Architecture

Cohort 1, Team 1:

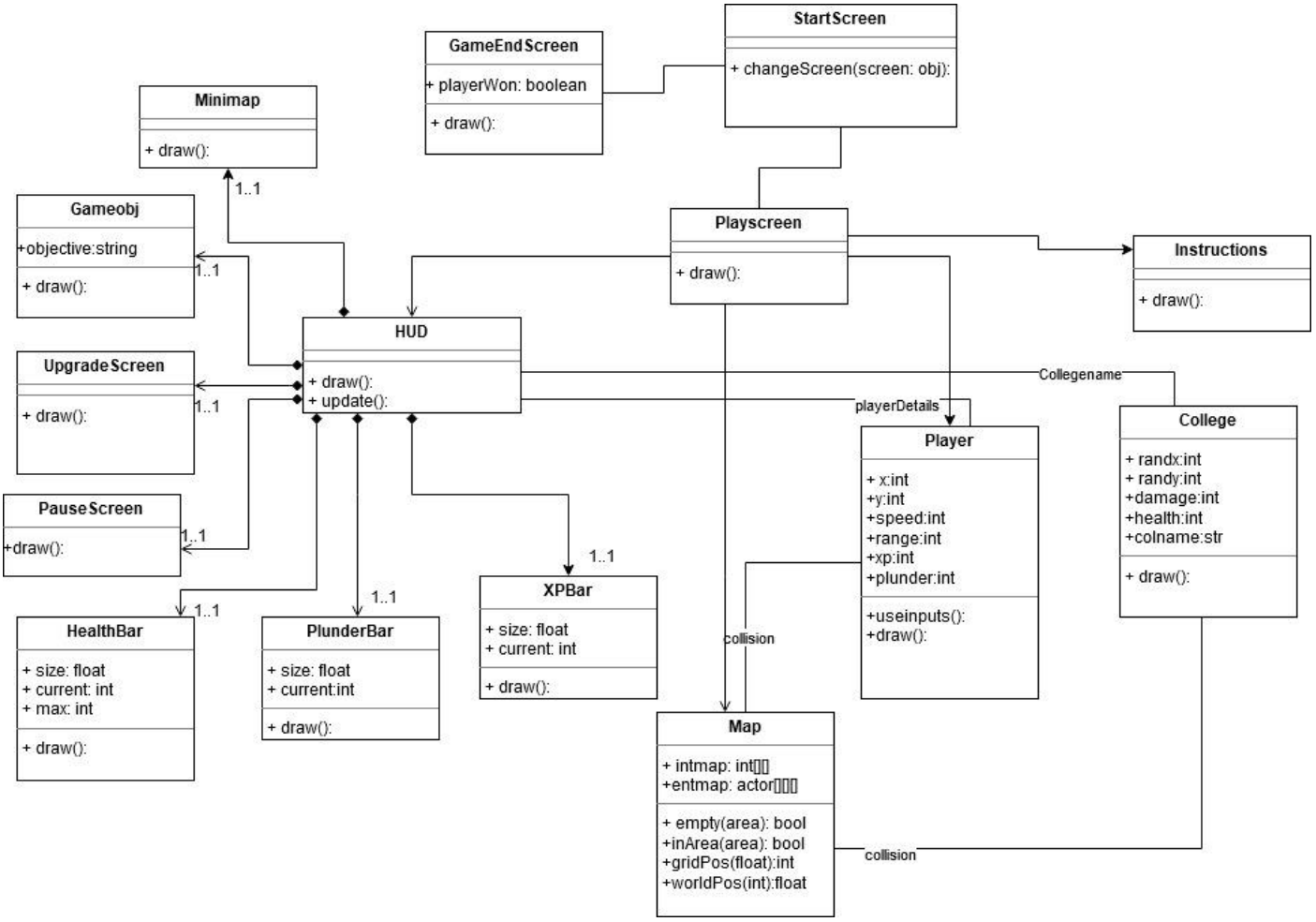
Connor Blenkinsop
David Luncan
Emily Webb
Muhidin Muhidin(MO)
Sam Laljee
Sooyeon Lee

The abstract and the concrete architecture was created via class UML using draw.io, a diagram maker.

