Style Guide

Enrico Wenger

Generally

- The Java Code Conventions by Sun Microsystems, Inc. applies to all source files.

- The file format has to be UNIX compatible
- No spaces in the filenames(underscore is allowed)
- The maximum length of line is 79 character (generally no more than 70 characters)

Name convention

- Filename are the name of the public classes in this file(the suffix of that is .java)
- Class names are beginning with a capital letter
- Names of methods are beginning with a small letter
- Constance variable are complete in capital letter
- All names exist only one time; exception running variables

The Java source file

- Each Java Source file contains a single public class or interface
- Private classes and interfaces are associated with a public class, you can put them in the same source file as the public class
- The public class should be the first class or interface in the file.

- The source files have the following ordering
  - Beginning comments
  - Package and Import statements
  - Class and interface declarations

Beginning comments

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

- For further details, see "How to Write Doc Comments for Javadoc„

  (http://java.sun.com/products/jdk/javadoc/writingdoccomments.html)
- which includes information on the doc comment tags (@return, @param, @see)

Package and Import statements

- The first non-comment line of most Java source files is a **package** statement.
- After that, **import** statements can follow.

- For example:

  - package java.awt;
  - import java.awt.peer.CanvasPeer;

Class and interface declarations

- Class/interface documentation comment (/**...*/)
- class or interface statement
- Class/interface implementation comment (/*...*/), if necessary((This comments should contain any class-wide or interface-wide information that wasn't appropriate for the class/interface documentation comment)
- Class (`static`) variables (First the `public` `class` variables, then the `protected`, and then the `private`.)
- Instance variables (First `public`, then `protected`, and then `private`.)
- Constructors
- Methods

Wrapping Lines

When an expression will not fit on a single line, break it according to these general principles:

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

if-else, if else-if else Statements

The if-else class of statements should have the following form:

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

Example

Comment of the file

```
/**
 * Descpription: what the class do
 *
 * @version date
 * @author
 *         <b><href=``mailto:wenger@inf.tum.de``>Enrico Wenger</a></b><br>
 */
```

Public class

```
public class Example { /* A class implementation comment can go here. */
   private static int classAttrib;  // description
   private final int CONSTANT; // description

   /*
    * class description
    *
    * @param parameter description
    * @return returnvalue description
    */

   public int methodName(int param) {
      ...
   } // methodName
} // Example
```

Method

End of file

Thank you for your attention !