Gopher Coding Conventions

This document specifies some conventions for Java code in Gopher. Consider it a work in progress. All developers are encouraged to bring forth issues and we will add them here.

First, all should be aware of conventions that are considered part of the language. Here are a couple of links to Oracle:

<http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>

<http://www.oracle.com/technetwork/java/codeconventions-135099.html>

Here are some conventions not followed or issues I have seen in our code:

Naming –

* Class names begin with upper case and are then camel case – ‘MyClass’.
* Test case class names end with ‘\_Test’ – ‘Header\_Test’.
* Variables begin with lower case and are then camel case – ‘headerTest’.
* Constants are static final or enum – ‘static final String MY\_CONSTANT = foo;’
* Method names begin with lower case and then camel – ‘getSomething ()’.

Scope of members and methods –

* Members are private until needed by a subclass – ‘private Boolean success;’.
* Methods are private until need publicly – ‘private Boolean getStatus()’.

Volatility of members and methods –

* Methods are static unless they access instance member data.
* Members are static only if they need to be shared between class instances.

Packaging -

* All package names are lower case.
* All framework classes we develop reside in a package somewhere under ‘com.echostar.gopher’.
* All test cases we develop for DANY belong under ‘com.echostar.dany.suites’.

Blocks -

* All blocks must begin and end in the same column –

while (truth == beauty ) {

Assert.assertEquals (life.equals (good));

}

Or

if (up.equals ( down))

{

throw new Exception (“I am confused, up equals down.”);

}

It is especially important this rule be followed with nested blocks –

while (someCondition) {

While (anotherCondition) {

doSomething();

}

}

Long lines –

* Avoid having very long lines.
* Multiple method calls on the same line may be lengthy –

String blee = getA().getB().getC().getBlee();

* Consider assigning local variables rather than having several calls on the same line –

A a = getA();

B b = a.getB();

C c = b.getC();

String blee = c.getBlee();

This also makes the code easier to maintain and debug. If you get a null pointer exception in the previous example, where is the null pointer?

Equality –

* a==b means they are the same instance.
* a.equals (b) means they are equivalent.
* Use the correct kind of comparison.

Warnings –

* Resolve all warnings.
* In a few rare cases one of the annotations which suppress warnings may be appropriate, for example where an API method returns a ‘raw type’ –

Query query = hibernateSession.createQuery(

"FROM TestCase WHERE className=\'"+className+"\'");

@SuppressWarnings("rawtypes")

List results = query.list();

Tabs/Indentation –

* Use a consistent number of blank spaces when indenting. 4 is a good number.
* Configure Eclipse/Your favorite editor to insert white space for tabs.
* Then use the tab key.