Window Programming Term Project Coding Convention

# Introduction

코딩 컨벤션이란? **읽고, 관리하기 쉬운 코드를 작성하기 위한 일종의 코딩 스타일 규칙**이다. 가독성, 일관성, 구체성의 세가지 원칙을 갖고 규칙을 지키려고 노력하며 재량에 따라 적용한다.

# General Rule

File은 최대 2000line을 넘기지 않도록 한다.

한 line에는 가급적 최대 80 characters를 넘기지 않도록 한다.

Tab : keep tab / tabstop = 4

# File Organization

Member들은 가급적 다음 순서로 정렬한다.

1. Class variables (public > protected > private)
2. Instance variables (public > protected > private)
3. Constructors
4. Methods (grouped by functionality)

# Definition

1. Class Definitions

|  |
| --- |
| Class MyClass {  …  } |

1. Method Definitions

|  |
| --- |
| public void doSomething(int withArgument) {  …  } |

# Indentation

Wrapping line rule

* Break after a comma
* Break before an operator
* Align the new line with the beginning of the expression at the same level on the previous line

|  |
| --- |
| method(longExpression1, longExpression2, longExpression3,  longExpression4, longExpression5);  longName1 = longName2 \* (longName3 + longName4) \* (longName5)  + 4 \* longName6; //PREFERRED  longName1 = longName2 \* (longName3 + longName4  ) \* (longName5) + 4 \* longName6; //AVOID  someMethod(int anArg, Object anotherArg, String YetAnotherArg,  Object andStillAnother)  {  …  } |

# Whitespaces

Blank space는 일반적으로 아래의 경우에 적용된다.

* Keyword와 “(“ 사이, “)” 와 “{“ 사이, “} or “{“와 keyword 사이
* Binary operator 와 expression 사이

다음에는 적용하지 않는다.

* Identifier 와 “(“ 사이에는 blank space가 적용되지 않는다.
* Unary Operator는 붙여서 쓴다.

|  |
| --- |
| for (…) {  …  }  Method(…) {  …  }  Double length = Math.sqrt(x \* -y \* y); |

# Statements

다음 항목들에서 제시하는 대로 괄호 () {} 나 띄어쓰기, 줄바꿈 등을 사용하도록 한다.

1. if, if-else, if else-if else statements

* {}는 생략하지 않는 것을 기본으로 한다.
* 예외 : 해당 statement가 아주 간단할 경우에는 생략해도 무관

|  |
| --- |
| if (a == b) {  …  } else if (…) {  …  } else {  …. |

1. for, while, do-while statements

* 다음 형식을 사용한다.

|  |
| --- |
| for (initialization; condition; update) {  …  }  for (initialization; condition; update)  ;  do {  …  } while (…); |

1. switch statements

* switch와 case를 같은 시작점에서 시작한다.
* case와 case 문 사이는 blank line을 가진다.
* break statement를 가지지 않는 case일 경우에는 정확한 code 흐름을 파악하기 위해서 /\* fall through \*/ comment를 add한다.

|  |
| --- |
| switch (condition) {  case ABC:  statements;  /\* fall through \*/  case DEF:  statements;  break;  case XYZ:  statements;  break;  default:  statements;  break;  } |

1. try-catch statements

* 다음 형식을 사용한다.

|  |
| --- |
| try {  statements;  } catch (ExceptionClass e) {  statements;  } finally {  statements;  } |

# Naming Conventions

1. Meaningful names

* 이름으로부터 의미를 알 수 있도록 한다.

|  |
| --- |
| // 좋지 않은 예제  if (a < 65) {  y = 65 – a;  } else {  y = 0;  }  // 올바른 예제  if (age < RETIREMENT\_AGE) {  yearsToRetirement = RETIREMENT\_AGE – age;  } else {  yearsToRetirement = 0;  } |

1. Acronyms

* Capitalize only the first letter in acronyms

|  |
| --- |
| setDSTOffset > setDstOffset |

1. Type Names

* Class 이름을 구성하는 각 단어의 첫 글자가 대문자가 되도록 한다.
* Class 이름은 영어 명사로 한다
* Abc를 상속하면서 Def 속성을 갖는 것은 DefAbc라 이름짓는 것을 원칙으로 한다.

