Code Conventions for the Java Programming Language

- description : 7 - Statements
- author : 오션
- email : shlim@repia.com
- lastupdate : 2022-04-22 Fri

7. 명령문(Statements)

7.1 단순 명령문(Simple Statements)

각 라인에는 최대 하나의 명령문만 포함해야 합니다. 예를 들어,

```
argv++;           // Correct
argc--;           // Correct
argv++;  argc--;  // Avoid
```

7.2 복합 명령문(Compound Statements)

복합 명령문은 중괄호 "**{ 명령문 }**"으로 묶인 명령문 목록이 포함된 명령문입니다. 예제는 다음 섹션을 참조하십시오.

* 중괄호로 묶인 명령문은 복합 명령문보다 한 수준 더 들여써야 합니다. * 여는 중괄호는 복합 명령문을 시작하는 줄 끝에 있어야 합니다; 닫는 중괄호는 줄을 시작하고 복합 명령문의 시작 부분으로 들여쓰기해야 합니다. * 중괄호는 **if-else** 또는 **for** 문과 같은 제어 구조의 일부인 경우, 단일 명령문을 포함하는 모든 명령문에 사용됩니다. 이렇게 하면 중괄호 추가를 잊어버려 실수로 버그가 발생하지 않고 명령문을 더 쉽게 추가할 수 있습니다.

7.3 리턴 문(return Statements)

값이 있는 **return** 문은 어떤 식으로든 반환 값을 더 명확하게 하지 않는 한 괄호를 사용해서는 안 됩니

다. 예를 들어

```
return;

return myDisk.size();

return(size ? size : defaultSize);
```

7.4 if, if-else, if else-if else Statements

if-else 문의 클래스는 다음의 형식을 가져야 합니다.

```
if (condition) {
    statements;
}


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


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

Note: if 문은 항상 중괄호 **{}**를 사용합니다. 오류가 발생하기 쉬운 다음 형식을 피하십시오.

```
if (condition)    // 중괄호 { }를 생략하는 표기 방식을 피하세요
    statements;
```

7.5 for 문

for문은 다음의 형식을 가져야 합니다.

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

빈 **for**문(초기화, 조건 및 업데이트 절에서 모든 작업이 수행되는 문)은 다음 형식을 가져야 합니다.

```
for (initialization; condition; update);
```

for 문의 초기화 또는 업데이트 절에서 쉼표 연산자를 사용할 때, 3개 이상의 변수를 사용하는 복잡성을 피하십시오. 필요한 경우 **for** 루프(초기화 절의 경우) 이전에 또는 루프의 끝(업데이트 절의 경우)에 별도의 문을 사용합니다.

7.6 while Statements

while문은 다음의 형식을 가져야 합니다.

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

비어있는 **while**문은 다음의 형식을 가져야 합니다.

```
while (condition);
```

7.7 do-while 문

do-while은 다음의 형식을 가져야 합니다.

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

7.8 switch 문

switch문은 다음의 형식을 가져야 합니다.

```
switch (condition) {
case ABC:
   statements;
   /* falls through */
case DEF:
    statements;
    break;
case XYZ:
    statements;
    break;
default:
    statements;
    break;
}
```

(**break** 문이 없는) 하나의 케이스를 통과할 때마다, **break**문이 정상적으로 있을 위치에 주석을 추가합니다. 이것은 **/* falls through */** 주석이 있는 앞의 코드 예제에 나와 있습니다.

모든 **switch** 문에는 기본 케이스가 포함되어야 합니다. 기본 케이스의 **break**은 중복되지만 나중에 다른 케이스가 추가되는 경우 폴스루 오류를 방지합니다.\

7.9 try-catch 문

try-catch 문의 형식은 다음과 같아야 합니다.

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

try-catch 문 다음에 **finally**가 올 수도 있으며, 이는 **try** 블록이 성공적으로 완료되었는지 여부에 관계없이 실행됩니다.

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

Ref Site

Code Conventions for the Java Language 7 - Statements

오션,, Statements

From:
http://125.132.25.164/dokuwiki/ - **문제를 잘 정의하는 것은 문제를 절반 해결한 것이다. - 2023.12**

Permanent link:
http://125.132.25.164/dokuwiki/doku.php?id=wiki:miscellaneous:code_conventions:statements&rev=1650841664

Last update: **2022/04/25 08:07**