코딩 규칙

1. 개요
   1. 목적
   2. 범위
   3. 정의 및 약어
2. 공통 규칙
   1. 작업 디렉토리
   2. 소스파일
3. 명명법
   1. 패키지명(Package Name)
   2. 클래스명(Class Name)
   3. 인터페이스명(Interface Name)
   4. 메소드명(Method Name)
   5. 필드명(Field Name
   6. 상수명(Constant Name)
   7. 파일명(File Name)
4. 주석문
   1. Java - 시작 주석문
   2. Java - 클래스 주석문
   3. Java - 멤버 필드 주석문
   4. Java - 멤버 메소드 주석문
   5. Java - 기타 주석문
   6. JSP - 시작 주석문
   7. JSP - 영역 주석문
   8. JSP - 항목 뒤주석문
5. 프로그래밍 스타일 가이드
   1. 들여쓰기(Indentation)
   2. 선언(Declaration)
      1. 패키지 / 임포트(Package / Import)
      2. 클래스 / 인터페이스(Class / Interface)
   3. 공백(Blank Space)
   4. 제어문 및 반복문(Control Statements and Loops)
      1. IF
      2. For
      3. Try / Catch
6. 참고문헌
7. 개요
   1. 목적
      1. Kosta 129기 3조 3차 프로젝트의 원활한 진행
      2. 팀원 간의 협업에 있어 코드 이해와 가독성 향상
      3. 분업 및 병합 작업 시행 시 혼란을 줄이고 작업 시간을 단축
      4. 프로젝트의 내용 변경 시 예상되는 확장/추가 기능 개발의 용이성
   2. 범위
      1. 본 문서의 코딩 규칙은 Kosta129기 3조 3차 프로젝트의 서버 단(Java, Servlet/JSP)와 클라이언트 단(Javscript/JQuery, HTML/CSS) 에 모두 적용한다.
8. 공통규칙
   1. 작업 디렉토리
      1. Project - Java Resources - src - Packages - \*.java
      2. Project - webapp - index.jsp
      3. Project - webapp - images - 필요한 이미지 파일 위치
      4. Project - webapp - css - 필요한 스타일 시트 위치
      5. Project - webapp - js - 필요한 자바스크립트 라이브러리 위치
      6. Project - resources- META-INF - MANIFEST.MF, \*.txt
      7. Project - webapp - WEB-INF - web.xml
      8. Project - webapp- WEB-INF - \*.jsp, \*.html
      9. Project - webapp - WEB-INF - lib – Libraries(\*.jar)
   2. 소스파일
      1. 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.
      2. 하나의 Method 혹은 Class에서 사용될 객체의 선언 및 생성은 상단에 위치시킨다.
      3. 하나의 파일은 여러 개의 섹션으로 구성하며 각 섹션은 공백라인으로 구분한다. 또한, 섹션의 시작 부분에는 필요에 따라 주석문을 이용하여 주석을 달아 놓는다.
9. 명명법
   1. 패키지명(Package Name)
      1. Java와 JSP의 경우 기본적으로 낙타형 표기법을 사용한다.
      2. ex) JavaClassName
      3. 패키지명은 기본적으로 com.or.fortune를 사용한다.
      4. 패키지를 대표하는 이름은 대문자로 시작한다.
      5. ex) com.or.fortune.Controller
   2. 클래스명(Class Name)
      1. 첫 번째 문자는 대문자이고, 두 개 이상의 단어로 구성될 때는 각 단어의 사이에 \_문자를 사용하여 구분하고, 시작 문자를 대문자로 한다.
      2. ex) Class Member\_DTO.java
      3. 클래스명은 직관적이고 명료하게 사용한다.
   3. 인터페이스명(Interface Name)
      1. 인터페이스의 이름은 대문자로 시작한다.
      2. 인터페이스의 이름 뒤에는 Action 이라 붙는다
      3. ex) InterfaceAction.java
   4. 메소드명(Method Name)
      1. 첫 단어는 동사로 구성하고 소문자로 하며 이후의 각 단어의 시작 문자는 대문자로 해주어야 한다.
      2. ex) createAddress()
   5. 변수명(variable Name)
      1. 변수, 모든 인스턴스, 클래스, 클래스 상수 등은 소문자로 시작하고, 두 개 이상의 단어로 구성될 때는 중간에 \_를 사용하여 나타낸다.
      2. ex) memo\_id
      3. 변수명은 직관적으로 파악이 가능하고 기억하기 쉽게 명명한다.
      4. 상수명은 모두 대문자로 해 주어야 하며, 두 단어 이상으로 구성될 때는 각 단어는 ‘\_’로 구분해 주어야 합니다.
      5. ex) Static final int CLIENT\_ID;
   6. 파일명(JSP File Name )
      1. 필드명과 같은 규칙을 적용한다.
10. 주석문
    1. Java 시작 주석문
       1. 파일 이름, 프로젝트명, 작성날짜, 작성자의 형식을 다음과 같이 따른다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7 | /\*  \* @FileName: 파일이름  \* @Project: fortune  \* @Date: 해당 일자  \* @Author: 작성자  \* @Desc: 설명  \*/ | [cs](http://colorscripter.com/info#e) |

