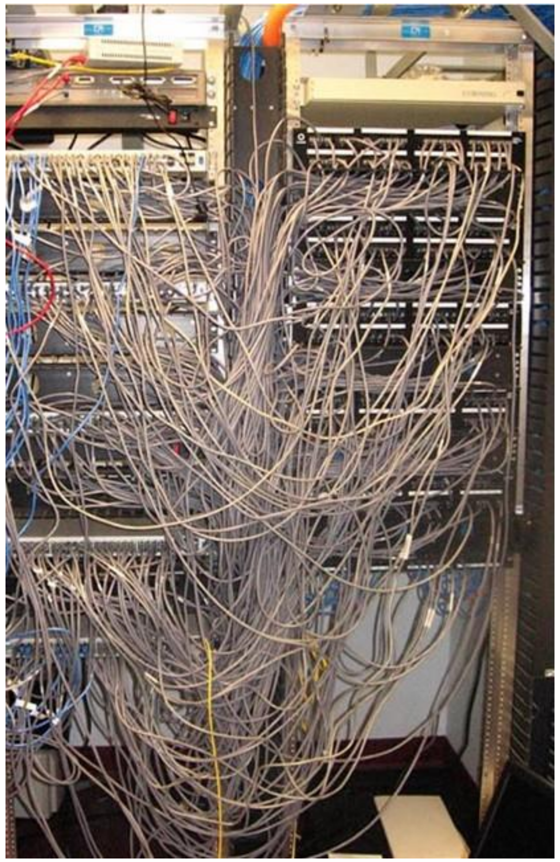
PEMROGRAMAN LANJUT

Code Convention

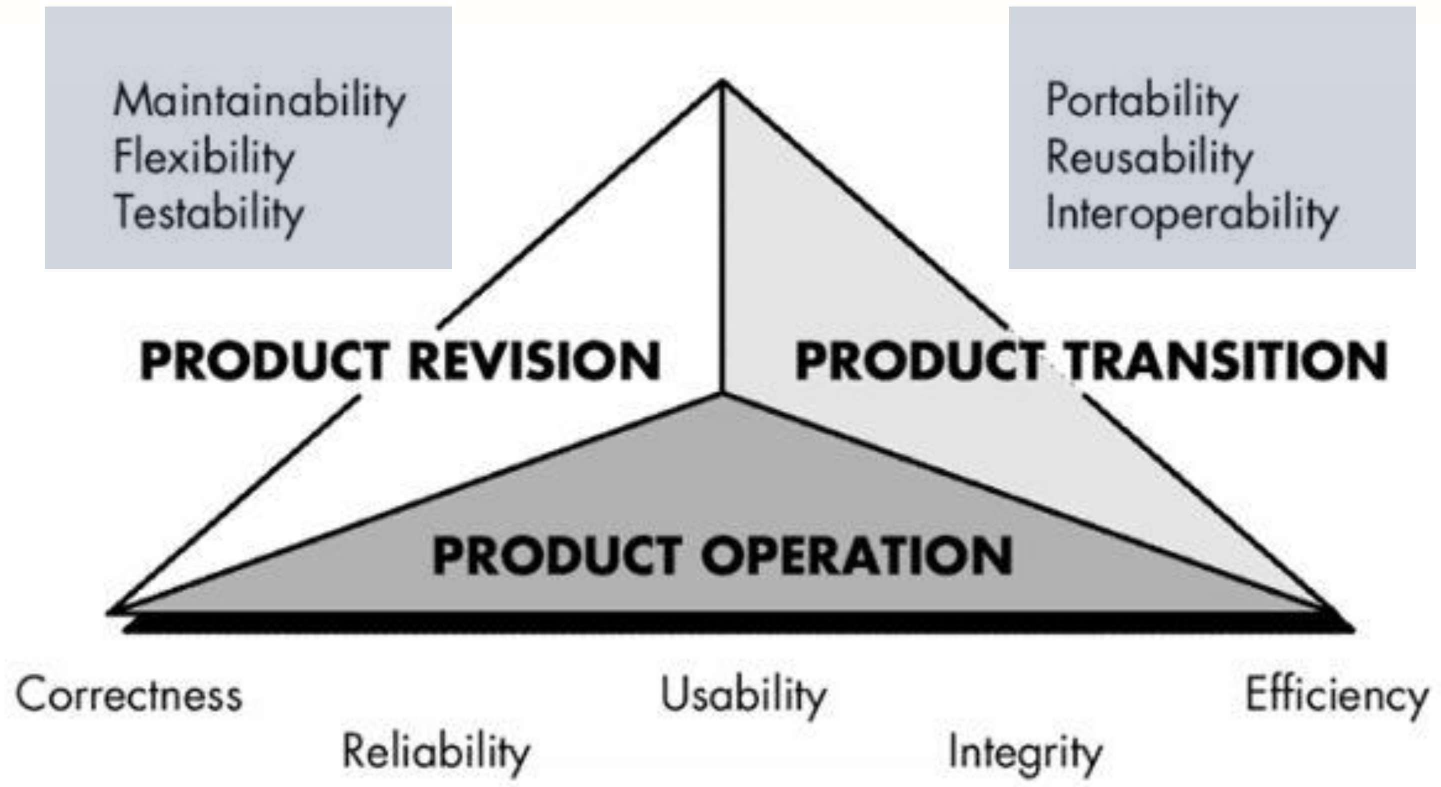
Oleh Politeknik Elektronika Negeri Surabaya
2021

Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Review


Bad Code

Clean Code



Mc Call Software Quality Metric

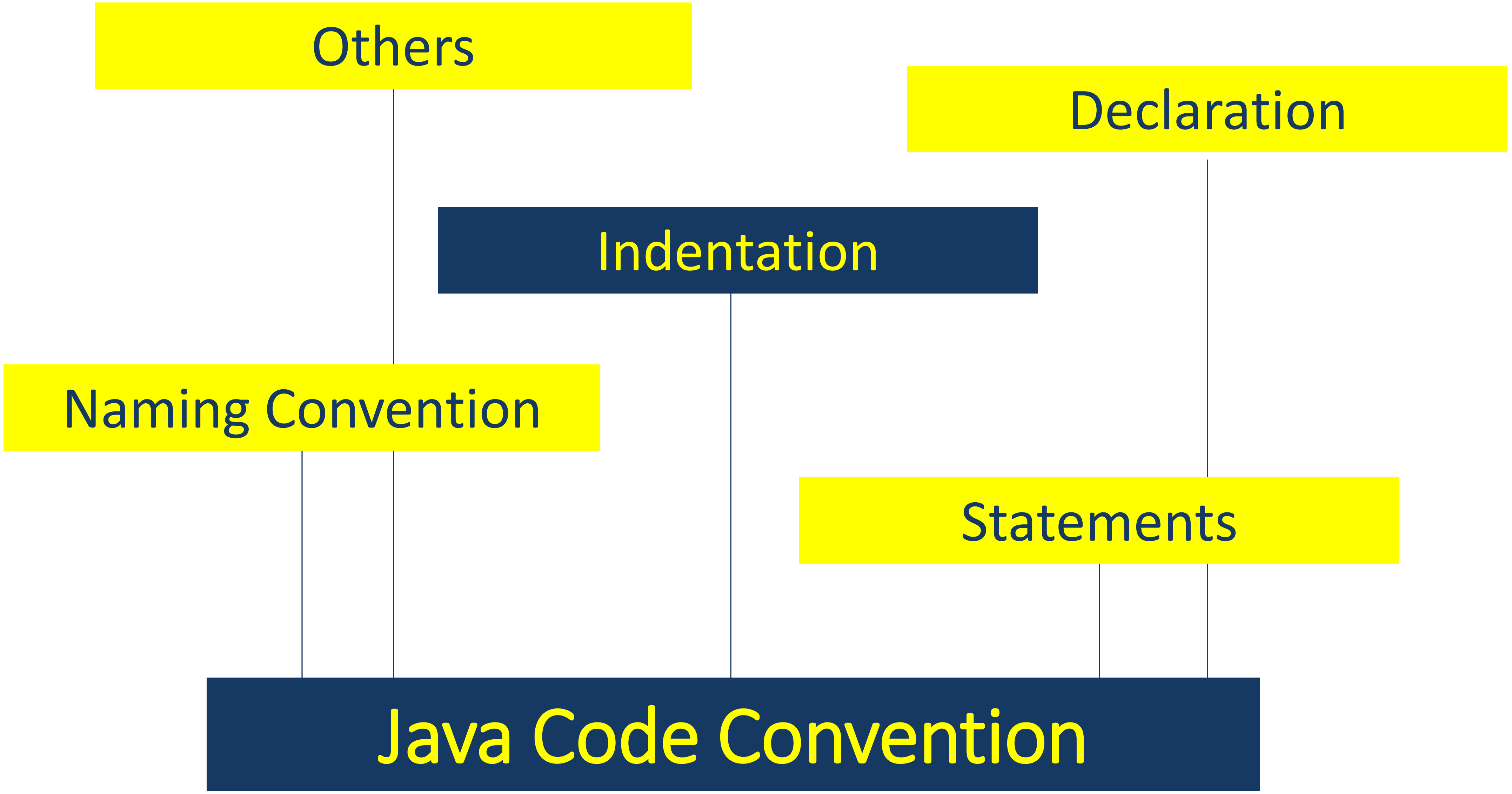
Naming Convention

1. Use Intention-revealing Name
2. Avoid Disinformation
3. Make Meaningful Distinctions
4. Use Pronounceable Name
5. Use Searchable Name
6. Avoid Encoding
7. Do not be Cute
8. Pick One Word per Concept
9. Do not Pun
10. Use Problem and Solution Domain Name
11. Do not Add Gratuitous Context
12. Add Meaningful Context

End of Review

Code Conventions

Set of guidelines for a specific programming language that recommend programming style, practices, and methods for each aspect of a program written in that language.


