ENG1 Team Project

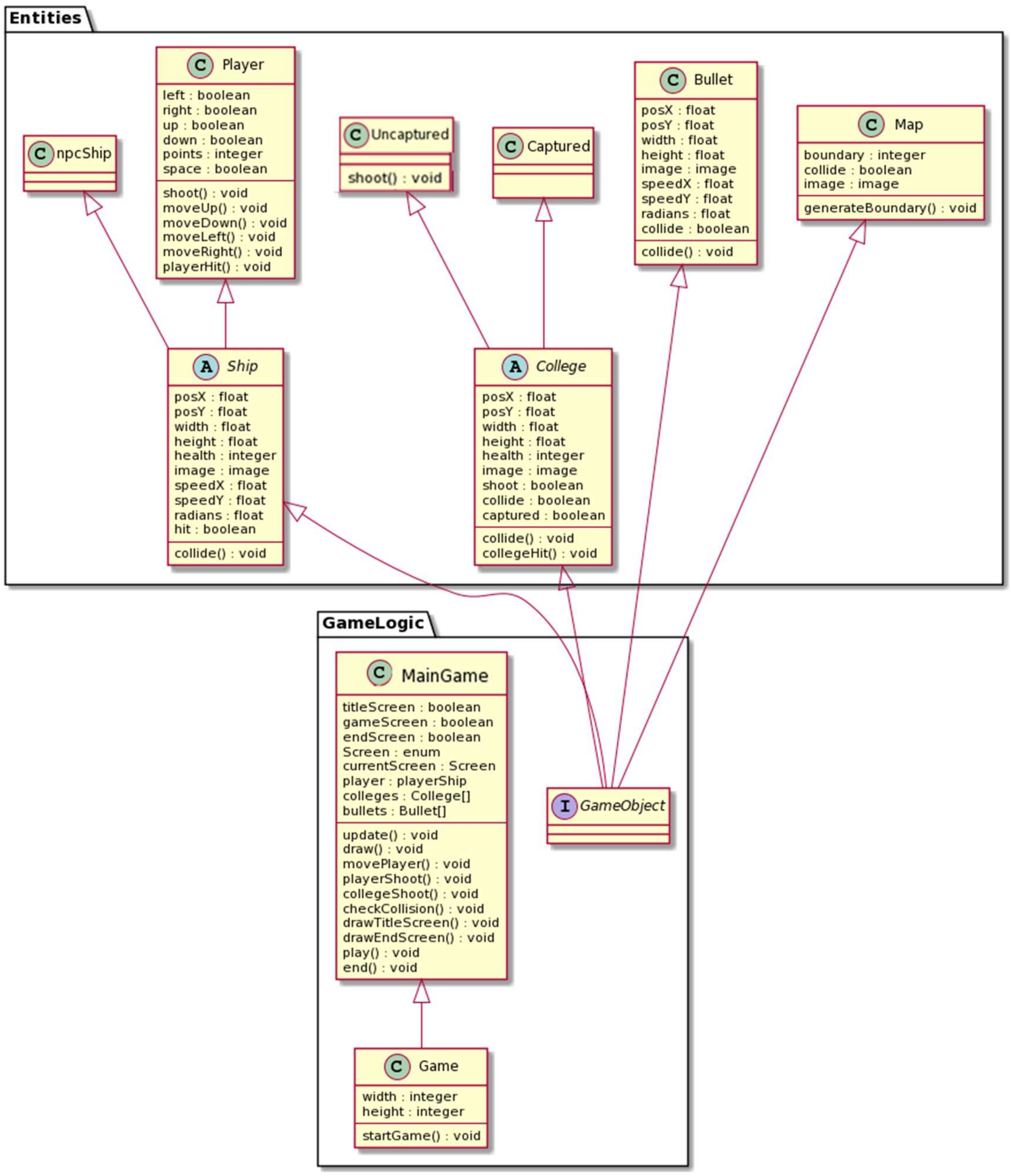
Deliverable 3: Architecture

Team 13 – Team Unlucky

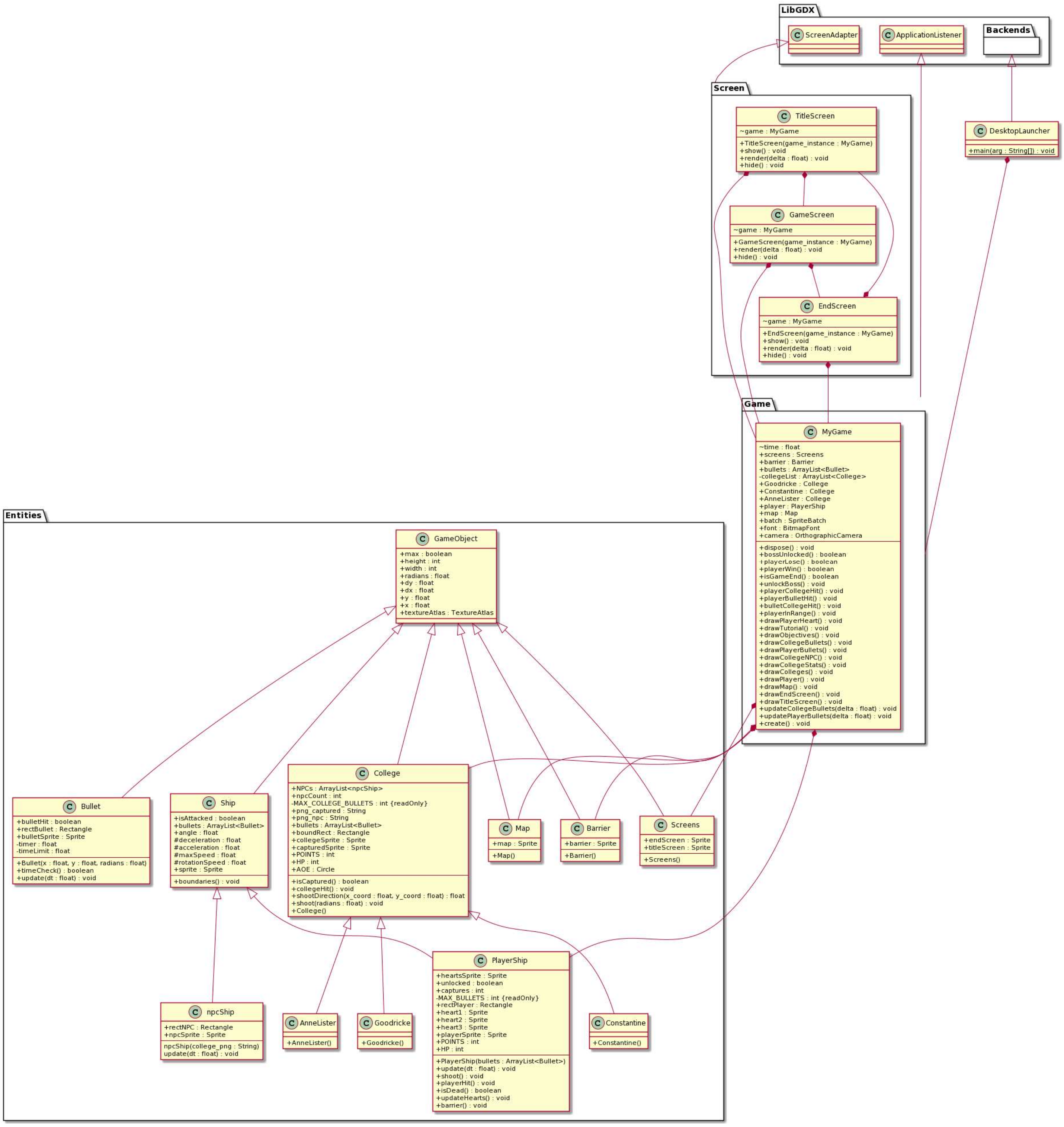
LILLY BROWN
ROSCOE GILLATT
ADAM JOHNSON
MATTHEW TAYLOR
BRANDON WEST
AYMAN ZAHIR

a)
The tool we used to produce the abstract and concrete architecture was PlantUML, this allows a user to create diagrams and flowcharts from a plain text language, we chose to produce a class diagram. We also used an IntelliJ IDEA UML Plugin that auto generates UML class diagrams from existing code but those were so noisy that we only used them as a reference to create the class diagrams with PlantUML.

