개발 규칙 가이드

1. 포매팅

1.1 파일 인코딩 UTF-8 사용

1.2 들여쓰기

- tab 문자 대신 space 문자 4개 사용

1.3 시작하는 중괄호는 새로운 라인에서 시작하지 않고 같은 라인을 사용

```
if (value == true) {
    System.out.println("true");
}
```

1.4 띄어쓰기

- 메소드 이름 다음에는 띄어쓰기 X

```
foo(i, j);
```

- 배열 다음에는 띄어쓰기 X

```
args[0];
```

- 이진 연산자 간에 양쪽 띄어쓰기 사용

```
a = b + c;
```

- 쉼표와 세미콜론 뒤에는 띄어쓰기 사용

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

- cast 사용시 띄어쓰기 없이 작성

```
(MyClass)v.get(3);
```

- if, while, for, switch, catch 문 뒤에는 띄어쓰기 사용

```
if (value)

while (i < 10)

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

catch (Exception e)
```

1.5. 클래스 멤버 정렬 순서

클래스 멤버는 필드(전역변수), 생성자, 메서드 순서로 정렬

```
class MyClass {
    // fields

    // constructors

    // methods
}
```

1.6 라인 최대 길이

- 한 줄에 100자 이내

2. 선언

2.1 한 줄당 선언문의 수

한 줄에 하나의 선언문 사용

```
int x; -> O
int y; -> O
int x, y; -> X
```

2.2 초기화

지역 변수의 경우 선언할 때 초기화(변수의 초기화 값이 다른 계산에 의해서 결정되는 경우 제외

2.3 배치

선언은 블록의 시작에 위치

상위 블록에서 선언된 동일한 변수 이름을 선언 x

```
void myMethod() {
    int x = 0;    // 메서드 블록의 시작

    if (condition) {
        int y = 0;    // if 블록의 시작
    }
}

// 예외
for (int i = 0; i < x; i++) { ... }

int count;
...
myMethod() {
    if (condition) {
        int count = 0;    // 상위 블록에서 선언된 것을 이렇게 사용 X
    }
}
```

2.4 클래스, 인터페이스 선언

- 메소드 이름과 그 메서드의 파라미터 시작 괄호 “(” 사이에는 공백 x

- 여는 중괄호 “{”는 클래스, 인터페이스, 메서드 선언과 동일한 줄의 끝에 사용
- 닫는 중괄호 “}”는 여는 문장과 동일한 들여쓰기를 하는 새로운 줄에 사용

```
class Sample extends Object {
    int ivar1;
    int ivar2;

    Sample(int i, int j) {
        ivar1 = i;
        ivar2 = j;
    }

    int emptyMethod() {}

    ...
}
```

3. 문(Statements)

3.1 간단한 문

각각의 줄에는 하나의 문(statement)만 사용

```
x++; -> O
y--; -> O
x++;y--; -> X
```

3.2 복합문

- 중괄호로 둘러싸여진 문들은 복합문보다 한 단계 더 들여쓰기
- 여는 중괄호 “{”는 복합문 시작 줄 마지막에 위치
- 닫는 중괄호 “}”는 새로운 줄에 사용, 복합문의 시작과 같은 들여쓰기

3.3 return 문

return 문은 특별한 방법으로 값을 표현하는 경우를 제외하고는 괄호 사용 x

```
return;
return myClass.method();
return (isTrue ? true : false);
```

3.4 if, if-else, if else-if else 문

- if-else 문을 사용할 때는 다음과 같이 작성

```
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}
```

```
if (condition){
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

- if 문은 항상 중괄호 사용

3.5 for 문

for 문은 다음과 같이 작성

```
for (initalization; condition; update) {
    statements;
}
```

3.6 while 문

while문은 다음과 같이 작성

```
while (condition) {
    statements;
}

do {
    statements;
} while (condition);
```

3.7 switch 문

switch 문은 다음과 같이 작성

```
switch (condition) {
case A:
    statements;
    break;

case B:
    statements;
    break;
```

```
case C:
    statements;
    break;

default:
    statements;
    break;
}
```

3.8 try-catch 문

try-catch 문은 다음과 같이 작성

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
}
```

4. 명명 규칙

4.1 패키지

패키지 이름은 도메인 이름으로 작성

4.2 클래스, 인터페이스

- 클래스, 인터페이스 이름은 명사로 작성
- 첫 글자는 대문자, 나머지는 소문자 사용
- 복합 단어일 경우 각 단어의 첫 글자는 대문자

4.3 메서드

- 메서드 이름은 동사로 작성
- 메서드 이름의 첫 번째 문자는 소문자를 사용하고 복합 단어일 경우 첫 단어는 소문자로 시작하고 이후에 나오는 단어의 첫 문자는 대문자 사용

4.5 변수

- 변수 이름의 첫 번째 문자는 소문자로 시작하고 아휴에 나오는 단어의 첫 문자는 대문자 사용
- 변수 이름은 사용 의도를 알 수 있도록 작성

4.6 상수

상수들의 이름은 모두 대문자로 쓰고 각각의 단어는 언더바 “_” 로 분리

```
static final int MIN_WIDTH = 4;
static final int MAX_WIDTH = 999;
```

5. Git 협업 가이드

Repository는 Upstream Repository, Origin Repository, Local Repository로 구성

Upstream Repository: PM이 생성한 Repository로 팀원이 공유하는 최신 소스코드가 저장된 원격 저장소

Origin Repository: Upstream Repository를 Fork한 원격 개인 저장소

Local Repository: 본인 컴퓨터에 저장되어 있는 개인 저장소

5.1 워크플로우

- Upstream Repository를 Origin Repository로 fork

```
안드로이드 repository 주소
https://github.com/SeolMyeongJae/GBT-Android

백엔드 repository 주소
https://github.com/SeolMyeongJae/GBT-BackEnd
```

- Origin Repository를 Local Repository로 clone

```
안드로이드
git clone https://github.com/[fork한 본인 Origin Repository]/GBT-Android.git

백엔드
git clone https://github.com/[fork한 본인 Origin Repository]/GBT-BackEnd.git
```

- Local Repository에서 새로운 Branch를 생성해서 작업

```
git branch -c [브랜치명]
```

- 작업을 완료한 Branch를 commit 후 Origin Repository로 push

```
git commit -m '커밋메세지'
git push origin [브랜치명]
```

- push한 Branch를 Upstream Repository로 Merge하는 Pull Request 생성

- 생성된 Pull Request 코드를 확인 후 Merge
- Merge된 Upstream Repository를 Local Repository로 pull

```
git pull upstream main
```

- pull한 Local Repository를 Origin Repository로 push

```
git push origin main
```

5.2 Commit 규칙

- 추가, 수정, 삭제 내용을 커밋 메세지에 명확하게 작성

5.3 Branch 규칙

- 도메인 이름(안드로이드는 화면이름)으로 Branch 생성