Others
Declaration
Indentation
Naming Convention
Statements
Java Code Convention

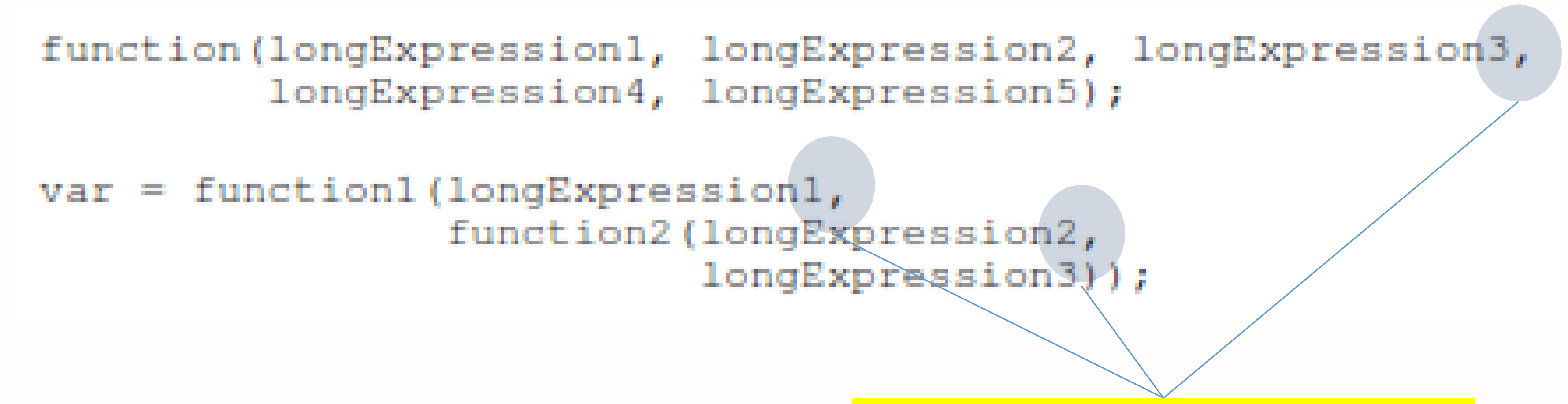
Indentation: Line Length

- Don’t restrict yourself to 80-character lines. Google’s Android style guide suggests 100-character lines, which is also the default setting in Android Studio.

Indentation: Wrapping Lines

- Break after a comma.
- Align the new line with the beginning of the expression at the same level on the previous line.
- Break before an operator.
- If the above rules (point 2) lead to confusing code or to code that's squished up against the right margin, just indent 8 spaces instead.

Indentation: Wrapping Lines

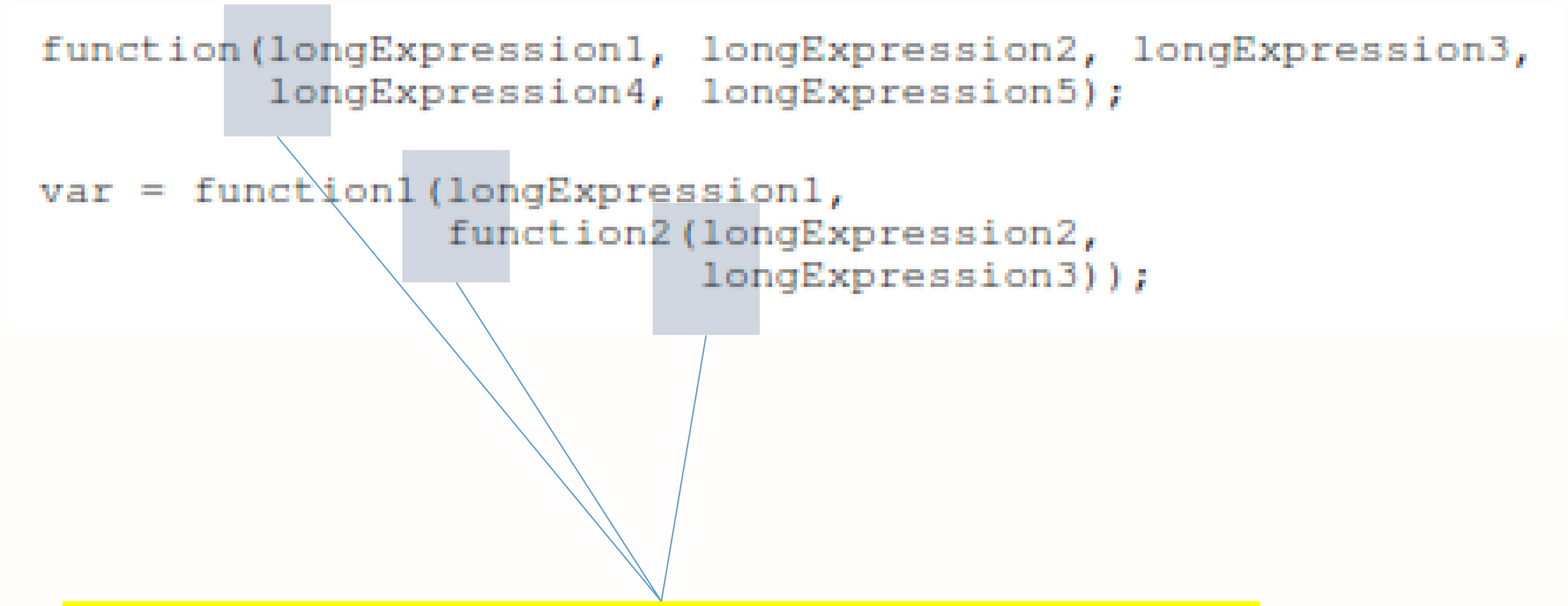


```
function(longExpression1, longExpression2, longExpression3,
         longExpression4, longExpression5);

var = function1(longExpression1,
                function2(longExpression2,
                          longExpression3));
```

