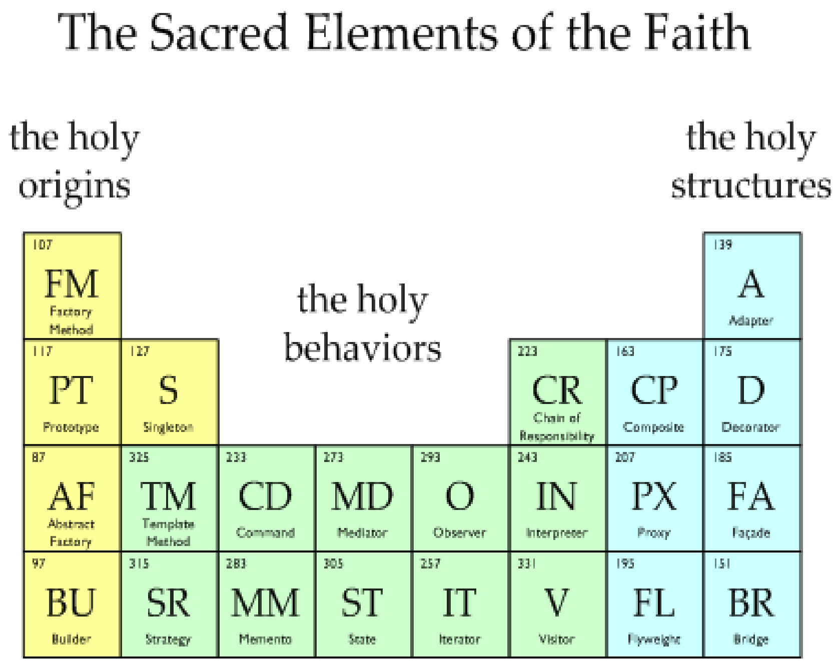
OOSE200 Report
Company Training Simulation

Connor Beardsmore - 15504319



Curtin University
Science and Engineering
Perth, Australia
October 2016

"Company Training Simulation"

Polymorphism

- Property - kept in map, polymorphically call calcProfit() via strategy
- Events + Plan - both use strategy so can call run() on parent class
- WageObserver list allows ANY class to become an observer if it implements

Design Pattern Implemented

- Factory - for Events and Plans
- Dependency Injection - import objects, new in main, no statics
- MVC - overall layout, multiple controllers
- Observer - wage observer for global updates
- Composite - company owns other companies, tree structure for profit
- Template Method - file reading, common code for opening/closing files
- Strategy - plans.run(), events.run() and property.calcProfit()
- Iterator - for loops are sick, even if super inefficient

Testability

- Test cases!! sample outputs to clear up order ambiguity
- Factory + Dependency Injection allow for easy mocking of objects, low coupling
- Mad toStrings() and debug output methods
- clear and consie exception handling
- tested on heaps of invalid file types for all 3 input files

Alternative Design Choices

- design is mad extensible and shit due to the patterns used
- Iterators instead of for loops
- Controllers are easy to switch in and out, could have used one bunta controller
- could have used scanner instead of BufferedReader