* 1. Java – 클래스 주석문
     1. [M] 클래스 코드 상단에 클래스 이름, 작성날짜, 작성자의 형식을 다음과 같이 따른다.

/\*

\* @Class

\* @Date

\* @Author

: BoardEditAction

: 2016. 4. 12.

: 이슬기

\* @Desc

: 게시판 수정을 위해 dao와 접촉 \*/

* 1. Java – 멤버 필드 주석문
     1. [R] 멤버 필드 주석문은 다음과 같이 라인 단위 주석으로 한다.
     2. ex) public static final int DATABASE = 0; // TAG 상수 값.
  2. Java – 멤버 메소드 주석문
     1. [R] 멤버 메소드의 주석문은 메소드 이름, 날짜, 작성자의 형식을 따르고 간단한 설명과 매개변수, 반환값은 필요에 따라 작성한다.

/\*\*

\*

\* @method Name : editBoard

\* @date

\* @author

:2016. 4. 12.

: 배철호

\* @description : 게시판을 고치기 위해 DB와 연

결하여 해결한다.

\* @param spec : HttpServletRequest boardata

\* @return : int

\*/

* 1. Java – 기타 주석문
     1. • [O] 코드 작성 중 설명이 필요한 부분에 경우 라인 단위 또는 멀티 라인 단위로 기입하도록 한다.
  2. JSP – 시작 주석문
     1. [M] JSP페이지 제일 상단에 다음과 같은 형식을 따라 작성한다.

<%--

@Project : DAVIZN

@File name : home.jsp

@Author : 박정진

@Data : 2016.4.12

@Desc : 서버 welcome-page

--%>

* 1. JSP – 영역 주석문
     1. • [M] 하단에 작성된 여러 줄의 함수나 작동에 대해서 설명할 시에 상단에 다음과 같은 형식을 따라 작성한다.

<%------------------------------------------------------------------

Kakao User Info를 DB 저장하는 부분

------------------------------------------------------------------%>

* 1. JSP – 항목 뒤 주석문
     1. • [M] 다음과 같이 라인 단위 주석문으로 작성한다.
     2. <a href=“http://localhost:8080/home.jsp”> Home </a> //Home으로 돌아감

1. 프로그래밍 스타일 가이드
   1. 들여쓰기(Indentation)
      1. • [M] Indentation 단위의 단위는 Tab을 기준으로 하며 Tab의 입력 횟수에 맞추어 공백 4개, 8개, 12개 순을 유지한다.
      2. • [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“);

}

* 1. 선언
     1. 패키지 / 임포트 선언(Package / Import Declaration)
        1. • [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.
        2. ex) package controller import java.io.IOException; import java.io.InputStream; import Util.KakaoParsing;5-2-2. 클래스 / 인터페이스 선언(Class / Interface Declaration)
        3. [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)

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7 | try {  statement;  } catch (IOException e){  statement;  } finally {  statement;  } | [cs](http://colorscripter.com/info#e) |

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. “JSP 프로그래밍 규칙”, http://mt1716.egloos.com/v/10539335

6. “네이밍 룰”, http://tisland.tistory.com/entry/%EB%84%A4%EC%9D%B4%EB%B0%8D%EB%A3%B0