Break after comma

Indentation: Wrapping Lines

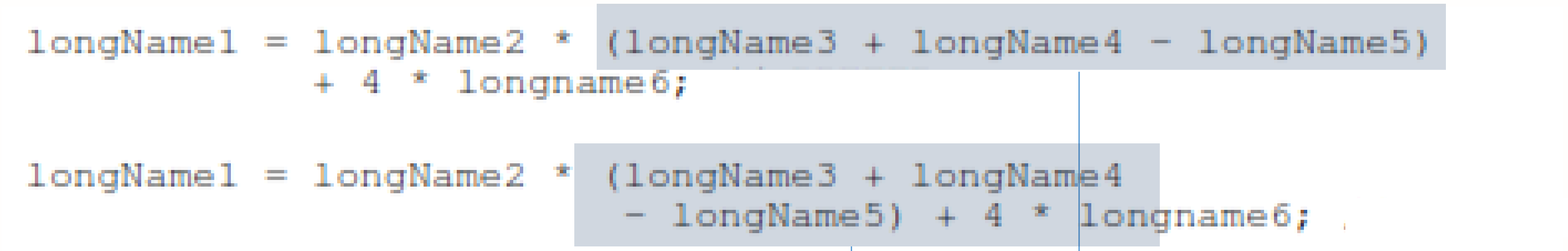


```
function(longExpression1, longExpression2, longExpression3,
         longExpression4, longExpression5);

var = function1(longExpression1,
                function2(longExpression2,
                          longExpression3));
```

Align the new line with the beginning of the expression at the same level on the previous line

Indentation: Wrapping Lines

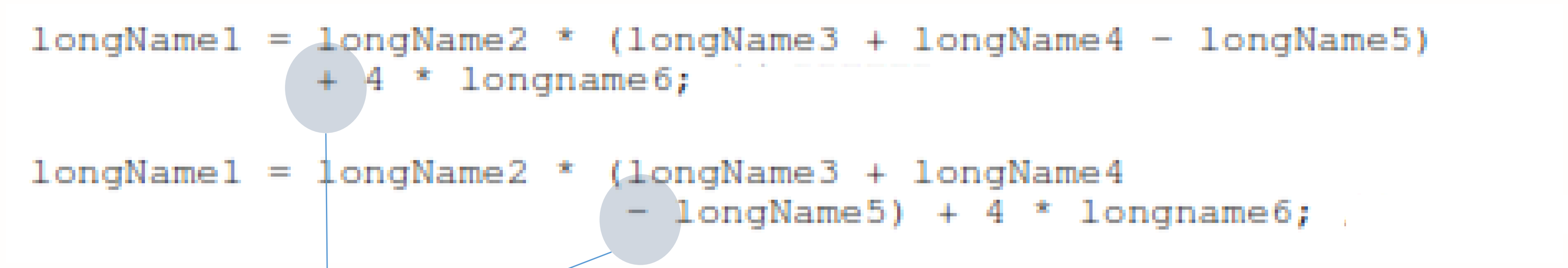


```
longName1 = longName2 * (longName3 + longName4 - longName5)
            + 4 * longname6;

longName1 = longName2 * (longName3 + longName4
                         - longName5) + 4 * longname6;
```

The first example is preferable, because the break occurs outside the parenthesized expression

Indentation: Wrapping Lines



```
longName1 = longName2 * (longName3 + longName4 - longName5)
            + 4 * longname6;

longName1 = longName2 * (longName3 + longName4
                         - longName5) + 4 * longname6;
```

Break before an operator

Indentation: Wrapping Lines

```
//CONVENTIONAL INDENTATION
someMethod(int anArg, Object anotherArg, String yetAnotherArg,
           Object andStillAnother) {
    ...
}

//INDENT 8 SPACES TO AVOID VERY DEEP INDENTS
private static synchronized horkingLongMethodName(int anArg,
        Object anotherArg, String yetAnotherArg,
        Object andStillAnother) {
    ...
}
```

