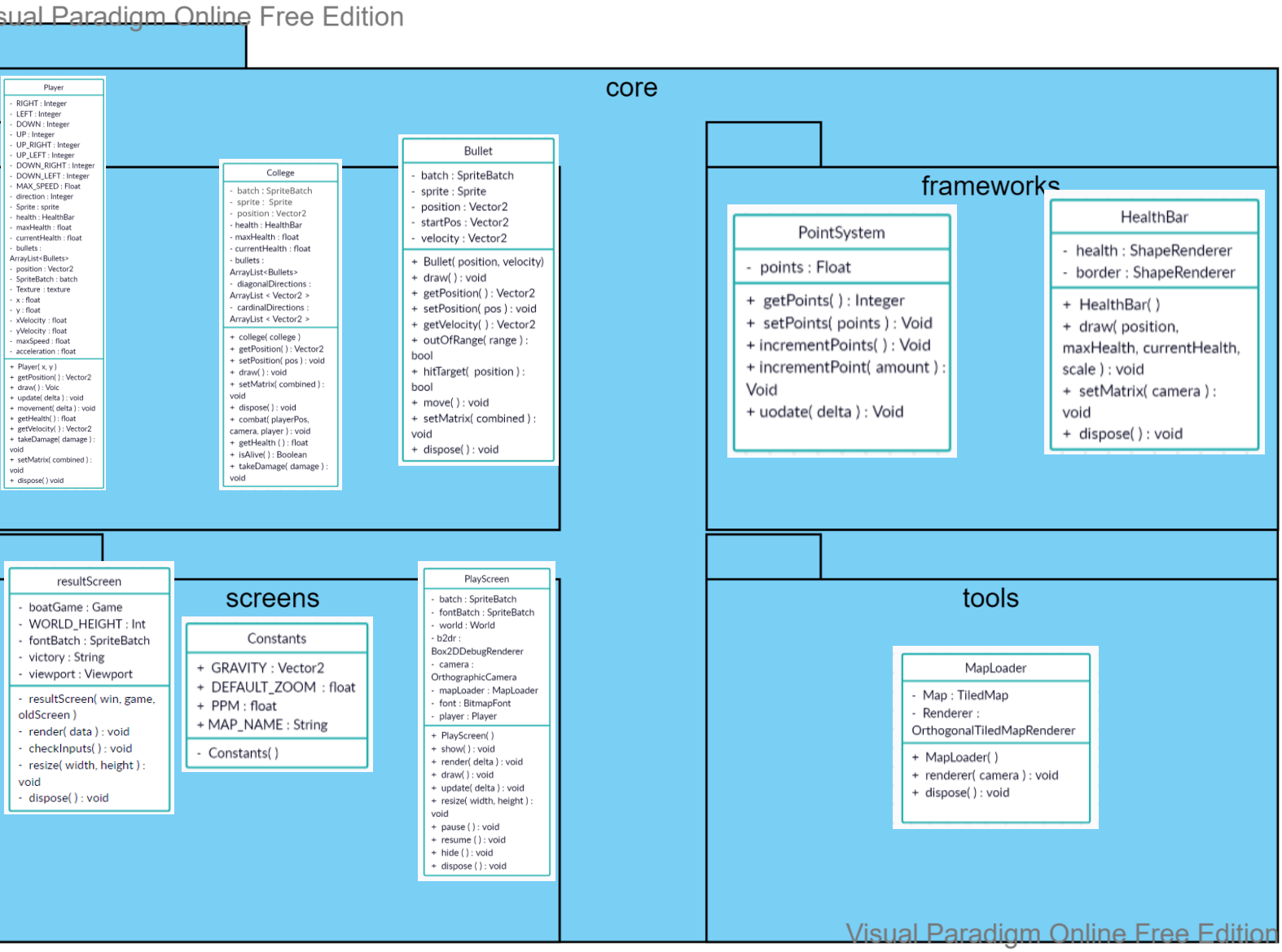
Architecture

To build the game we decided to use the java programming language since the game engine we were using is LibGDX (which is written in java). We went with this as most of us are more familiar with the language, mainly because of previous modules, so all we really had to learn was the framework of the game engine which is pretty well documented online. There are also plenty of video tutorials online to help as well. Furthermore, the LibGDX framework is great for smaller sized games like the one we are developing.