|  |
| --- |
| FeedXmlDownloader, Request, DetailActivity |

1. Interface Names

* 영어 명사 혹은 형용사로 한다.
* 대문자 I로 시작한다.

|  |
| --- |
| IHmcpMessage, IDrawable, IContentDownloadNotification |

1. Method Names

* 영어 동사로 소문자로 시작하며, 따라오는 각 단어의 첫 자는 대문자로 한다.

|  |
| --- |
| void withdraw(int amount);  Image getScaledInstance();  bool isValid();  string getName(); |

* 명사, 전치사, 형용사, 부사 등을 덧붙여 이름으로 하는 일을 알 수 있게 한다.

|  |
| --- |
| createTable(), makeVoiceCall(), … |

* 함수 인자의 이름으로 의미를 알 수 있게 한다.

|  |
| --- |
| switchTask(string name), startsWith(char ch), … |

* 함수 인자로 의미를 알 수 있는 경우 함수 이름에 중복해서 쓰지 않는다.

|  |
| --- |
| bindFeatureAndDevice(Feature feature, Device device); // 좋지 않음  bind(Feature feature, Device device); // 좋음 |

* bool 상태를 얻어 오는 함수는 3인칭 현재시제를 사용한다.

|  |
| --- |
| isAvailable(), hasJob(), endWith(char ch), canShowIcons(), … |

* Notification method는 명사+과거분사 형태로 사용한다.

|  |
| --- |
| messageReceived(Msg, msg), ringOccurred(string callerID), … |

1. Variable Names

* 영어 명사로 소문자로 시작하며, 따라오는 각 단어의 첫 자는 대문자로 한다.
* Collection type의 경우 영어 복수형을 사용한다.

|  |
| --- |
| string name;  XmlHandler xmlHandler;  ListView coverflowEpisodeListView;  ArrayList<ContentRequest> allRequests;  string[] targetNames; |

* Parameter에 member와 같은 이름이 사용되는 경우 this->로 구분한다. 하지만, 혼동의 우려가 있으니 가급적 다른 이름을 사용하도록 한다.

|  |
| --- |
| public void setName(string name) {  this->name = name;  }  public void setName(string aName) {  name = aName;  } |

1. Constant Names

* 영어 명사를 사용한다.
* 대문자로 구성하며, 각 연결되는 단어에는 “\_”를 이용하여 구분한다.
* enum에 대해서도 동일한 규칙을 적용한다.

|  |
| --- |
| public static constant int ALIGN\_LEFT = 0;  public static constant int ALIGN\_RIGHT = 2;  public static constant string CONTENT\_SMS\_URL = “content://sms” |

# Synchronization

* Comment가 Code와 항상 Sync가 맞아야 한다.
* Abc를 상속하면서 Def 속성을 갖는 것은 DefAbc라 이름 짓는 것을 원칙으로 한다.

# Comment types

1. Documentation Comment

* Programming interface를 설명할 때 사용하며, JavaDoc 사용법에 준한다.

|  |
| --- |
| /\*\*  \* Document Comment (JavaDoc Comment)  \*/ |

1. Standard Comment

* Code를 지우지 않고 숨길 때 사용

|  |
| --- |
| /\*  If (smsManager.getSmsListSize() < 2) return;  currentPage++;  if (currentPage >= smsManager.getSmsListSize()) CurrentPage = 0;  refreshWidget();  \*/ |

1. One-line or end-line Comment

* 상세 구현 설명을 할 때 사용한다.

|  |
| --- |
| If (availableSize <= contentLength) {  // not enough SD card memory  // this.dnbase.notifyNotEnoughMemory();  // isNotEnoughMemory = true;  continue;  } |

# References

The Elements of Java Style ( sigs reference library )

<http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>

UbiVelox Java Code Convention ( 2009-11-12. Kim, Ki-Yong)