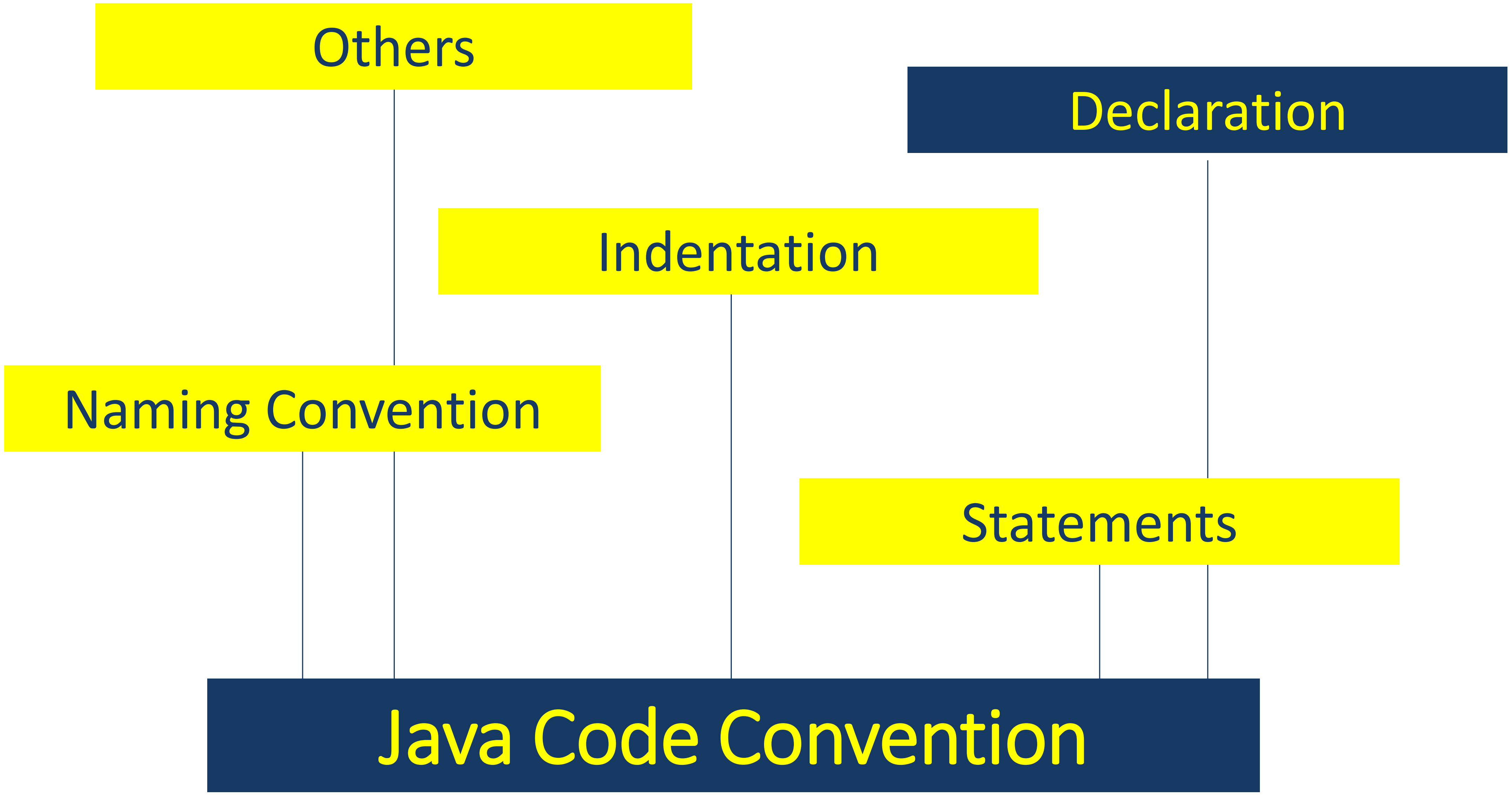
The first example presents conventional indentation which align the new line with the beginning of the expression at the same level on the previous line. Thus, it can make a very deep indent.

Indentation: Wrapping Lines

```
//CONVENTIONAL INDENTATION
someMethod(int anArg, Object anotherArg, String yetAnotherArg,
           Object andStillAnother) {
    ...
}

//INDENT 8 SPACES TO AVOID VERY DEEP INDENTS
private static synchronized horkingLongMethodName(int anArg,
        Object anotherArg, String yetAnotherArg,
        Object andStillAnother) {
    ...
}
```

To avoid a very deep indent, you can use 8 spaces to indent the new line as presented in the second example


Others
Declaration
Indentation
Naming Convention
Statements
Java Code Convention

Declaration: Number Per Line

- One declaration per line is recommended.
- Do not put different types on the same line.

```
int foo, fooarray[]; //WRONG!
```

Declaration: Placement

- Put declarations only at the beginning of innermost block that encloses all uses of the variable.
- A block is any code surrounded by curly braces "{" and "}".

```
void myMethod() {
    int int1 = 0;          // beginning of method block

    if (condition) {
        int int2 = 0;      // beginning of "if" block
        ...
    }
}
```