Abstract Architecture:



Concrete Architecture:



b)
We used our abstract architecture as a preliminary guide for implementing the program in code. Since we had not selected the infrastructure beforehand we kept the abstract architecture prescriptive, based on the basic requirements and loosely planned approaches to implementing them. We decided it’s best not to complexify it before selecting an infrastructure. The design includes a simple OOP Hierarchy for the game Entities and the initial plan for the game and logic surrounding the game Entities in a fairly closed architecture system.

As this was only used as a guide the concrete architecture changed considerably, especially with the LibGDX infrastructure. It expands on the abstract architecture to become a more semi-open architecture with higher cohesion between packages. One key change is the implementation of the MyGame class which acts as a facade to delegate all the other entities behind the scenes and provides a simplified interface for the game logic and shared resources. Another key change is the screen package which now. Instead of switching between screens inefficiently with conditional statements we now have three separate screens for each state of the game: each screen has its own class with screen specific logic and passes the shared resources between them.

The concrete architecture is more complex, however, it still recontextualizes most of the logic of the abstract architecture while adding or modifying certain aspects. Having the abstract architecture prescriptively structured allowed us to interpret how to implement most of the program using the LibGDX infrastructure while easily adding new components to the packages such as Entities to expand the functionality of the program.

Concrete architecture justification

Architecture construct	Requirement ID	How it is fulfilled
PlayerShip, College	UR_GAMEPLAY_ MODE	MyGame initializes one PlayerShip and Colleges from a list of colleges. The update method in PlayerShip handles player movement and shooting.
MyGame, DesktopLauncher	UR_DESKTOP	The main function in DesktopLauncher initializes a new desktop application with a new instance of MyGame upon launching the game
MyGame, BitMapFont	UR_USAGE_MOD E UR_EASE_GAME PLAY. FR_TUTORIAL, NFR_USABILITY	MyGame’s drawTutorial method triggers when new game is created and overlays timed BitMapFont based instructions
GameScreen, Orthographic Camera	UR_DISPLAY NFR_SCREEN_C OMPATIBILITY	GameScreen has code in render function which tracks the screen dimensions with the camera dimensions to preserve aspect ratio
PlayerShip	UR_INTERACTION	PlayerShip is controlled through keyboard input in the update function using Gdx.Input.Iskeypressed()

Map, MyGame	UR_MAP	Map initializes a new sprite that is designed to look like a lake and the sprite is then loaded by GameScreen's render function
PlayerShip	FR_ONE_SHIP	The player uses the keyboard to control only the player ship
PlayerShip, Bullet	FR_FIRE	PlayerShip's shoot() initialises an instance of the class Bullet and adds the bullet to the list of bullets in MyGame.
College, Circle, MyGame, PlayerShip	FR_BUILDINGS	When a College is initialized it has a Circle with a set radius. The method playerInRange under MyGame checks if PlayerShip overlaps with the Circle in which case college shoots at the PlayerShip
TitleScreen, Endscreen	FR_MENU	TitleScreen() and EndScreen() initialise short clear menu screens to start and reset the game
MyGame, Player, College	FR_ANIMATION	The method drawPlayerHearts draws the varying heart sprites of the player based on the player's current HP. The method drawColleges draws a collegeSprite or capturedSprite based on whether the college is captured or not.
Ship, PlayerShip, Barrier	FR_BOUNDARIES	Ship's boundary method and PlayerShip's barrier method define the bounds of the map when the boss is unlocked and locked respectively to restrict player movement.
College, PlayerShip, MyGame	FR_POINTS	PlayerShip increments POINTS by the College POINTS once a it's captured in the bulletCollegeHit() function under MyGame
GameScreen	FR_ONE_SCREEN	All the GameObjects initialized in MyGame is rendered only in the GameScreen
MyGame, PlayerShip, Barrier	FR_TASKS	MyGame's drawObjectives() function displays the task of capturing colleges, displays how many colleges are captured based on PlayerShips captures, and only displays the final objective when the task of capturing all other colleges is complete. The boss college is unreachable initially due to the barrier restriction on the player
GameObject, College, PlayerShip, npcShip	FR_COLORBLIND	All GameObject sprites images are custom made for each College, PlayerShip and npcShip with different images so that they are identifiable without color