Coding Convention

Team DooBaeLee

Contents

1. Java language rules

A. 예외를 무시하지 말기

절대 예외를 무시하면 안된다. 만약 예외를 무시하면 코드에 문제가 생기게 되고 그 문제를 코드를 다 짜고 나서는 찾기가 굉장히 힘들어 지기 때문에 예외처리를 제때제때 해주는 것이 중요하다.

Ex)

```
 void setServerPort(String value) {
 try {
        serverPort = Integer.parseInt(value);
 } catch (NumberFormatException e) {
          throw new RuntimeException("port " + value " is invalid,
", e);
            }
 }
```

B. 예외를 일반화 형태로 처리하지 말기

거의 모든 상황에서 한방에 예외를 처리하거나 Throw 처리하는 것은 부적절하다. 왜냐면 일반적인 예외 뿐만 아니라 오류도 포함될 수 있기 때문이다. 이 경우 굉장히 위험하다. 그러므로 하나의 try 문 뒤에 각각의 예외에 대하여 처리하는 여러개의 catch 블럭으로 예외처리를 해야한다.

C. Finalize를 사용하지 말것

자바 문법인 안드로이드 프레임 워크에도 finalize() 메소드가 있는데 이때 객체가 소멸되기 직전에 호출되며, 원칙상으로 소멸되기 직전의 일련의 작업을 수행하고자 할때 finalize()에 코드를 작성하면 된다. 하지만 finalize() 메소드는 자바에서도 오버라이딩을 권장하지 않는 메소드이고 안드로이드에서도 마찬가지다. 즉 언제 finalize 가 호출될지 보장할 수 없으며 호출될지도 장담할 수 없다. 그러므로 되도록이면 쓰지 않고 만약 절대적으로 필요한 경우에만 정확하게 문서화를 해놓고 써야한다.

D. Import 문은 전체를 다 사용하기

간단히 import문에 * 를 사용하지 말라는 뜻이다.

2.Java Style Rules

A. Java Doc 표준을 준수하기

Copyright는 문서 상단에 기술하고 그 아래 package, import, class 또는 interface 순으로 기술하며 각 블럭은 빈 줄로 구분 그 아래 클래스나 인터페이스는 어떤 기능을 하는지 Javadoc commentfh 기술한다.

B. 메소드를 짧게 작성하기

메소드는 40줄이 넘지 않게 짧게 작성하는것이 좋다. 40줄이 넘어간다면 설계가 잘못된 것일 수도 있다.

C. 멤버변수를 정해진 위치에 선언하기

Fields 선언은 표준위치에 즉, 초기에 선언하기나 바로 전에 선언하자.

D. 변수의 스코프를 최소화하기

지역변수의 범위를 최소화하자. 읽기도 좋고 관리도 편해진다.
지역변수는 브레이스 블럭 내부에서 사용해라.

E. Import 문의 순서에 신경쓰기

Import 문의 그룹핑 순서는
첫번째 Android import
두번째 3rd party import(com, junit, net, org)
세번째 java and javax
이며 각 그룹은 알파벳 순으로 정렬되어 있어야 한다.
그리고 각 그룹핑 사이에는 빈 공백 라인이 한 줄 들어가야 한다.

F. Tab 대신 Space로 들여쓰기를 하기

4 개의 스페이스를 들여쓰기로 사용하고 있다. 탭의 사용을 금지하기 때문이다.

G. 멤버 변수명 규칙

Non-public, non-static field 는 m으로 시작

Static field 는 s로 시작

나머지는 소문자로 시작

Public static final fields 는 모두 대문자로 쓰기.

Ex)

```
public static final int SOME_CONSTANT = 42;
public int publicField;
private static MyClass sSingleton;
Int mPackagePrivate;
private int mPrivate;
protected int mProtected;
```

H. 표준 중괄호 스타일에 따르기

시작 중괄호는 "{" 따로 한줄을 차지하지 않는다.

Ex)

```
class MyClass {
    int func() {
        if (something) {
        // ...
        } else if (somethingElse) {
        // ...
        } else {
        // ...
        }
    }
}
```

3. 로그에 대해서

로깅은 성능에 큰 영향을 미치기 때문에 먼저 로그 레벨을 이해해야한다.

ERROR: 항상 기록되는 레벨이므로 조심스럽게 사용해야한다. 먼저 치명적인 상황을 기록한다.

WARNING: 이것 또한 항상 기록되는 레벨이며 심각한 상황을 기록한다. 예로 사용자가 인식할 수 없는 오류결과가 나타나고, 사용하던 데이터가 날라가버리는 등이 있다.

DEBUG: 디버그모드에서만 기록되는 레벨로 디버깅관련과 추가 정보를 기록할 때 사용한다. 로깅과 관련한 소스가 컴바일 되기 때문에 성능을 고려하여 if(LOCAL_LOG) or if(LOCAL_LOGD) 구문내에 구현해야 한다.

VERBOSE: 뭐든 로깅해도 되는 level 이지만, debug level 처럼 반드시 if 구문내에 구현해야한다.

4. JavaTests 스타일 규칙

Test<MethodName>_<TestName>형식으로 작성한다.

Ex)

testMethod_specificCase1

testMethod_specificCase2

```
void testIsDistinguishable_protanopia() {
    ColorMatcher colorMatcher = new
ColorMatcher(PROTANOPIA)
    assertFalse(colorMatcher.isDistinguishable(Color.RED,
Color.BLACK))
    assertTrue(colorMatcher.isDistinguishable(Color.X, Color.Y))
}
```