To organize a plan for our game we decided to create a Class for the game which consisted of a HUD, a map, and multiple entities on top of the map, collisions between entities would communicate with the map class.

Abstract architecture
A copy of the Abstract.drawio can be found at:
https://drive.google.com/file/d/1rIGCJSOYcKlvkXnU_mSbHuFmSjPhRjWf/view?usp=sharing

The abstract architecture displays the structure of the code, which is the relations between all the classes but does not include all of the methods and attributes as we didn't know what framework we would use at the time.

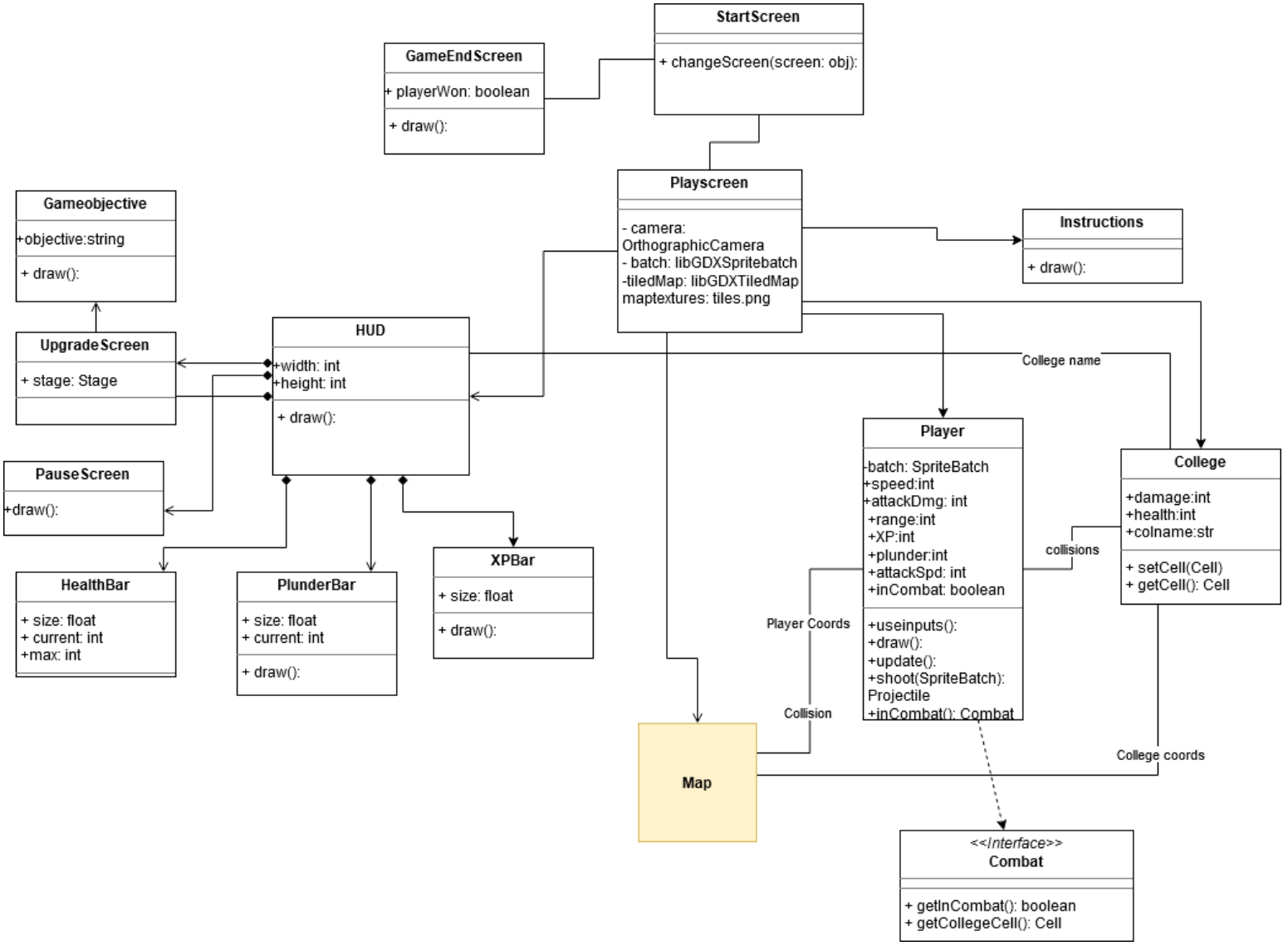


Concrete architecture

The concrete architecture is implemented using the libGDX framework meaning that libGDX types are implemented as basic types like attributes. This means that the player class and projectile class is inherited from the sprite class that is imported from libGDXhowever the contents are not referenced, The same goes for tiledMap which is called in multiple classes.

The concrete architecture has been separated into another diagram to reduce complications as the map class was made more complicated in the libGDX framework, requiring 3 more classes.

A copy of the concrete architecture can be found at:
https://drive.google.com/file/d/1HKyYzGQ8SdtjMWq6n-K2Wos24pIYHrUL/view?usp=sharing

