
  ...
};
```


All methods have appropriate access modifiers and return types (2/2)

- 副程式都沒有加上適當return type，無法得知副程式的執行是否正常。(C++ as example language)

```
int main() {
  bool openA
    ifstream i
    ifs.open(f
    if (!ifs.is_
      return
    ...
    return tru
  }
```

```
Int main() {
  string filePath;
  cin >> filePath;
  bool isFileOpenedAndProcessed;
  isFileOpenedAndProcessed =
    openAndProcessFile(filePath);
  if(isFIleOpenedAndProcessed) {
    ... // do something
  } else {
    ... // error handling code
  }
}
```

Client checks if the file is opened & processed.

Are there any redundant or unused variables?

- 副程式 / 類別內存在多餘的變數。這些變數至始至終都沒有被存取及使用過。
  - 刪除副程式或類別中未曾使用存取過的變數。

```
public int calculateClassAverage (List<Integer> scores) {
  int rank = 0; // never used
  int sum, average = 0;
  for (int i = 0; I < scores.size(); i++) {
    sum += scor
  }
  return averag
}
```

```
public int calculateClassAverage (List<Integer> scores) {
  int sum, average = 0;
  for (int i = 0; I < scores.size(); i++) {
    sum += scores.get(i);
  }
  return average;
}
```

Delete unused variable

Indexes or subscripts are properly initialized, just prior to the loop

- 迴圈下標或條件值無適當地初始化。
  - 迴圈下標或條件值應適當地初始化。

```
int i;
while (i < 0) {
  doSomething();
  i++;
}
```

```
int i = -10;
while (i < 0) {
  doSomething();
  i++;
}
```

```
int i;
for (i ; i < someInt; i++) {
  doSomething();
}
```

```
int i = 0;
for (i ; i < someInt; i++) {
  doSomething();
}
```

Is overflow or underflow possible during a computation?

- 計算時沒有檢查是否會發生overflow或underflow
  - 計算後應該檢查是否會發生overflow或underflow

```
int main () {
  short int adde
  short int auge
  short sum = ac
  doSomething(
};
```

```
int main () {
  short int addend, augend;
  cin >> addend;
  cin >> augend

  if (addend + augend > numeric_limits<short>::max() ||
      (addend + augend < numeric_limits<short>::min()) {
    throw "short integer overflow / underflow"
  short int sum = addend + augend;
}
```


Are divisors tested for zero?

- 做除法運算未檢查除數是否為0
  - 做除法運算前應檢查除數是否為0

```
  int divisor;
  int dividend;
  cin >> divisor;
  cin >> dividend;
  int quotient = dividend / divisor;
  ...
}
```

```
  int divisor;
  int dividend;
  cin >> divisor;
  cin >> dividend;

  if (divisor == 0) {
    throw "divisor is 0";
  }
  int quotient = dividend / divisor;
  ...
}
```


Inconsistent coding standard

- 程式碼與規定的程式碼標準不一致
  - 程式碼應與規定的程式碼標準一致

```
1  成員變數名稱前應加底線。
2  務必使用有意義的命名。
```

```
class Car {
public:
  int getAbc();
  string getXyz();
  ...
private:
  int id;
  string manufactureDate;
  ...
};
```

meaningless naming

Inconsistent coding standard

```
class Car {
public:
  int getVehicleId ();
  string getManufactureDate();
  ...
private:
  int _id;
  string _manufactureDate;
  ...
};
```


Data clumps

- 指常常在不同地方成群出現的相同資料項

```
public class Custom
  private String na
  private String titl
  private String ho
  private String str
  private String cit
  private String po
  private String co
  private String name;
  private String title;
  private Address customerAddr;

  ...
}
```

```
public class Addres
  private String ho
  private String st
  private String ci
  private String co
  ...
};
```

```
public class Customer {
```

```
public class Staff {
  private String lastname;
  private String firstname;
  private String house;
  private String street;
  private String city;
  private String postcode;
  private String country;
  private String name;
  private String lastname;
  private String firstname;
  private Address staffAddr;