Declaration: Class and Interface

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

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

    int emptyMethod() {}

    ...
}
```

Open brace "{" appears at the end of the same line as the declaration statement

Closing brace "}" starts a line by itself indented to match its corresponding opening statement, except when it is a null statement the "}" should appear immediately after the "{"

Declaration: Class and Interface

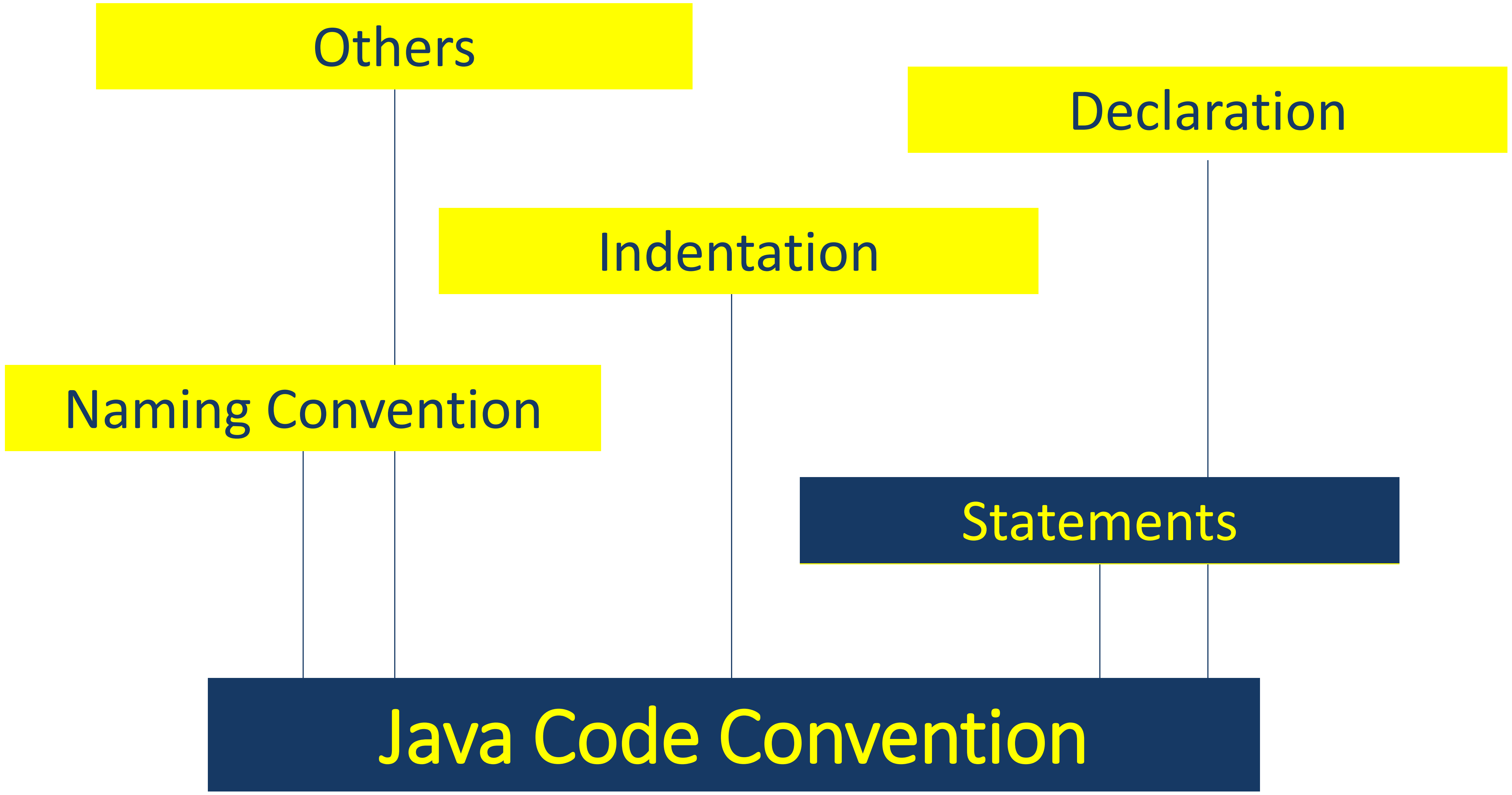
```
class Sample extends Object {
    int ivar1;
    int ivar2;

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

    int emptyMethod() {}

    ...
}
```

Methods are separated by a blank line


Others
Declaration
Indentation
Naming Convention
Statements
Java Code Convention

Statements

- Simple statement: Each line should contain at most one statement
- Compound statement: Lists of statements enclosed in braces

The enclosed statements should be indented one more level than the compound statement

Braces are used around all statements

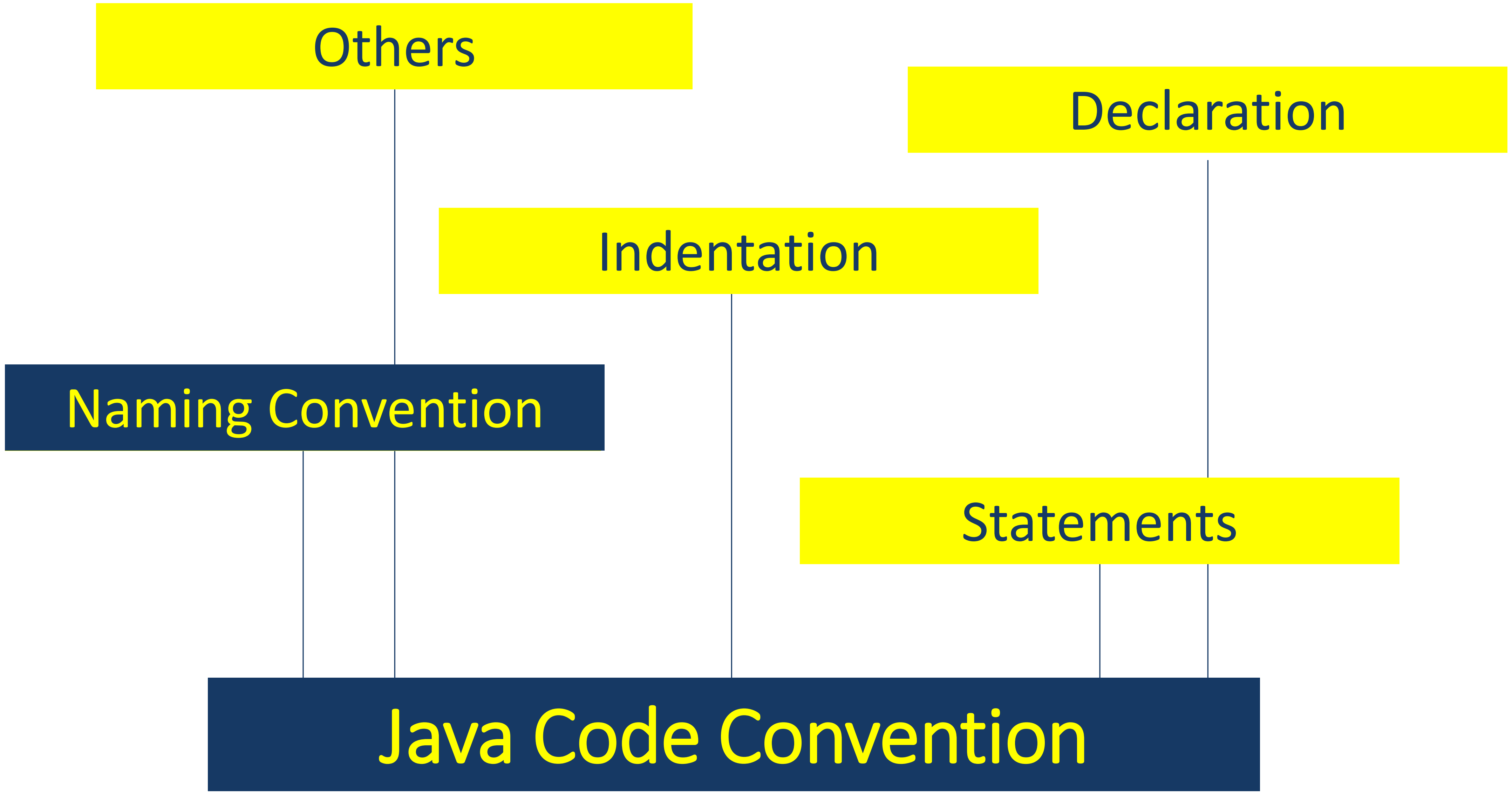
Statements: return Statements

- A return statement with a value should not use parentheses unless they make the return value more obvious in some way.

