Owens Portfolio

**Decorator, Strategy, Template, Observer, Singleton**

In depth knowledge of design patterns, critique different solutions to a design problem

Design principles, code refactoring, code smells using appropriate terminology and tools

*Marks:*

* Novelty/Originality of work
* Explore how pattern can be tailored to suit a specific situation (push or pull, or an abstract class vs an interface) **20%**
* The fit of the design pattern to the design scenario, motivation of the problem pros and cons etc. **10%**
* The use of the design principle, code smells, refactoring in the documentation and discussion. **40%**
* Explanation of the working of your design pattern inc. UML diagrams. **30%**

Overall Performance **20%** of total mark:

* The use of tests to ensure safe refactoring, demonstrate through a clear understanding of the refactoring to or from the patterns. **30%**
* Unity of work, a project rather than a mixed bag of design patterns. **50%**
* Git log, consistent work. **20%**

Distinction:  
- Awarded to a learner who has substantially exceeded the minimum requirements.  
-- Excellently motivated and very clearly communicated design scenarios showing an expertise in both the language and terminology of software design patterns and software design.  
-- A clear ability to critique code and alternative designs, and to implement solutions  
-- A demonstration of reading and skill beyond the course materials.  
-- An excellent ability to identify and justify the need for inclusion or removal of design patterns in code that demonstrates an ability to acquire and apply learning.  
-- The application/patterns at this level tend to be real world applications and typically are interesting/challenging for other reasons. A distinction award is testimony to a mastery of design patterns themselves, including a knowledge of the flexibility of the patterns and an ability to identify and weigh up the pros and cons of design decisions. This includes the use and deep understanding of terminology, principles,  code smells, refactoring, process, tools (UML, Git), etc

Zip of repo , link to repo, this document – No Dracula, 2500-3000 words excluding code

Motivating reason for change, principles, examples, demo, conclusions, implementation

Intents, reason for introducing pattern, argue why use it, is the change a good choice

Add something to it, add flexibility, intent, strategy push pull multithreaded scenario abstract instead of interface?