  ...
}
```


Switch statement

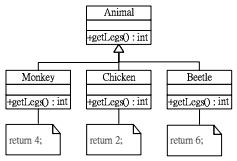
- 可使用物件導向中的多型(Polymorphism)來取代switch statement。

- Not good:

```
public int getLegsNum() {
  switch(animal) {
  case 'chicken':
    return 2;
  case 'monkey':
    return 4;
  case 'beetle':
    return 6;
  default:
    return 0;
  }
}
```

- Better solution:

```
public int getLegsNum(Animal a) {
  return a.getLegs();
}
```



Large class

- 通常發生在一個類別作了太多的事情，必須將這種類別去做妥善的分工。

```
public class 班長() {
  public void 辦班遊() {
    ...
    規劃行程();
    收錢();
    ...
  }
  public void 維持秩序() {...}
  public void 收錢() {...}
} ...
}
```

```
public class 風紀() {
  public void 維持秩序() {
    ...
  }
}
public class 總務() {
  public void 收錢() {
    ...
  }
}
public class 康樂() {
  public void 規劃行程() {
    ...
  }
}
```

Long parameter list

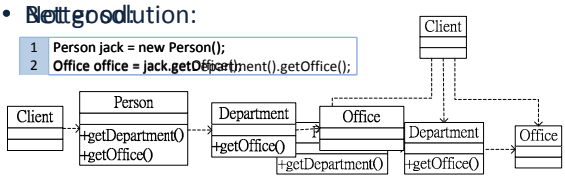
- 建議可以使用 Introduce Parameter Object 重構方法將過長的參數包裝成一個「參數物件」。
- Not good solution:

```
1  public class Member {
2     public createMember(
3        Name name,
4        String address) {
5        String postcode,
6     }  String city,
7  }     String street,
8        String house) {
9        ...
10    }
11 }
```


Message Chains

- 如果程式碼中出現客戶向一個物件請求另一個物件，然後在向後者再請求另一個物件…，這就是Message Chain。
- Not good solution:

```
1  Person jack = new Person();
2  Office office = jack.getOfficer().getOffice();
```



Literal constants

- 應該先用(static) const或define關鍵字(視語言而異)定義常數，然後再使用之。
- Not good:

```
1  public double potentialEnergy(double mass, double height) {
2     return mass * 9.81 * height;
3  }
```

- Better solution:

```
1  public double potentialEnergy(double mass, double height) {
2     final static double GRAVITATION = 9.81;
3     return mass * GRAVITATION * height;
4  }
```


Every variable is properly initialized

- 在寫程式時總是會宣告一些必須使用的變數，每個變數都需要有正確的初始化。
- Not good:

```
1  Person person;
2  Manager = person.getManager();
3  int workHours, hourlyWage;
4  Int salary = workHours * hourlyWage;
```

- Better solution:

```
1  Person person = new Person();
2  Manager = person.getManager();
3  int workHours = 40, hourlyWage = 120;
4  Int salary = workHours * hourlyWage;
```


There are uncalled or unneeded procedures or any unreachable code

- 任何 uncalled、unneeded、unreachable 的程式存在在專案中，總是會造成閱讀上的不便和負擔，也會讓程式太過冗長，浪費不必要的空間。

There are uncalled or unneeded procedures or any unreachable code

```
1  if(i < 60) {
2     //unreachable
3     if(i == 60) {
4        System.out.println("PASS");
5     }
6     else{
7        System.out.println("NOT PASS");
8     }
9  }
10 else{
11    System.out.println("PASS");
12 }
```

```
1  public class Client {
2     public createMember(Name name)
3     {
4        Name name = new Name();
5        Member.createMember(name);
6     }
7  }
```

```
1  public class Member {
2     public Member createMember(
3        Name name
4     ) {…}
5     //uncalled or unneeded procedure
6     public Member createMember(
7        String lastName,
8        String firstName,
9     ) {…}
10 }
```

Does every switch statement have a default?

- 對於每個 switch-case ，都必須有宣告的 default 動作。
- Not good:

```
1  switch(weekday) {
2    case 'Monday':
3      System.out.println("國文課");break;
4    case 'Tuesday':
5      System.out.println("英文課");break;
6    case 'Thursday':
7      System.out.println("數學課");break;
8  }
```

- Better solution:

```
1  switch(weekday) {
2    case 'Monday':
3      System.out.println("國文課");break;
4    case 'Tuesday':
5      System.out.println("英文課");break;
6    case 'Thursday':
7      System.out.println("數學課");break;
8    default:
9      System.out.println("休息");break;
12 }
```


The code avoids comparing floating-point numbers for equality

- 為了避免浮點數運算時的誤差，應該要盡力去避免做浮點數值的相等比較。
- Not good:

```
1  double x = 1e-10, y1 = 20e-10, y2 = 19e-10;
2  double y = y1 – y2;
3  if(x == y) {
4    System.out.println("X == Y");//並不會成立
5  }
```

- Better solution:

```
1  double x = 1e-10, y1 = 20e-10, y2 = 19e-10;
2  double y = y1 – y2;
3  if(Math.abs(x – y) < 1e-5) {
4    System.out.println("X == Y");//成立
5  }
```


All comments are consistent with the code

- 註解和程式碼必須一致
- Not good:

```
1  // 計算一年獲利, 傳入參數(int amount)
2  public void countProfit(int amount, double rate) {
3    _profit = amount * ( 1 + rate );
4  }
```

- Better solution:

```
1  // 計算一年獲利, 傳入參數(int amount, double rate)
2  public void countProfit(int amount, double rate) {
3    _profit = amount * ( 1 + rate );
4  }
```


CODING STANDARD EXAMPLES

White space and empty line(1/4)

- 假設會在下列位置加上空白
  - ① 左大括號前加上空白
  - ② 函式若有兩個以上參數，於逗號後加上空白
  - ③ 類別實作多個介面，或介面繼承多個介面，於逗號之後加上空白

```
1  public void encoc
2  }
3
4  public abstract v
5
6  public class Bird e
```

```
1  public void encode(WIMAXBurst burst) {
2  }
3
4  public abstract void connect(String host, int port) {}
5
6  public class Bird extends Animal implements Flyable, Singable {}
```


White space and empty line(2/4)

- 假設以下情況不加空白
  - ① for, while, do, switch 關鍵字與左括號之間
  - ② 函式的第一個參數與左括號之間
  - ③ 函式的最後一個參數與右括號之間
  - ④ 函式名稱與左括號之間
  - ⑤ 轉型(Type casting)宣告與變數之間

```
1  while (...) {}
2
3  public abstract void conn
4
5  WIMAXPDU pud = (WIMA
```

```
1  while(...) {}
2
3  public abstract void connect(String host, int port) {}
4
5  WIMAXPDU pud = (WIMAXPDU)pdus.get(index);
```

White space and empty line(3/4)

- 適時的空行有助於了解程式的組織
  - 過多或不當的空行反而會造成凌亂
- 假設以下情況會加入一行空行
  - function 主體
  - for, while, if, switch 等迴圈控制
  - 一段有組織且有意義的程式碼(Line 10~15)

White space and empty line(4/4)

```
1  public Boolean isClosed() {
2    return _closed;
3  }
4  public RSSIMa
5    _maps = nev
6    _listeners =
7    // Initialize a
8    _stabilizes =
9    _stabilizes.a
10   _stabilizes.a
11   _stabilizes.a
12   _stabilizes.a
13   _stabilizes.s
14 }
```

