Project Conventions Team Aardvark

COMP2541: Software Engineering
University Of Leeds

"Why did the functions stop calling each other?
Because they had constant arguments..."

Revision No. 1

February 26, 2016

Contents

1 Code Conventions

1.1 Code Style

For the sake of consistency (which is rather important), the code conventions will be as follows:

- Use of CamelCase when naming variables, methods, classes, and anything else
- All indentations are done using tabs as oppose to spaces.

An example to demonstrate the above:

```
def computeBill(self):
    """
    Calculates the total price of all the food ordered.

    :return: The total bill for the table.
    """
    totalBill = 0
    for food in self.orderHistory:
        totalBill += food.price
        return totalBill
```

1.2 Test Driven Development (TDD)

All team members are required to write tests. A TDD approach is preferable but it **isn't** mandatory although writing sufficient tests to ensure your code is working **is**.

The tests should be written preferably using the python unittests library but if it proves to be insufficient, another suitable library is always welcome.

1.3 Docstrings

All code must have well documented docstrings in order for Sphinx to produce a **lovely** .rst file. That being said, it is highly recommended to read on how to write docstrings so that Sphinx can parse them.

The bare minimum format for each doc string is similar to the code snippet below:

```
def findItem(self, foodName):
    """
```

```
     Attempts to find the given food name on the menu and
    then returns the food object. If the name cannot be
    found, then raises a NameError exception.

     :param foodName: The name of the food to be searched.
     :return: The Food object being looked up otherwise a
    NameError.
     """
```

Where the key interest is in the `:para foodName:` and `:return:`; any word between the double colons are picked up by Sphinx.

2 Wiki Conventions

2.1 Naming Meeting Notes

The naming convention of the documents should be `<type>_<num>` where `<type>` refers to either: planning, status, review or supplementary. Also, `<num>` refers to the meeting number of that type. Some examples of correct naming are:

- planning_1
- review_1
- status_2
- supplementary_1

2.2 Where To Upload Meeting Notes

The wiki follows a specific folder structure for storing documentations, namely, `aardvark.wiki/assets/meetings`. So for instance, suppose we would like to store the second status meeting notes for sprint 2 on the wiki, then firstly, the naming of the file should be `status_2` which should then be stored in `aardvark.wiki/assets/meetings/sprint2`.

Please do not attempt to attach the document directly through the browser or else the storage location varies significantly and it makes life a bit more stranger...

3 Git Conventions

3.1 Branch Naming

The naming should follow the convention of `<type>-<class>` pattern, where the first term (i.e. `<type>`) is a three letters in length and the second term is unrestricted. So for instance,

- dev-server
- dev-database
- exp-client

Are all valid branch names. (Note that 'dev' refers to development and 'exp' refers to experimental).

3.2 Commit Messages Style

Commit messages must adhere to some rules:

- The first letter of the sentence is must be lower case.
- Full stops must not be the last character of the message.
- Sentences preferably should be kept short (i.e. under 50 characters)
- As a rule of thumb, the amount of work between one commit and another should be roughly 6 hours worth of work.
- The first word of the sentence should be a verb without a suffix. (i.e. an imperative verb)

'Good' examples:

- *update README.md file*
- *fix a bug in server method do_GET that sent the wrong data*
- *ate some mushrooms*

'Bad' examples and reasoning:

- *updated documentation* (Too vague)
- *Fixed bug in server method do_GET.* (Capitalization, non imperative verb and a full-stop)

- *Ate some toast, then some cabbage followed by some cucumber. And oh yes, some other things including glass and a tiny a bit (just a tiny bit) of lead...* (Too long; if it is reasonably long, then split the content into a title and body).

For a more comprehensive guide, the above was inspired by:

Chris Beams–Committing With Style

3.3 Tagging Style

Tags should be of form `<type>-<version>` with the exception of tags denoting sprint runs. Also, the only commits that should be tagged are those that are fully functional, that is, contain fully working code. So for instance, valid tags are:

- server-v1
- client-v2
- sprint3