```
return;

return myDisk.size();

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


Others
Declaration
Indentation
Naming Convention
Statements
Java Code Convention

Naming Convention: Packages

- The prefix of a unique package name is always written in all-lowercase ASCII letters
- Example:

```
com.sun.eng
com.apple.quicktime.v2
edu.cmu.cs.bovik.cheese
```

Naming Convention: Classes

- Class names should be nouns, in mixed case with the first letter of each internal word capitalized.
- Try to keep your class names simple and descriptive.
- Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).
- Example:

```
class Raster;
class ImageSprite;
```

Naming Convention: Interface

- Interface names should be capitalized like class names.
- Example:

```
interface RasterDelegate;
interface Storing;
```

Naming Convention: Methods

- Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.
- Example:

```
run();
runFast();
getBackground();
```

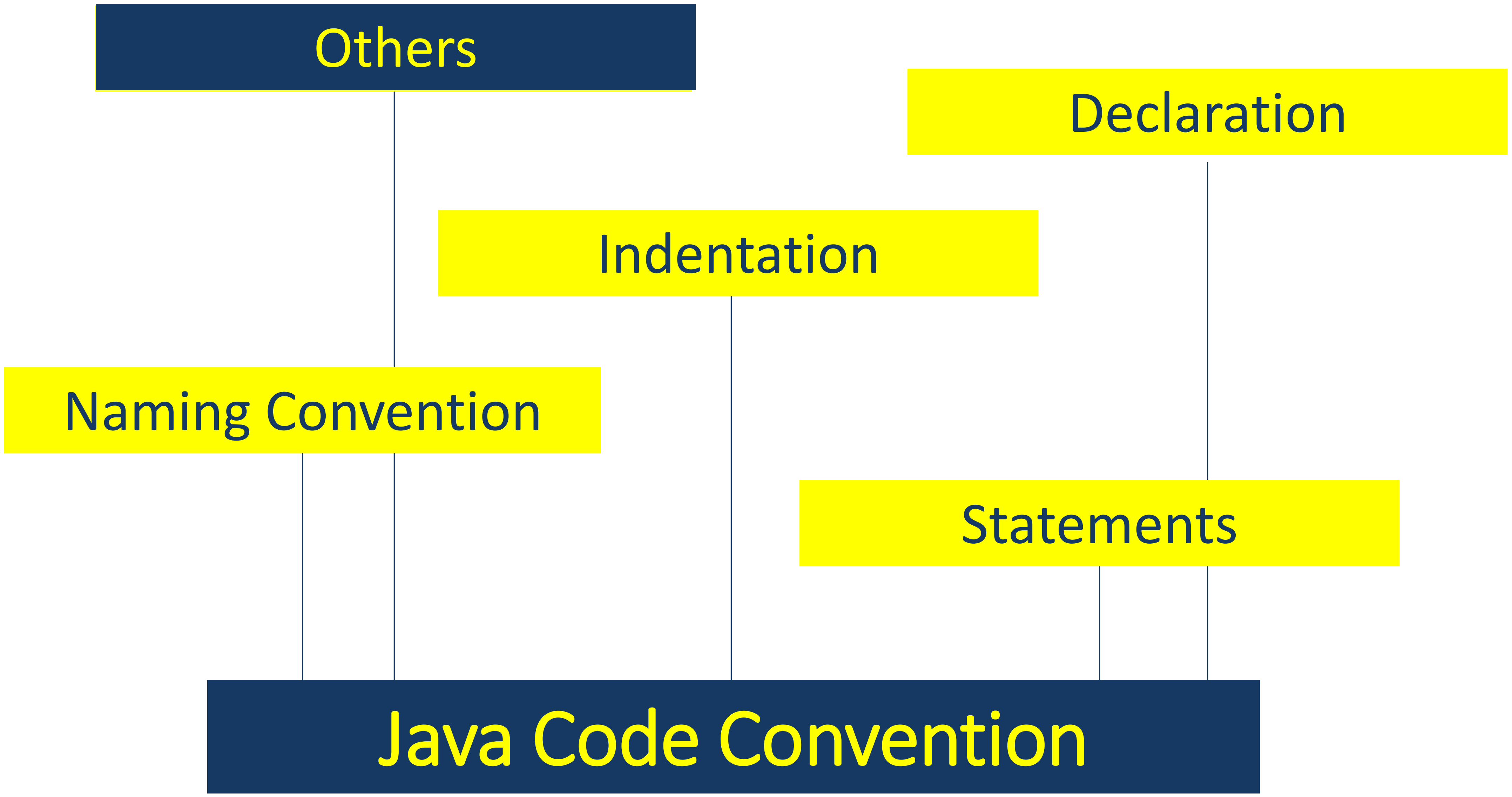
Naming Convention: Variables

- Variable are in mixed case with a lowercase first letter. Internal words start with capital letters.
- Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed.
- Variable names should be short yet meaningful.
- The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use.

Naming Convention: Constants

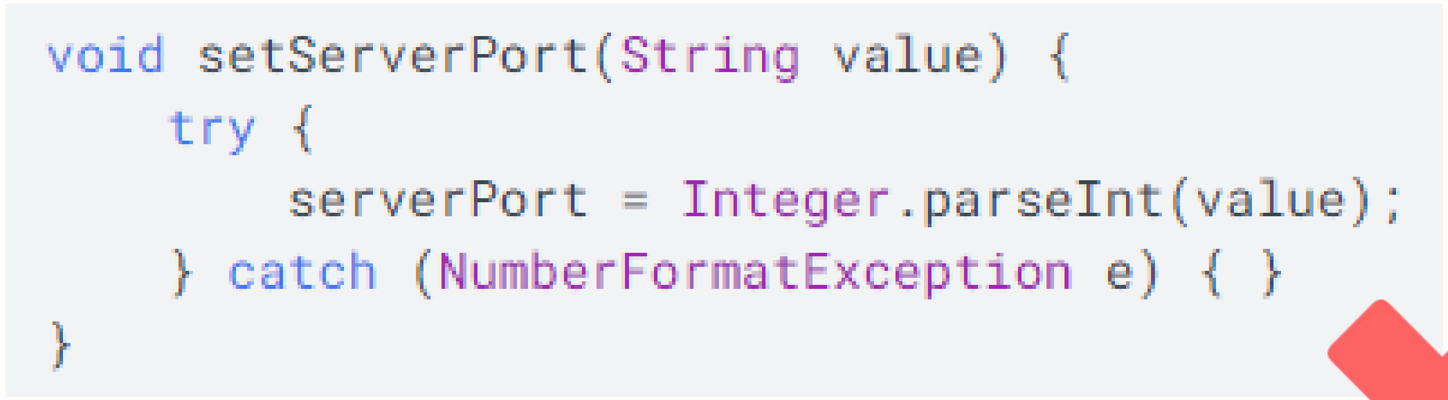
- Constant name should be all uppercase with words separated by underscores ("_").
- Example:

```
static final int MIN_WIDTH = 4;
