CS486C – Senior Capstone Design in Computer Science

Project Description

Project Title: CrossDoc	
Sponsor Information:	James Palmer, Associate Professor John Georgas, Associate Professor Nakai McAddis, Lecturer NAU / SICCS Contact: James.Palmer@nau.edu

Project Overview:

Computer code comments are an important source of information for developers. Comments may describe API interfaces, how programs work, features that need to be implemented, and in multicultural teams comments may be needed in multiple languages. In some companies the professional writers write API documentation. Current tools and practice do not strongly support comments created by non-developers, multi-lingual comments, or comments aligned to different software concerns.

In previous work, Palmer and McAddis developed a system called CrossDoc to support source comments in a more flexible way than traditional embedded comments. Using the system, source code is annotated with stable comment anchors to which comments are later attached. These comments are stored externally to the code in locations called comment stores which can be files on the system or other applications like wikis. Comments are then dynamically fetched from the comment stores and injected into the source beneath their associated anchors when accessed through a text editor which is CrossDoc savvy[1].

Comments may also be edited directly through a comment store if that comment store provides an interface for this. For instance, a wiki which functions as a comment store would allow comments to be edited through the usual mechanism for editing wiki pages. Of course, simple text files functioning as comment stores may be edited directly.

McAddis developed a prototype of the CrossDoc system which uses the Atom text editor and a wiki for the comment store. In this project, you will develop a second generation CrossDoc tool based on what we learned from the prototype. While you may use parts of the existing code base, that is not a requirement. Specific features you should implement include:

- Beginning and end tags that denote comment scope – the current system is unscoped
- Tools should provide a command line interface for inserting, changing, and removing CrossDoc comments – the current system must be used from Atom
- Integration support for three of Emacs, Vim, Atom, or Sublime – the current system's support for Atom should be reimplemented
- Integration support for git using git hooks to ensure that only comment anchors/tags are committed (and not the comments)
- Integration support for JavaDoc and Doxygen

[1] A screenshot of the prototype system on the last page demonstrates some of these capabilities

- Support for hierarchical comment stores and/or comment mixins – the current system supports only selecting a single comment store at a time.
- Both application based comment stores and file based comment stores – the current system supports only a wiki based store
- Excellent documentation, test suites, examples, and error checking
- Excellent packaging making the end package easy to install (e.g., brew, pip)
- Formative testing with, at least, CS seniors or maybe even experienced developers from ITS or USGS
- Stretch goal: mechanism to flag documentation as being potentially out of date when code changes or if API doesn't match documentation (using JavaDoc or Doxygen).

Knowledge, skills, and expertise required for this project:

- Experience with git
- Experience with C, Python, and/or JavaScript
- Comfortable using the command line (on Windows, OS X, or Linux)
- Some team members with Emacs, Vim, Sublime, and/or Atom experience is a plus

Equipment Requirements:

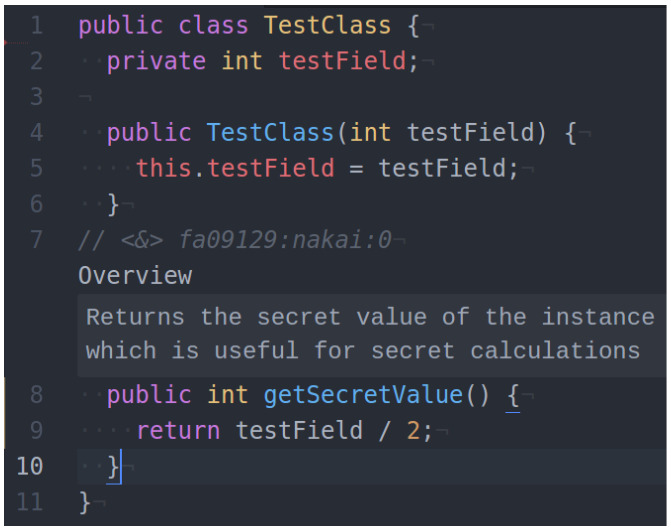
There should be no equipment or software required other than a development platform and software/tools freely available online.

Software and other Deliverables:

- Crossdoc command line tool with excellent packaging
- Excellent documentation (including github style MD, man pages, and PDF)
- A strong as-built report detailing the design and implementation of the product in a complete, clear and professional manner. This document should provide a strong basis for future development of the product.
- Complete professionally-documented codebase, delivered both as a repository in GitHub, BitBucket, or some other version control repository; and as a physical archive on a USB drive.

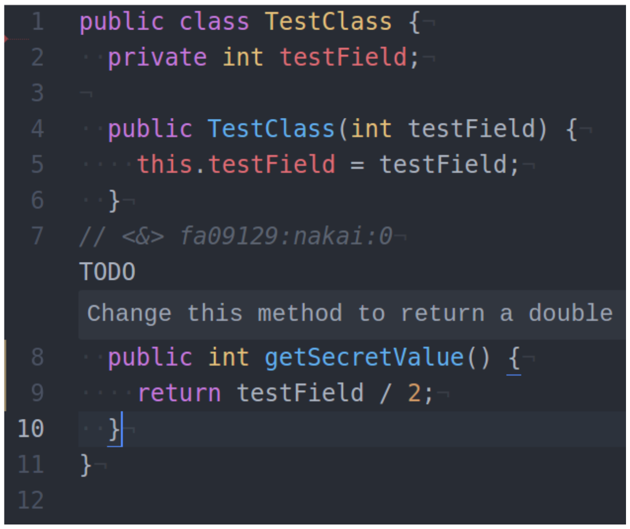
Example Screenshot:

This image shows a sample Java file with a single comment anchor (the line comment beginning with <&>). This anchor has two comments attached to it, an 'overview' comment shown in the left frame and a 'todo' comment in the right. A user can switch which comment is shown for an anchor with a keyboard shortcut.



```
public class TestClass {
  private int testField;

  public TestClass(int testField) {
    this.testField = testField;
  }
// <&> fa09129:nakai:0
Overview
Returns the secret value of the instance
which is useful for secret calculations
  public int getSecretValue() {
    return testField / 2;
  }
}
```



```
public class TestClass {
  private int testField;

  public TestClass(int testField) {
    this.testField = testField;
  }
// <&> fa09129:nakai:0
TODO
Change this method to return a double
  public int getSecretValue() {
    return testField / 2;
  }
}
```