ParentApp Coding Style Guide

I. Naming Conventions

1.1 CamelCase

Java class, field, method, and variable names are written in CamelCase: Words are smashed together and the first letter of each word is capitalized. Constants are in all-caps with words separated by underscores. Constants should also have the keyword final, and they are usually static.

thisIsAnExample

XML UI elements ids can be named with underscores, however.

this_is_an_example

There is one exception to CamelCase: constants. Constants are in all-caps with words separated by underscores. Constants should also have the keyword final, and they are usually static:

THIS_IS_AN_EXAMPLE

1.2 Class and interface names

Class and interface names are generally noun or noun phrases and begin with a capital letter.

interface AqueousHabitat { ... }

class FishBowl implements AqueousHabitat { ... }

1.3 Method names

Method names generally begin with a lowercase letter and are generally verbs describing what they are trying to achieve.

setTitle()

The name of a boolean function is often a verb phrase starting with "is", thus describing what the value means.

isATriangle()

1.4 Variable names

Variable names generally start with a lowercase letter, and should be as mnemonic as possible, giving the reader a good idea what the meaning are.

II. Format conventions

2.1. Indentation and braces { }

Sub-statements of a statement or declaration should be indented. Only 1 statement on a line.

Open curly braces " { " should at the end of a line, and not on a line by itself. The closing brace " } " appears on its own line.

2.2 Always use braces for control flow structures

Put a space between the keyword and the following parenthesis, like this

```
if  (someBooleanValue){
        // do something
}
```

Always use curly braces with if else, even if they contain only one statement.

III. Comments

Has a class level comment for every class. Longer methods should also have a comment describing what they do.

Use single line comment // for single-line comments. Use block comments /** */ for comments that take several lines.

Don't over comment. Assume reader has basic understanding of Java.

IV. Code organization

4.1 Field and class variable declarations go at the beginning of a class

Place all field and class (static) variables at the beginning of a class, before all the methods. That is where a reader will look for them.

4.2 Use returns to simplify method structure

You may hear from some people that a method should return only in the last statement of the method, at the end of the method body. In some cases, that can lead to a messy, incomprehensible method body. Use of return statements in several places, along with appropriate use of assertions, can lead to a far more comprehensible method. Further, this use of returns is developed naturally during stepwise refinement. However, one must be disciplined with the use of the returns and assertions.

4.3 Put the shorter of the then-part and else-part first

4.4 Declare local variables close to first use

A local variable is a variable declared within a method body.

The tendency is to declare all variables at the beginning of the method body, and to start off inserting there all the variables you think you might need. Fight this tendency! It leads to a proliferation of variables that often are not needed, and it forces the reader to look at and think about variables at the wrong time.

Principle: Declare a variable as close to its first use as possible.