```
1  public Boolean isClosed() {
2    return _closed;
3  }
4
5  public RSSIMapCollection() {
6    _maps = new Hashtable<String, RSSIMap>();
7    _listeners = new Vector<RSSIMapCollectionEventListener>();
8
9    // Initialize a selection property for multiple stabilizations
10   _stabilizes = new SelectionProperty(STABILIZES_LABEL);
11   _stabilizes.addElement(Stabilize.NONE);
12   _stabilizes.addElement(Stabilize.THRESHOLD);
13   _stabilizes.addElement(Stabilize.AVERAGE);
14   _stabilizes.addElement(Stabilize.WIEGHTED);
15   _stabilizes.setSelectedItem(Stabilize.THRESHOLD);
16 }
```


Indention(1/2)

- 假設以下情況會加入縮排與換行
  - ① 左大括號不換行
  - ② 右大括號換行，且與左大括號所在之行對齊
  - ③ 程式碼依內容階層深度縮排
  - ④ case, default 關鍵字與 switch 對齊
  - ⑤ break 關鍵字與內容縮排
  - ⑥ catch, finally 關鍵字與 try 對齊

Indention(2/2)

```
1  try {
2    stream = socket.getInputStream();
3  } catch (Exception e) {
4    System.out.println(e.toString());
5  } finally {                    ⑥
6    socket.close();
7  }                              ②
8                                 ④
9  switch(type) {
10   case SOCKET_CONNECTED:
11   default:
12     close();
13   break;                       ⑤
14 }
```

```
1  try {
2    stream = socket.getInputStream();
3  }
4  catch (Exception e) {
5    System.out.println(e.toString());
6  }
7  finally {
8    socket.close();
9  }
10
11 switch(type) {
12 case SOCKET_CONNECTED:
13 default:
14   close();
15   break;
16 }
```


註解

- 區塊註解(/**.../)用途
  - 檔案說明
  - 函式介面說明
  - 類別說明
- 除上述用途以外，其餘使用單行註解
  - 獨自使用一行
  - 不加在Statement的分號之後

區塊註解-類別說明

- 類別說明包含：用途、版本、作者以及相關類別(若無相關類別則免)。(Using Javadoc)

```
1  /** The EthernetSocket extends the abstract Sock
2  * TCP implementation for establishing and closing
3  * After the connection is established, the client ca
4  * data from the socket with TCP protocol.
5  *
6  * @version 3.0.0
7  * @author Spirit Du
8  * @see EthernetSeverSocket
9  * @see <a href="somewhere">Observer</a> patte
10 class EthernetSocket extends Socket { }
```

javadocTest

Class EthernetSocket

java.lang.Object
  └java.net.Socket
    └**javadocTest.EthernetSocket**

public class **EthernetSocket**
extends java.net.Socket

The EthernetSocket extends the abstract Socket class t
data from the socket with TCP protocol.

Version:
3.0.0
Author:
See Also:
EthernetSeverSocket, Observer pattern

區塊註解-檔案說明

- 每個檔案起始必須包含一個註解區塊，用來說明檔案的檔名以及授權。

```
/* MPEGTSManager.h
 *
 * Copyright (C) 2010 Software Development Research Center,
 * National Taipei University of Technology.
 * All rights reserved.
 *
 * This file is a part of NTUT DVB-H. NTUT DVB-H is free software: you can
 * redistribute it and/or modify it under the terms of the GNU General Public
 * License.
 * ...
 */
```


區塊註解-函式介面說明

- 程式的介面說明須包含函式名稱、參數用途以及回傳變數說明。(□ 表示空白)

```
/** Connect to the remote host. This function may throw
 * UnkownHostException, if the remote client specified by the host
 * is unable to resolved by <a href="somewhere">DNS</a> protocol.
 *
 * @param host the host name
 * @param port the TCP/UDP port number
 * @return error code; return 0 when succes
 */
public int connect(string host, int port) { }
...
```

connect

```
public int connect(java.lang.String host,
                   int port)
```

Connect to the remote host. This function may throw

Parameters:
host - the host name
port - the TCP/UDP port number

Returns:
error code; return 0 when successful

命名規則

- 為了增進程式的可讀性及可維護性，在coding standard中訂定明確的程式命名規則是相當重要的，其中又可以將其分為三個部分
  - **類別、介面、列舉**
  - **變數**
  - **函式**

命名規則-通用規則

- 類別、介面與列舉(enum)名稱首字母大寫，若由多個單字組成，每個單字的第一個字母大寫。
- 不以C或I開始來分別class或interface。
- 若使用專有名詞縮寫，則全部大寫。

