프로젝트 코드 정의서

**1. 개요 ············································································································**

**1) 목적 ·········································································································**

**2. 공통 법칙 ··································································································**

**1) 작업 디렉토리 ·························································································**

**2) 소스파일 ··································································································**

**3. 명명법 ··········································································································**

**1) 공통규칙·····································································································**

**2) 클래스명·····································································································**

**3) 메소드명·····································································································**

**4) 필드명········································································································**

**4. 주석문 ·········································································································**

**1) 공통 주석문 ······························································································**

**2) 멤버 필드 주석문 ·····················································································**

**3) 멤버 메소드 주석문 ··················································································**

**4) 기타 주석문 ······························································································**

**5. 기타 스타일 ································································································**

**1) 들여쓰기·····································································································**

**2) 선언············································································································**

**1) 패키지 / 임포트·························································································**

**2) 클래스 / 인터페이스··················································································**

**3) 공백············································································································**

**4) 제어문 및 반복문·······················································································**

**1) IF ·············································································································**

**2) For ···········································································································**

**3) Try / Catch ·······························································································**

**1. 개요**

**1-1. 목적**

소프트웨어를 개발하는 모든 과정에 들어가는 비용 중 80%가 유지보수에 쓰여진다. 소프트웨어의 직접 개발한 개발자가 그 소프트웨어의 유지보수를 담당하는 경우는 거의 볼 수 없는 경우다. 규칙을 정하고 코드를 작성하게 되면 가독성이 높아져 다른 개발자가 소스 코드를 처음 보더라도 더 빠르고 확실하게 이해할 수 있게 된다. 따라서 규칙을 정하고 지킴은 개발자 간 반드시 지켜야 할 약속임을 항상 인지해야 한다.

**2. 공통법칙**

**2-1. 작업 디렉토리**

프로젝트-> 패키지 -> 모듈-> 클래스 형태로 형성한다.

**2-2. 소스파일**

1) 각 자바 소스 파일은 하나의 Public Class 혹은 Public Interface를 포함한다.

2) 하나의 Method 혹은 Class에서 사용될 객체의 선언 및 생성은 상단에 위치시킨다.

**3. 명명법**

**3-1. 공통규칙**

두 개의 단어로 이뤄져 있는 고유명사의 경우 두번째 단어의 시작문자를 대문자로 한다.

(ex) ClassNameMethod / addName() / inputOrder )

**3-2. 클래스명**

1) 단어의 첫 글자마다 대문자를 사용하도록 한다.

2)클래스의 이름은 반드시 명사여야 한다.

3) 약자로 나타내는 것은 반드시 피하고, 전체 이름을 사용해야한다.

**3-3. 메소드명**

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) getIdNum() )

**3-4. 필드명**

1) 변수, 모든 인스턴스 등은 첫 글자는 소문자로 구성하고 이후의 각 단어의 시작 문자는 대문자로 해주어야 한다.

2) 클래스 변수의 이름은 타입의 이름과 동일하게 지정한다.

4) 일반적으로 i, j, k, 등은 알아보기 힘든 변수명은 지정하지 않는다.

(ex) String i; : 변수명으로 어디서 사용될지 모름.)

**4. 주석문**

**4-1. 공통 주석문**

시작과 클래스의 주석문은 사용하지 않는다.

**4-2. 멤버 필드 주석문**

멤버 필드 주석문은 다음과 같이 라인 단위 “ // “ 주석으로 한다.

ex) public int appletree = 0; // 사과 나무수

**4-3. 멤버 메소드 주석문**

멤버 메소드의 주석문은 메소드 이름, 작성자의 형식을 따르고 매개변수, return값 및 간단한 설명은 필요에 따라 작성한다.

**4-4. 기타 주석문**

코드 작성 중 설명이 필요한 부분에 경우 라인 단위 또는 멀티 라인 단위로 기입하도록 한다.

코드 수정시 각자 어느 부분을 수정하였는지 확인하기 위해 수정 부분의 아래위로 주석 기입한다.

//nhj

Public void add()

//nhj

**5. 기타 스타일**

**5-2. 선언**

**5-2-1. 패키지 / 임포트 선언**

필요에 따라 package 문이 나타날 수 있다. Package 문 이후엔 한 줄을 건너 띄우고 다음으로 import 문이 나오도록 한다. import의 경우 한 줄에 하나의 import만을 명시한다.

ex)

package controller

import java.io.IOException;

import java.io.InputStream;

**5-2-2. 클래스 / 인터페이스 선언**

1) 인터페이스, 공통으로 사용할 클래스는 com.victoree2.common 패키지에 만든다.

2) main 함수는 com.victoree2.main 패키지에 만든다.

3) 시스템 클래스(백단으로 사용하느)는 com.victoree2.system 패키지에 만든다.

**5-3. 공백**

\* 조건/제어문의 키워드 (if, while, return, switch, for등)와의 관계

1) 시작 시 ‘ ( ‘ , ‘ { ‘ 의 괄호들과 빈칸을 두지 않고 작성한다.

2) 끝날 시 ‘ ) ‘ , ‘ } ‘ 의 괄호들 뒤에는 어떠한 코드도 작성하지 않고 밑줄로 넘어간다.

**5-4. 제어문 및 반복문**

**5-4-1. IF**

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**

For 문에서 '{'는 같은 줄에, '}'는 새로운 줄에 위치한다.

ex)

int i;

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

statement;

}

**5-4-3. Try / Catch**

try / catch 문에서 '{'는 같은 줄에, '}'는 새로운 줄에 위치한다.

ex)

try{

statement;

}catch (IOException ioexception){

statement;

} finally {

statement;

}

**5-5. Scanner scan**

Scan은 반드시 메서드 안에서 사용.

**5-6 selectMenu() == -1**

리턴값이 -1은 메뉴 기능에서 뒤로가기 기능으로 사용.

Ex)

While((key = selecetMenu()) != -1){

Switch(key) ~~~

}