UML Packages

The architecture is split into four packages all contained within the core package, these being the: entities, frameworks, screens and tools.



Entities Package

The entities package contains the college, player and bullet class. The Player class is responsible for constructing the boat that the user will control and contains the methods which allow them to move around, interact with other sprites and all in all, play the game. The College class is responsible for constructing the AI that the user will come up against when playing the game, which are of course the colleges of the University of York. These two classes also contain attributes and classes relating to their health in the game which indicate their state in the game (dead or alive, weak or strong). Finally the bullet class is responsible for constructing the bullets that the user will use in combat with other colleges.

This package is important in fulfilling the following user requirements of our game: USR2 USR7 USR9 USR17 USR19. These requirements are mainly related to the combat between the player and the colleges which happens through each side using bullets to deplete each other's health bar till they are empty and completing the goal of the game. This is in exception of USR2 and USR17 which are about the game being single player and colleges being controlled by an automated AI which are also carried out by this package as the user only controls the entity constructed by the Player class.

The functional requirements: FR02 FR04 FR05 FR06 FR07, are completed from this package too. These requirements are about the user controlling their sprite to engage in combat with the other colleges and in turn the colleges being able to fight back at the player. FR03 varies slightly from the rest but I still feel it is related as it is the requirement of there being a victory and failure page but that wouldn't occur unless the combat between the player and colleges determined a winner and loser.

Requirement NF03, which is to do with no gorey images, is also fulfilled here as the bullets and combat between player and college isn't horribly violent.

Frameworks Package

The Frameworks package contains the PointSystem class which, as the name suggests, tracks the user's points as they play the game. This is a main framework of the game as it is an indicator of how close the game is to finishing. It also contains the HealthBar class which is the main framework that tracks the health of the colleges and the player and helps indicate their state in the game (dead or alive, weak or strong).

This package helps fulfill user requirements 4 and 7 which is mainly related to how the game finishes and also to do with the health of the player and colleges. USR4 is about the game lasting from 5-10 mins which gets fulfilled by this package mainly from the point system as it would finish the game after a threshold has been reached and USR7 is about colleges being captured or destroyed which is carried out by the HealthBar class

The Functional requirements carried out by this package are: FR03 FR06 FR10 FR12. These are mainly talking about gaining points and winning/finishing the game apart from FR06 which is the requirement of the user taking damage if they are attacked successfully by an opposing college.

Screens Package

The Screens package contains three classes which are the constants, the PlayScreen and the resultScreen. Constants just contains, as the name suggests again, the constants for the game such as the values for gravity, the name of the map, the zoom value etc. Then there is the PlayScreen class which constructs the screen that the user will see in order to play the game. This includes methods for rendering items, resizing them, updating them etc. The results screen constructs what the user would see appear on the screen once the game is over showing how well they did.

This package fulfills the user requirements: USR11 USR12 USR14 USR16. These requirements are the ones that state how the screen that the user plays on looks like such as USR11 which talks about how the game should have varying screen sizes or USR16 which requires the game to be clean and non violent.

The functional requirements that are satisfied by this package are: FR01 FR03 FR09, which are to do with fitting the game into any sized screen and the different screens displayed in the game. These being a start screen, an end screen, a pause screen etc.

Tools Package

We then lastly have the tools package which contains the MapLoader which is the class that renders the map that the user will play on

This package completes the user requirement for making it a 2D game MapLoader renders a 2D map and functional requirement of it being able to fit on any sized screen as the map can be resized.

It may be evident that we have not listed all of our requirements here as this is because not all the requirements can be necessarily met by the code and the architecture directly. An example of this is user requirement 3: Intended for visiting potential students and families or non functional requirement 5: should be intuitive. This is because these are more to do with the overall design aspect of the game rather than the concrete architecture.