static final int MAX_WIDTH = 999;
static final int GET_THE_CPU = 1;
```


Others
Declaration
Indentation
Naming Convention
Statements
Java Code Convention

Don't Ignore Exceptions

- While you may think your code will never encounter this error condition or that it isn't important to handle it, ignoring this type of exception creates mines in your code for someone else to trigger some day.



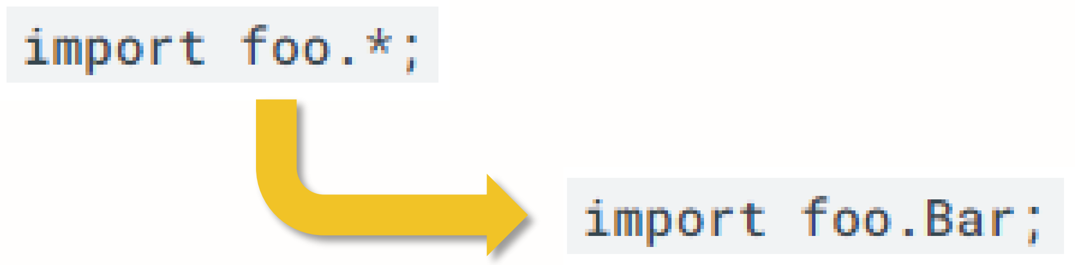
```
void setServerPort(String value) {
    try {
        serverPort = Integer.parseInt(value);
    } catch (NumberFormatException e) { }
}
```

Don't Catch Generic Exceptions

```java
try {
    someComplicatedIOFunction();          // may throw IOException
    someComplicatedParsingFunction();     // may throw ParsingException
    someComplicatedSecurityFunction();    // may throw SecurityException
    // phew, made it all the way
} catch (Exception e) {                   // I'll just catch all exceptions
    handleError();                        // with one generic handler!
}
```

- In most cases you shouldn't be handling different types of exceptions in the same way.

Don't Use Wildcard Imports



```
import foo.*;
```

```
import foo.Bar;
```

- Makes it obvious what classes are used and the code is more readable for maintainers.

References

- https://www.oracle.com/java/technologies/javase/codeconventions-contents.html
- https://firefox-source-docs.mozilla.org/code-quality/coding-style/index.html
- https://source.android.com/setup/contribute/code-style