

# Programming Guidelines

**Prof. Dr. Dirk Riehle**

**Friedrich-Alexander University Erlangen-Nürnberg**

**ADAP B01**

Licensed under [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/)

# Programming Guidelines

- Programming guidelines ...
  - are a set of rules and conventions that determine
    - naming,
    - formatting and
    - structuring
  - of source code and related artifacts
- Programming guidelines ...
  - make developers more effective
  - ease reading source code
  - should be mandatory

**Respect existing  
rules and conventions**

# Java Programming Guidelines [S97]














- We use the original Java guidelines: <http://goo.gl/hNaJsG>
  - While old, these are still the current guidelines
- Where those fail, we use Google's Java guidelines
  - See <https://google.github.io/styleguide/javaguide.html>






# File Names

- Stick to established file naming conventions
  - For example, file names should match main class name
- Stick to established suffixes (.java, .gradle, .war)
- Stick with established file names (README, LICENSE)

# File Organization

- Stick with folder structure provided as default (web apps)
- Separate source from test folder hierarchy

```
▼  src/main/java
  ▶  org.wahlzeit.agents
  ▶  org.wahlzeit.apps
  ▶  org.wahlzeit.handlers
  ▶  org.wahlzeit.main
  ▶  org.wahlzeit.model
  ▶  org.wahlzeit.model.persistence
  ▶  org.wahlzeit.services
  ▶  org.wahlzeit.services.mailing
  ▶  org.wahlzeit.servlets
  ▶  org.wahlzeit.tools
  ▶  org.wahlzeit.utils
  ▶  org.wahlzeit.webparts
```

```
▼  src/test/java
  ▶  org.wahlzeit.handlers
  ▶  org.wahlzeit.model
  ▶  org.wahlzeit.model.persistence
  ▶  org.wahlzeit.services
  ▶  org.wahlzeit.services.mailing
  ▶  org.wahlzeit.testEnvironmentProvider
  ▶  org.wahlzeit.utils
```

# File Content

- Beginning comments
  - Without the programmer name
- Package declaration
- Import statements
- Program code with comments
  - Class declaration
  - Field declarations
  - Method definition
  - ...

# File Content Example

## 3.1.1 Beginning Comments

All source files should begin with a c-style comment that lists the class name, version information, date, and copyright notice:

```
/*  
 * Classname  
 *  
 * Version information  
 *  
 * Date  
 *  
 * Copyright notice  
 */
```

...



# Naming Elements

- Nouns
  - Classes
  - Interfaces
  - Packages
  - Fields
- Verbs
  - Methods

# Order of Adjectives in the English Language

adjectives in English absolutely have to be in this order: opinion-size-age-shape-colour-origin-material-purpose Noun. So you can have a lovely little old rectangular green French silver whittling knife. But if you mess with that word order in the slightest you'll sound like a maniac. It's an odd thing that every English speaker uses that list, but almost none of us could write it out. And as size comes before colour, green great dragons can't exist.

# Naming Objects and Classes 1 / 2

quantity - opinion - size - age - shape - color - origin - material - purpose

- Class and interface names
  - File // no adjective
  - TempFile // adding age
  - BigTempFile // adding size
  - BigTempHotRedFile // adding “color”
  - BigTempHotRedBinaryFile // adding “material”
  - BigTempSequentialHotRedBinaryFile // adding shape
- Object names (variable/field names)
  - invalidBigTempSequentialHotRedBinaryFile // adding opinion
  - doublyInvalidBigTempSequentialHotRedBinaryFile // adding quantity

# Naming Objects and Classes 2 / 2

- An adjective can take the genitive form (but should rarely)
  - Usually only happens if the genitive form is an established domain term
    - Examples: FileOfBits vs BinaryFile, QuoteBook vs. BookOfQuotes
- Use of reserved adjectives (abstract, default, etc)
  - By default lead the noun (no other explanation but convention)
    - Examples: AbstractSequentialFile, DefaultHashMap
- Fields have an implied specifier (a, one, many, the, all)
  - It is often omitted for reasons of brevity
- More conventions in lectures on class and interface design

# For Sale: Baby shoes. Never worn.

"For sale:

Baby shoes. Never worn."

E. Hemingway

ErnestHemingwaysForSaleNeverWornBabyShoesUltraShortStory

# Naming Methods

- Methods are commands
  - Simplify, use active voice (e.g. prepare rather than bePreparedFor)
- Follow established patterns
  - Specifically, container(-like) classes have a fair number of conventions
    - Add and remove
    - Insert, prepend, append, remove
    - What is the difference between remove and delete?
- More conventions in lectures on method types and properties

# Code Formatting

- One statement per line
- Semantic line wrapping after defined length
- Use of consistent bracketing
  - Sun's guidelines don't tell you how
  - How to decide what to use?
- Indentation for readability
  - Use tabs not spaces

# Techniques for Avoiding Bugs

- Initialize fields where declared
- Declare variables only at beginning of blocks
- Bracket even single-statement blocks
- ...
- Consider using FindBugs or similar tools



# Techniques that Depend on Context

```
if (arg == null) {  
    return -1;  
}  
  
if (someCondition) {  
    return 1;  
}  
  
if (anotherCondition) {  
    return 2;  
}  
  
return 0;
```

```
int result = 0;  
  
if (arg == null) {  
    result = -1;  
} else if (someCondition) {  
    result = 1;  
} else if (anotherCondition) {  
    result = 2;  
} else {  
    result = 0;  
}  
  
return result;
```

# Techniques Too Smart for the Rest of Us

```
if (null == arg) {  
    doSomething();  
}
```

# Clear Commit Messages

[android](#) / [platform](#) / [packages](#) / [apps](#) / [GlobalSearch](#) / [592150ac00086400415afe936d96f04d3be3ba0c^!](#) / .

**commit** 592150ac00086400415afe936d96f04d3be3ba0c

[\[log\]](#) [\[tgz\]](#)

**author** Anonymous Coward <nobody@android.com>

Wed Aug 19 15:44:42 2009 -0700

**committer** Anonymous Coward <nobody@android.com>

Wed Aug 19 15:56:16 2009 -0700

**tree** [33a5517dcfba62a2652a5b809b71a6c9ff4514cd](#)

**parent** [500413e48c7b96743ea13357aa2fba856edd1c8c](#) [\[diff\]](#)

Remove the search engine setting. Hard-code the search engine to Google, using EnhancedGoogleSearchProvider if available, otherwise GoogleSearch if available, otherwise throwing a RuntimeException requiring one of these packages.

diff --git [a/res/xml/preferences.xml](#) [b/res/xml/preferences.xml](#)

index 25cd7f5..eae8034 100644

--- a/res/xml/preferences.xml

+++ b/res/xml/preferences.xml

@@ -20,16 +20,8 @@

<PreferenceCategory

android:title="@string/web\_search\_category\_title">

# Review / Summary of Session

- General programming guidelines
  - What are programming guidelines?
  - Why do we have them? What is their importance?
- Java programming guidelines
  - What are example guidelines?
  - How and where to read up for a complete set?
- Things not covered by guidelines
  - What naming conventions beyond guidelines are there?
  - What techniques? How and when to learn?

# Thanks! Questions?

**[dirk.riehle@fau.de](mailto:dirk.riehle@fau.de) – <http://osr.cs.fau.de>**

**[dirk@riehle.org](mailto:dirk@riehle.org) – <http://dirkriehle.com> – [@dirkriehle](#)**

# Credits and License

- Original version
  - © 2012-2018 [Dirk Riehle](#), some rights reserved
  - Licensed under a [Creative Commons Attribution 4.0 International License](#)
- Contributions
  - ...