COSC 603

Software Maintenance & Testing

Spring 2016

**Project #2 – Refactoring and Design Smells**

Name: Marlene Encinas

**Task 15 – Summing it All Up.** Upon completion, each student is to submit a short report that includes your write-ups from the previous tasks (clearly labeled) and the following:

* A description (2-3 paragraphs) of what you learned from this project and how it relates to some of the topics covered in lecture

Learning how to use different tools for refactoring like Junit and JDeodorant is important for the organization I work that it main focus is on sustainment phase. In addition, I could recall the knowledge learned in project 1 about using other tools like GitHub, Git for Configuration Management and Javadoc for documentation.

I learned refactoring is an important task that cannot be avoided. It will make source code easier to maintain and will strengthen its design and implementation. Exists a number of approaches that detect smells in source code to alert developers of their presence. There are several approaches that identify smells based on code analysis techniques; I could address during this project how to solve Type checking, Long Method and Feature Envy smell problems.

* A description (2-3 paragraphs) of what you liked about Eclipse’s support for refactoring including its strengths and limitations as well as your impression of JDeodorant

I really like how Eclipse automatically updates of all references to methods, variable or classes when they are changed or adjusted for refactoring reason; these could have not been done fester and efficiently with a simple find a replace tool.

Limitations I found is that Eclipse does not go through the real semantic or intention of the code so propose refactoring changes can cause more problems than solutions if not observed carefully prior applying them.

* A description (1-2 paragraphs) of how/why unit tests are important when doing refactoring

Unit test will ensure that the changes implemented are accurate, do not break the code or cause changes in code functionality. Unit test helps to write down use cases; it could assist to replicate a break and save time when trying to recreate an error. Unit test could provide a complete list of the problems the code has to solve.