```
public interface IShape {...}

public class CTriangle implements IShape {...}

public class mimoAdapter {...}
```

```
public interface Shape {...}

public class Triangle implements Shape {...}

public class MIMOAdapter {...}
```


命名規則-類別

- 若為抽象類別(abstract class)，請於類別名稱加上Abstract字樣。
- 類別公開(public)成員函式請依public getters →public setters →other public operations的順序宣告。

```
public abstract class Node {
    public void addChild() {...}
    public Node getParent() {...}
    public void setParent(Node node) {...}
}
```

```
public abstract class AbstractNode {
    public void setParent(Node node) {...}
    public Node getParent() {...}
    public void addChild(Node node) {...}
}
```


命名規則-列舉

- 列舉命名規則與類別命名規則相同。
- 成員命名規則等同類別靜態常數命名規格(所有字母大寫，並以底線隔開單字)。

```
public enum season {
    spring,
    summer,
    fall,
    winter
}
```

```
public enum Season {
    SPRING,
    SUMMER,
    FALL,
    WINTER
}
```

命名規則-介面

- 介面的靜態變數宣告僅限用於常數，命名規則等同類別靜態常數(所有字母大寫，並以底線隔開單字)。
- 函式宣告命名規則等同類別函式命名規則。

```
1 public interface MemoryMonitor extends Runnable, Cachable {
2    static int memoryLimit = 8 * 1024 * 1024;
3 }
```

```
1 public interface MemoryMonitor extends Runnable, Cachable {
2    static int MEMORY_LIMIT = 8 * 1024 * 1024;
3 }
```


命名規則-變數 (1/2)

- Class成員變數以底線開始，底線後第一個子母小寫，若由多個單字組成，從第二個單字開始，每個單字的第一個字母大寫
- 區域變數及函式參數命名規格與class成員變數相似，僅差不以底線開始。

```
1 public class VariableExample1 {
2    private int privatemember;
3    public int publicmember;
4 }
```

```
1 public class VariableExample2 {
2    private int _privateMember;
3    public int publicMember;
4 }
```


命名規則-變數 (2/2)

- 變數名稱不加形態(type)縮寫
- Array、list或vector等容器，名稱結尾建議加s表示複數，存取其中的元素時，元素名稱不加s以示區別

```
1 public class VariableExample2 {
2    public int iPublicMember;
3    public Vector<Integer> publicVector = new Vector<Integer>();
4 }
```

```
1 public class VariableExample2 {
2    public int publicMember;
3    public Vector<Integer> publicVectors = new Vector<Integer>();
4 }
```


命名規則-函式 (1/2)

- 函式命名一律小寫開始，從第二個單字開始，除介系詞(in、to等)外，每個單字的第一個字母大寫。
- 每個函式表示一種操作，因此一律使用動詞作為第一個單字。

```
1 public void CountTotal() {…}
2 public void transferDataToDB() {…}
3 public void TotalCount() {…}
```

```
1 public void countTotal() {…}
2 public void transferDatatoDB() {…}
3 public void countTotal() {…}
```


命名規則-函式 (2/2)

- Getter除了取得bool值使用**isXXX()**之外，其餘一律使用**getXXX()**。
- Setter一律使用**setXXX()**。
- Callback function或event listener都是在某個動作後或某個事件發生後被呼叫，因此以過去分詞結尾

```
1 public int Id() {}
2 public boolean getVip() {}
3 public void inputId(int id) {}
4 public void baseStationSwitch (int oldStationID, int newStationID) {}
5 public void mouseClick(int x, int y, int button) {}
```

```
1 public int getId() {}
2 public boolean isVip() {}
3 public void setId(int id) {}
4 public void baseStationSwitched(int oldStationID, int newStationID) {}
5 public void mouseClicked(int x, int y, int button) {}
```


Thank you for your time.

Q & A