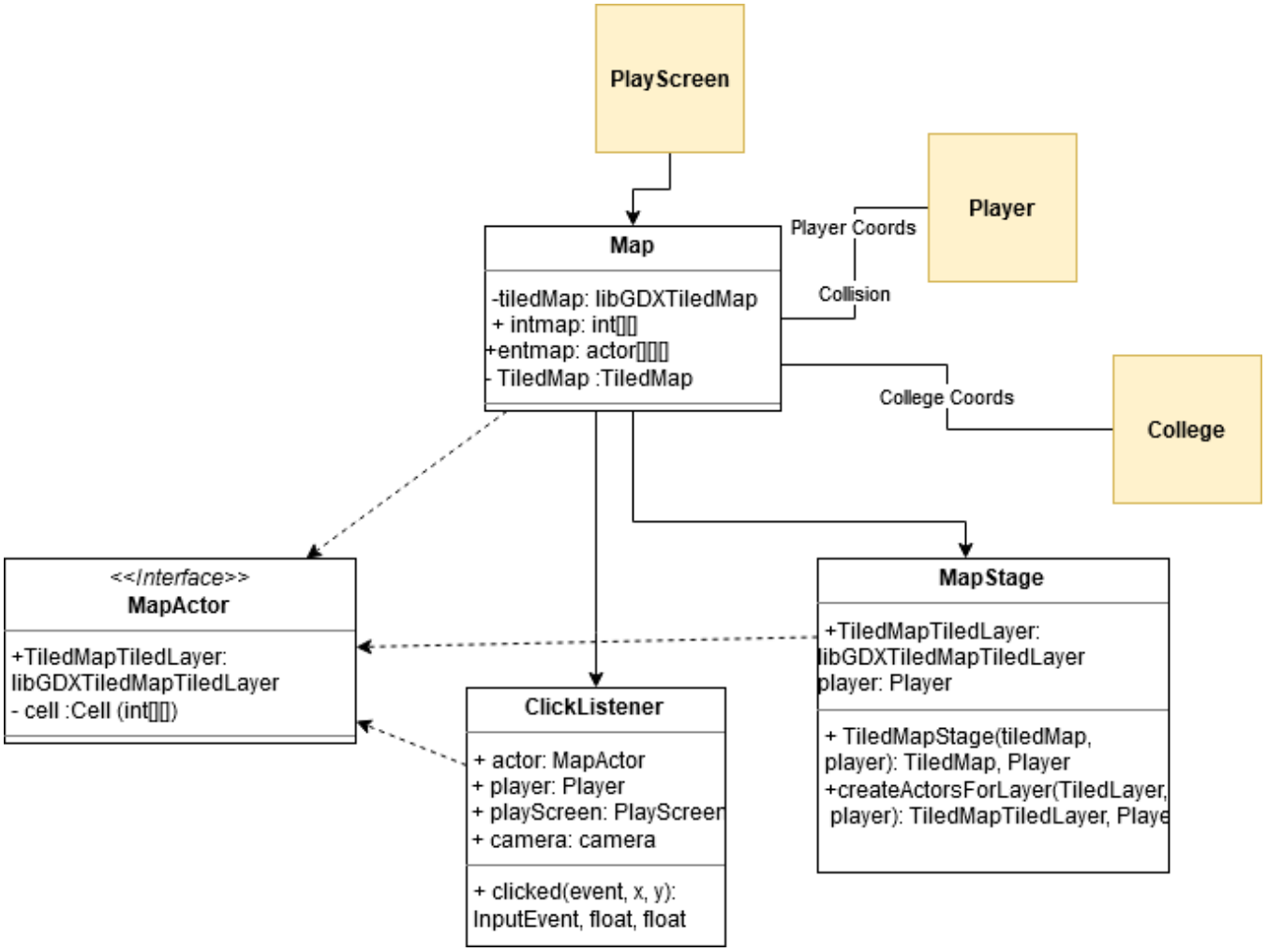


Map

Part of the concrete architecture, added an extra 2 classes to allow the creation of the actor layer and allow the map check for clicks on the screen.

Copy of the diagram can be found at:

https://drive.google.com/file/d/1jZBnRzA7CmC2ClCo4gzoJ3J97gVg55RB/view?usp=sharing



Systematic justification for the abstract and concrete architecture

Why we used OOP

In our process of collecting information we decided to use Object-Oriented programming (OOP) due to a couple of reasons:

- The outline of the game and the information we gathered from our customer meeting pointed towards a hierarchical structure that we implemented with the HUD being the parent class to many classes as well as allowing polymorphism with the upgrade screen and the pause screen. This also will be used for the different types of tiles as many of them would be similar.
- Our team all have experience with OOP and java, meaning that it was easier for our whole group to understand it.

Design of abstract architecture

The abstract architecture allowed us to represent the relationships in its basic form, allowing it to be understood without the framework that we used to influence the relations. It also represented our initial thoughts on how we would have created the game without any problems and setbacks.

How the concrete architecture builds on the abstract architecture

The concrete architecture adds the methods and attributes that were necessary when using the libGDX framework as well as editing the abstract architecture in ways that were necessary for us to use in the code. The aspects that were changed in the concrete architecture are:

- We removed the minimap feature as it felt unnecessary when in the game as it was easier to see where all the colleges were than previously thought when testing the framework and took up a lot of space on the screen.
