Project Coding Style – Java

1. 개요

1-1. 목적

• 소프트웨어 비용의 80% 정도가 유지보수에 들어간다. 소프트웨어의 유지보수란 소프트웨어를 사용하면서 필요에 따라 소스 코드를 읽고, 수정하는 등의 작업을 말한다고 할 수 있다.

• 대부분의 경우 소프트웨어를 원래 개발한 사람이 끝까지 유지보수 할 수는 없다. 따라서, 유지보수하는 사람은 다른 사람이 이미 작성해 놓은 소프트웨어를 읽고 이해해야 한다.

• 코드 규칙은 소프트웨어의 가독성(readability)을 높여 줌으로써, 개발자가 처음 접하는 새로운 소프트웨어를 보다 빠르고 완벽하게 이해할 수 있도록 해 준다.

• 소프트웨어를 상품으로서 판매할 경우, 소프트웨어 코드를 깔끔하게 작성해야 한다. 소프트웨어를 구입한 구매자의 경우 그 소프트웨어를 읽고 이해하여 새로운 기능을 구현하거나 또는 새로운 소프트웨어를 개발할 수도 있다.

• 이 때, 한 가지 중요한 것은 모든 사람이 코드 규칙을 만족하도록 프로그램을 작성해야 한다는 것이다. 코드 규칙이란 다른 사람을 위한 배려이고, 서로간의 약속이기 때문이다.

1-2. 범위

• 본 문서에 작성되는 Coding Style Guide는 Bitcamp 4조 1차 프로젝트의 서버단(Java)에 모두 적용된다.

1-3. 정의 및 약어

• 본 Coding Style Guide의 지침의 정도는 크게 세 단계로 나뉘어진다. 각 단계 중 특정한 programming language에 대응되는 규칙에 대해서는 language의 이름을 뒤에 붙인다.

1) M(Mandatory) : 의무적으로 지켜야 함.
2) R(Recommended): 의무적으로 지켜야 하는 것은 아니지만, 지킬 것을 권장함.
3) O(Optional): 지침을 참고하여 지키기로 결정되는 경우 지킴

2. 공통법칙

2-1. 작업 디렉토리

- [M] 이클립스 IDE를 사용하는 경우 프로젝트-> 패키지 -> 모듈-> 클래스 형태에 수직분류로 이룬다.

- [M] 프로젝트가 여러 모듈로 이루어진 경우 소스 안에서 다시 서브 디렉토리를 나눈다.

- [M] 프로젝트 구조는 예는 다음과 같다.

- Project - Java Resources - src – Packages - Java Class Files
- Project – Java Resources – Libraries – Libraries(*.jar)

2-2. 소스파일

- [M] 각 자바 소스 파일은 하나의 Public Class 혹은 Public Interface를 포함한다. Public Class와 관련된 Private Class/Interface는 함께 두어도 무방하다. 단, Public Class/Interface의 경우 파일의 제일 상단에 위치시킨다.

- [R] 하나의 Method 혹은 Class에서 사용될 객체의 선언 및 생성은 상단에 위치시킨다.

- [R] 하나의 파일은 여러 개의 섹션으로 구성하며 각 섹션은 공백라인으로 구분한다. 또한, 섹션의 시작 부분에는 필요에 따라 주석문을 이용하여 주석을 달아 놓는다.

3. 명명법

- [M] Java의 경우 기본적으로 낙타형 표기법을 준수한다. (ex. JavaClassName, JavaMethodName)

3-1. 패키지명(Package Name)

• [M] 패키지 이름의 접두어는 반드시 아스키 문자 소문자로 시작해야 하며, 도메인의 최상위 레벨의 이름 중 하나여야 한다. 현재, com, edu, gov, mil, net, org, 또는 ISO 표준 3166, 1981에 서정의하고 있는 각 국가를 구분하기 위한 두 개의 영문자 일 수도 있다. 한국의 경우 "kr"이다.
ex) kr.or.bit.controller, libraryServer

3-2. 클래스명(Class Name)

• [M] 클래스의 이름은 반드시 명사여야 하고, 첫 번째 문자는 대문자이고, 두 개 이상의 단어로 구성될 때는 각 단어의 시작 문자를 대문자로 한다. 두 개의 단어로 이뤄져있는 고유명사(ex. KakaoTalk)의 경우 단어의 시작 문자를 대문자로 할 수 있다.과 같이 클래스 이름은 직관적이고 간단명료해야한다. 머릿글자로 나타내거나 또는 약자로 나타내는 것은 반드시 피하고, 전체 이름을 사용해야한다. URL 또는 HTML에서는 약자를 많이 사용할 수 있다.
ex) Class Library.java, Class User.java

3-3. 인터페이스명(Interface Name)

• [M]인터페이스의 이름은 클래스 이름과 같이 반드시 대문자로 시작한다.
ex) Interface DaumParser.java

3-4. 메소드명(Method Name)

• [M] 메소드의 이름은 주로 동사이며 두 개 이상의 단어로 구성될 때는 첫 단어를 동사로 구성하고 소문자로 하며 이후의 각 단어의 시작 문자는 대문자로 해주어야 한다.
ex) Void getCellNum(), void setNumber()

3-5. 필드명(Field Name)

