COS 121 Project

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- Date Issued: **21 September 2015**
- Final Due Date: **26 October 2015** at **midday**
- Submission Procedure: **Upload via the CS website**
- Submission Format: **zip or tar + gzip archive (tar.gz)**

1 Introduction

This project covers practical sessions 8 to 10 and will count the equivalent of three practicals in the mark weighting for practicals.
During this project you will be required to implement several design patterns you have learnt about this semester in creating a text-based console game, you will be required to document all the classes and create the appropriate UML diagram to show all classes created in this project.

You have been tasked with the creation of a simulated war between the notorious forces of evil and yourself as the controller of the light forces. You will be given a basic map, which you are allowed to edit slightly and may be given a different map on demo day. There is also a bonus task for random map creation

Note: All questions for the project can be posted on the COS121 Discussion board and will be answered weekly in the tutorial sessions, Please be sure to participate in the discussion board.

2 Constraints

1. You may complete this assignment individually or in pairs **NO more than 2 members per group**, bookings will be made available for demonstration of the project, only one member needs to complete the booking procedure and will select their partner.
2. Bookings for teams will be made available on the CS website
3. All tasks will need to be uploaded weekly by **ALL** members of the team, this will be checked before the final demonstration.
4. You may ask the Teaching Assistants for help but they will **NOT** be allowed to give you the solutions.
5. Demonstration bookings will be made available for the demonstration of your project. Please ensure you book for your team.
6. Your practical will not be marked if:
    - You fail to book for a demonstration session.
    - You fail to upload one or more of the three tasks.
7. If you work in a pair, both team members will be required to answer any question about the project.
8. Tasks will be marked up to your last upload. i.e. if you upload task 1 and task 3, only task 1 will be marked due to the failed upload of task 2.

3 Submission Instructions

You will be required to upload tasks every week, for the remainder of the project:

- Task 1 is due 12 October 2015
- Task 2 is due 19 October 2015
- Remaining tasks will be due 26 October 2015.
- Bookings will be made available for demonstrations

4 Mark Allocation

Task	**Marks**
The Unit Heirachy	25
Game Master and Unit movement	25
Game play and fitting it all together	25
UML Diagrams and Doxygen	25
TOTAL	**100**

5 Assignment Instructions

This practical consists of three main tasks a UML task and a Bonus task.
You are required to create documentation for all tasks using Doxygen.
Note: The method of implementing the design patterns discussed per task is to be used as a guideline only, please feel free to implement the patterns in their own way or add additional pattern's, should you feel the execution and code are more elegant.

1. The unit hierarchy
2. Game master and unit movement
3. Game play and fitting it all together
4. A UML diagram showing all classes of this project
5. Bonus Tasks:
    - Map generator
    - Stylized Terminal

Task 1: The Unit Hierarchy ..... (25 marks)

In this task you will create the classes for all your units, by utilizing Prototype and Abstract Factory design patterns.
You will be required to implement the Prototype within the Abstract factory, this will be done by creating a prototype of each unit which will be cloned to complete your army

- An abstract Unit Class.
- Player and Monster classes that inherit from Unit.
- There will be 3 Monster classes and 3 Unit classes with 3 types of units:

Types	Magic	Bludgeoning	Piercing
Player	Mage	Soldier	Thief/Rogue
Mobs	Elemental	Ogre	Goblin

- The type of the unit determines if that unit deals more damage to another unit, similar to Rock-Paper-Scissors, magic deals more damage to bludgeoning, bludgeoning deal more damage to piercing and finally piercing deals more damage to magic.
- The health and damage points of the units are given in the table below

Unit	Health	Damage
Mage	80	5
Soldier	100	8
Thief/Rogue	60	10
Elemental	85	4
Ogre	120	5
Goblin	50	12

Task 2: Game Master and Unit movement ..... (25 marks)

In this task you will need to use the Observer, Adapter, Facade and Mediator design patterns to create and simulate movement among units. You will need the classes to be able to satisfy the Observer, Adapter, Facade and Mediator design patterns.
By making use of the Observer and Mediator patterns, you will be able to simulate game play amongst yourself and the computer. The Observer should simply observe the game play and alert each the player when it is their turn and allow the computer to be informed when it is it's turn. The Mediator pattern is implemented by using the observer pattern in such a way that the classes should not communicate directly with one another but rather communicate through the mediator and observer respectively. The Adapter and Facade pattern should

be implemented in such as way to allow for adaptation between the classes and the Observer and Mediator patterns. The Facade is used to give the impression that the classes offer a simple method of taking damage, thereby hiding the Rock, Paper, Scissors complexity (i.e. you will have one function to allow for damage to be taken by all units).

You will need to allow for the map to be updated after each turn as the units move through the map.

- The Observer and Mediator patterns will be used to organise the game play into a turn based playing system.
- The Adapter and Facade patterns will be used as a layer of abstraction to allow for a simplified interface to be used by the Observer and mediator classes.

Task 3: Game play and fitting it all together ..... (25 marks)

In this task you will modify your unit classes to use the Bridge design pattern. As well as implement the Chain of Responsibility design patterns. Chain of Responsibility should be used in such a way that if you have a team of 3 units, with Health 2, 4 and 10 and a unit deals a total damage of 8 attack points it should deal 2 damage to the first unit, leaving a attack point value of 6, it shall then deal 4 points to unit number 2 and finally deal 2 points of damage to unit 10. Thus the damage must be chained together to allow for the amount of damage to be distributed amongst the creatures. The Bridge pattern should be used in such a way to provide a deeper level of abstraction to the units and subclasses, so as to allow more units to be added easily and created by the Abstract factory and Prototype.

- The Chain of responsibility shall be used to deal damage to each unit in turn if one unit cannot take all of the damage.
- The Bridge pattern shall be used here to change the initial layout of task 1, as a overall more efficient structure.

Task 4: UML Diagrams and Doxygen ..... (25 marks)

You will be required to create the following UML diagrams for all classes of your project.
Note: Please upload all diagrams and documents in PDF format, if you are unable to create a PDF for Doxygen you may also upload the HTML files.

- UML Class Diagram - With annotated participants, i.e. tell us which is the Mediator Design pattern and then state what each participant is for each.
- Sequence Diagram - Please complete a sequence diagram on your project consisting of a sequence which originates from the main class and goes through at least 4 classes with different function calls.
- Activity Diagram - Show a basic activity diagram of how the attack function attacks a player and deals damage to the opponent. You must also show whether the unit dies or not.
- You will be required to show all your Doxygen documentation for all classes and functions.

You will be expected to be able to identify the design patterns within the UML class diagram and show members of the patterns.

Software: You can use Visual Paradigm to create your class diagrams. The license details are available on the COS121 discussion board, but you may also use the Community Edition.

Task 5: Bonus Tasks ..... (25 marks)

You will only receive Bonus marks if all other tasks have been correctly implemented.
Note: The more impressive the functionality of the bonus task(s) implemented the more marks you will receive

- Create a class to randomly generate the maps for your game.
- Use Colours and animation in the Terminal to make your game more appealing.
- A game GUI can be created, so as to not only having the game played via the terminal.
- You may add any additional functionality and marks will be awarded according to the difficulty of the added functionality.