- Instead of the college being a separate moveable entity we place the colleges in set places on the tilemap and allowing the range for combat to be calculated easier
- Instead of adding random locations for colleges we are using a few different maps that will be randomly chosen at the start of each game.
- Added a combat interface to check whether the player is in combat with a college
- Added new classes to the map for click listener and actor layer to check for when the game screen has been pressed and actor layer to map the entities onto the screen at the right coordinates

Class	Description	requirement
GameObjective	Displays what the user has to do to complete the game	UR_GOAL
HUD	Parent class of the upgradeScreen and HealthBar, XPbar and PlunderBar displaying it on the game screen	UR_STATS

GameEndScreen	Reached when objective college is defeated (captured or destroyed) or the player has run out of Health	FR_END_SCREEN
GameEndScreen	Contains a boolean that checks whether the user has run out of health, if yes defeat on the GameEndScreen is shown	FR_USER_DEFEAT
XPBar	Displays the amount of XP the user has at that moment	FR_XP
PlunderBar	Displays the amount of Plunder the user has at that moment	FR_PLUNDER
StartScreen	Contains a menu and is the page when you can start a new game	FR_START_SCREEN
Map	The map contain at least 3 colleges on each different one	FR_PLOT_COLLEGES_COUNT
Player	The user moves the player using the mouse and the keyboard	NFR_INPUT
Instructions	The instructions pop up when the game begins	NFR_USABILITY FR_INSTRUCTIONS
ClickListener	Always checking to see whether the TiledMap has been clicked so allows for fast updating and rendering	NFR_RESPONSIVENESS
College	Colleges increase the damage and health they have once a college has been defeated	FR_SCALING
PauseScreen	The pause screen allows you to resume the game or toggle the sound.	FR_MUTABILITY