• [M] 변수, 모든 인스턴스, 클래스, 클래스 상수 등은 소문자로 시작하고, 두 개 이상의 단어로 구성될 때는 두 번 째 단어 이후부터 각 단어의 시작 문자를 대문자로 한다. 변수명을 나타낼 때 '_' 또는 '$' 문자를 사용할 수 있더라고 명수명은 '_' 또는 '$' 문자로 시작해선 안된다. 변수명은 직관적으로 파악이 가능하고 기억하기 쉽게 명명해야한다. 일반적으로 i, j, k, m, n, x 등은 정수형 임시 변수명으로 사용할 수 있고, c, d, e 등은 문자형 임시 변수명으로 사용할 수 있다.
ex) String keywore, int type, String profile_Image

3-6. 상수명(Constant Name)

• [M] 상수명은 모두 대문자로 해 주어야 하며, 두 단어 이상으로 구성될 때는 각 단어는 '_'로 구분해 주어야 합니다.
ex) Static final int CLIENT_ID;

4. 주석문

4-1. Java – 시작 주석문

• [M] 시작 주석문은 따로 명세하지 않는다.

4-2. Java – 클래스 주석문

• [M] 클래스 주석문은 따로 명세하지 않는다.

4-3. Java – 멤버 필드 주석문

• [R] 멤버 필드 주석문은 다음과 같이 라인 단위 주석으로 한다.
ex) public static final int DATABASE = 0; // TAG 상수 값.

4-4. Java – 멤버 메소드 주석문

• [R] 멤버 메소드의 주석문은 메소드 이름, 날짜, 작성자의 형식을 따르고 간단한 설명과 매개변수, 반환값은 필요에 따라 작성한다.

// 간단설명

4-5. Java – 기타 주석문

• [O] 코드 작성 중 설명이 필요한 부분에 경우 라인 단위 또는 멀티 라인 단위로 기입하도록 한다.

5. 프로그래밍 스타일 가이드

5-1. 들여쓰기(Indentation)

• [M] Indentation 단위의 단위는 Tab을 기준으로 하며 Tab의 입력 횟수에 맞추어 공백 4개, 8개, 12개 순을 유지한다.
• [M] '{'기호 는 메소드 이름과 같은 줄에 위치하고 '}' 기호는 다른 줄에 위치하며 그 줄에는 Comment를 제외한 어떠한 코드도 위치할 수 없다.
ex)

```
public void showGoodExample(){

        System.out.println("This is Good Example");
}
```

5-2. 선언

5-2-1. 패키지 / 임포트 선언(Package / Import Declaration)

• [M] Comment 이후에 필요에 따라 package 문이 나타날 수 있다. Package 문 이후엔 한 줄을 건너띄고 다음으로 import 문이 나오도록 한다. import의 경우 한 줄에 하나의 import만을 명시한다.
ex)

```
package controller

import java.io.IOException;
import java.io.InputStream;
import Util.KakaoParsing;
```

5-2-2. 클래스 / 인터페이스 선언(Class / Interface Declaration)

• [M] import문 이후로 class 또는 interface 선언문을 명시한다. class 또는 interface 선언문의 구성은 다음과 같은 순서로 나타낸다.

순서	구성성분	설명
1	Class/interface 주석문	Class/Interface 설명을 위한 주석.
2	Class/interface 선언문	Class/Interface 선언 부분.
3	Class Static Member Field	Class/Interface에 속하는 Static 멤버들. 선언은 private->protected->public 순으로 한다.
4	Class Member Field	Class/Interface에 속하는 멤버 필드들. 선언은 private->protected->public 순으로 한다.
5	Class 객체 생성자	객체 생성을 위한 생성자
6	Class Member Method	객체에서 사용 될 메소드

5-3. 공백

• [M] Keyword (if, while, return, switch, for등) 와 '('사이에 빈칸 없이 작성한다.
• [M] Keyword (if, while, return, switch, for등) 와 '{ '사이에 빈칸 없이 같은 줄에 작성한다.
• [M] Keyword (if, while, return, switch, for등) 와 '} '독립적으로 다른 줄에 작성한다.
• [M] ','뒤에 새로운 줄이 시작되지 않는 경우, 다음 항목과 한 칸을 띄운다.

5-4. 제어문 및 반복문(Control Statements and Loops)

5-4-1. IF

• [M] If-else statement에서 else 부분은 같은 줄에 위치한다. If-else if-else의 경우에 else if부분은 같은 줄에 기술한다.

ex)

```
if(condition){
      statement;

      } else  if (condition2){

      statement2;
      }  else {

      statement3;
 }
```

5-4-2. For

• [M] For statement에서 '{'는 같은 줄에, '}'는 새로운 줄에 위치한다. For문에 사용되는 변수는 따로 선언될 수도있고 For statement 안에서 선언될 수도 있다.

ex)

```
int i;
for(i=0; i<10; I++){

    statement;
}

for(int i=0; i<10; I++){

    statement;
}
```

5-4-3. Try / Catch

• [M] try / catch 문에서 '{'는 같은 줄에, '}'는 새로운 줄에 위치한다.

ex)

```
try{
    statement;
}catch (IOException ioexception){

    statement;

}  finally {

    statement;

}
```

6. 참고문헌

1. L.W. Cannon, et. al., Recommended C Style and Coding Standards Revision 6.0, 1990.
2. Doug Klunder, Hungarian Naming Conventions, 1988.
3. C/C++ Programming Standard, 삼성전자 기간네트워크 사업부
4. Steve McConnell, Code Complete, Microsoft Press, 1993.
5. "네이밍 룰", http://tisland.tistory.com/entry/%EB%84%A4%EC%9D%B4%EB%B0%8D%EB%A3%B0