**Architecture**

ENG1 Team 16, ‘Team Team’

Luke Batten

Owen Crucifix

Samuel Humphreys

Robbie Parr

Jude Daniels-Smith

Alan Yang

**Architecture**

This document aims to show how we plan to connect our requirements to the implementation of the software.

Abstract Architecture

An abstraction of the architectural style that we plan to use. Both diagrams were created using draw.io.

Diagram, schematic

Description automatically generatedDiagram

Description automatically generated

Concrete Architecture

Below is our concrete diagram which represents the implementation of the code. The diagrams will closely represent the code and design of the game implemented using libGDX. This was created using plantuml. For readability purposes, we cut the diagram into two

Graphical user interface, diagram

Description automatically generated with medium confidence

Diagram

Description automatically generated

For readability as the diagram is across to pictures. Below state where lines go from the second picture to the first

* Playercollege class goes to Player and College class
* Enemycollege class goes to the college class.
* The line for Enemy class goes to Player class
* The line from Main class goes to Player class

**Architecture Justification**

We have used an inheritance approach within our concrete design firstly because we all have experience in designing software that way and also as relatively there are not that many entities an inheritance approach is appropriate.

Requirements to Abstract

* Main: UR\_Game, FR\_CollegeSelection, The main state is used to start the game the user can select which college they intend to be and then the game can begin.
* Win game: UR\_Win, This state is for when the user wins the game, It has to change states for the game to be over so the game will end.
* Lose game: UR\_Loss, When the user reacts a stage where the game has been lost then it has to exit the game so it finishes and the user can’t keep playing
* Lake: FR\_Map, the lake represents the game map and everything apart of that the objects on the lake
* Player: UR\_Game, this represents the user entity and the objects that the user is in control of.
* Ship: UR\_Ships, This is an object which will be on the map the user will control one and enemies will have their own
* College: UR\_Colleges, This is an object which will be on the map the user will control one and enemies each have their own.
* Task: FR\_Tasks, FR\_Objective, This is a state where the user must complete tasks the be able to complete the final objective and then finally win the game
* Enemy: FR\_NPC, This is an enemy entity which links to its own college and ships to interact with the user.

Abstract to Concrete

User: FR\_Movement,NFT\_inputLatency

FR\_CollegeSelection, FR\_Capture, FR\_GainPoints, FR\_GainPlunder

* Player is the Concrete version with all methods for the player to interact. It inherits from the entity superclass which allows players to move and act as an object in the program.

Enemies, FR\_NPC

* Enemy is the concrete version with all methods for the Enemy ships and colleges. It is where the AI can be implemented and where the drawing of the Enemies is located. Allowing its inclusion/exclusion simple. This also enables it to be temporarily removed, which would help in testing other aspects of the project.

Ships, FR\_Destroyed, FR\_Health

* Ship is a concrete version of all methods for the ships to use. PlayerShip, EnemyShip & FriendShip (cut off diagram) all inherit from Ship and it enables them to have positions, basic movement, a traveling range(moveRange), attack range (range) and health. Which (when combat is implemented) can be reduced to 0.

Mapping, FR\_Graphics, NFT\_Screen

UI. Talk about what information the user is able to see on screen

* Although a work in progress at the time of writing this, the Tasks object, Map object & TileType object share the concrete version of all the methods related to these requirements. Map enables the user to see the map and processes most transactions related to it. TileType scales the map to fit the screen and enables each tile to be selected for movement, showing a red outline if you cannot move to it. And Tasks (in progress) will display the health and current task in the top left hand corner.