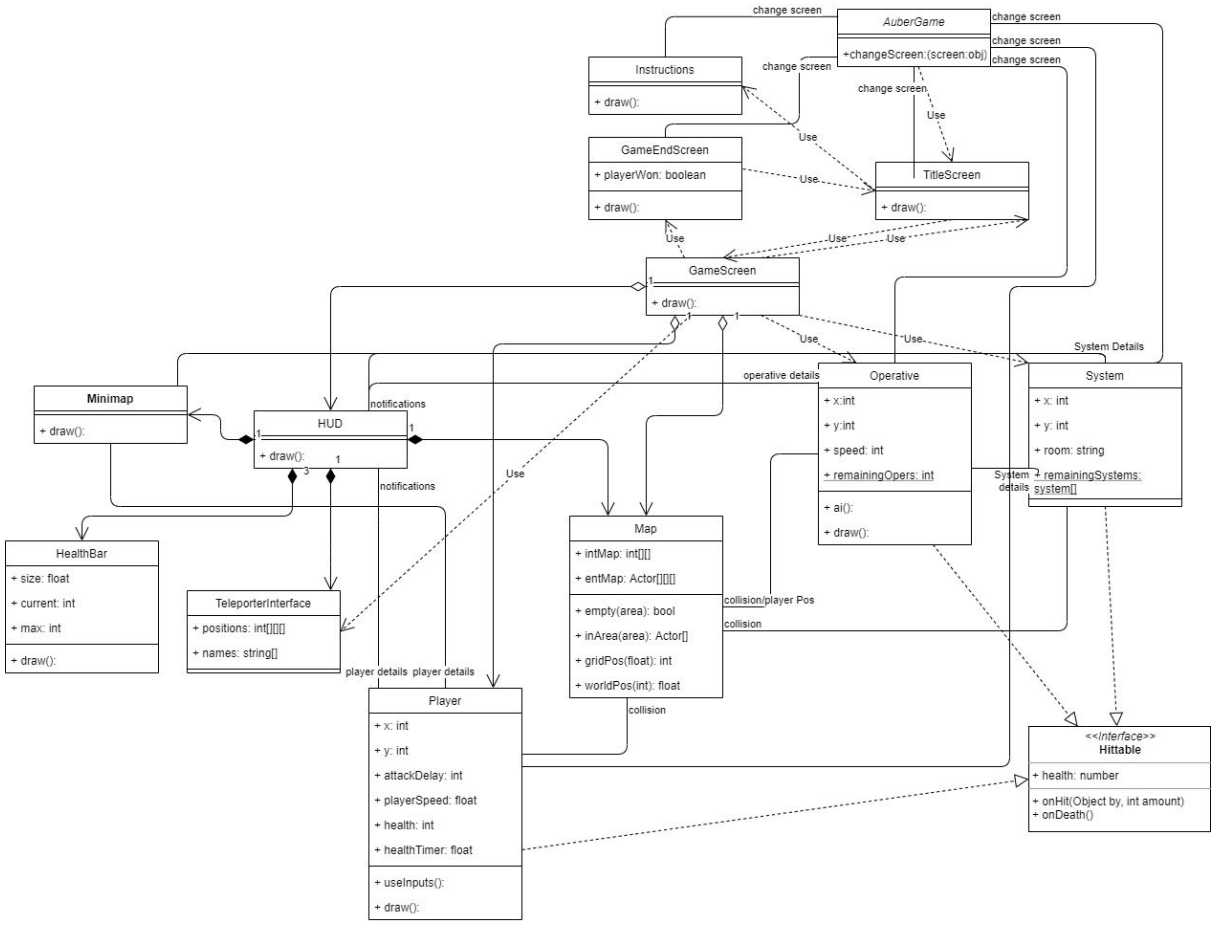
Architecture

Cohort 1, Team 4:

Robert Watts
Adam Wiegand
Josh Hall
Travis Gaffney
Xiaoyu Zou
Bogdan Lescinschi

Both the abstract and concrete representations of the architecture were created in Class UML using draw.io a free online diagram maker.

Our initial plan was to use a main AuberGame class to swap between the different game states or screens, with GameScreen (the one where you play the game) consisting of a HUD (heads up display), a map background, and multiple entities on top of that background (but behind the HUD). These entities would communicate with the map class for collison and hitboxes.



A copy of Abstract-Representation.drawio can be found at:
https://drive.google.com/drive/folders/1eBISsvp-kbzssY28SMd0jNNHQn3ll2rZ?usp=sharing

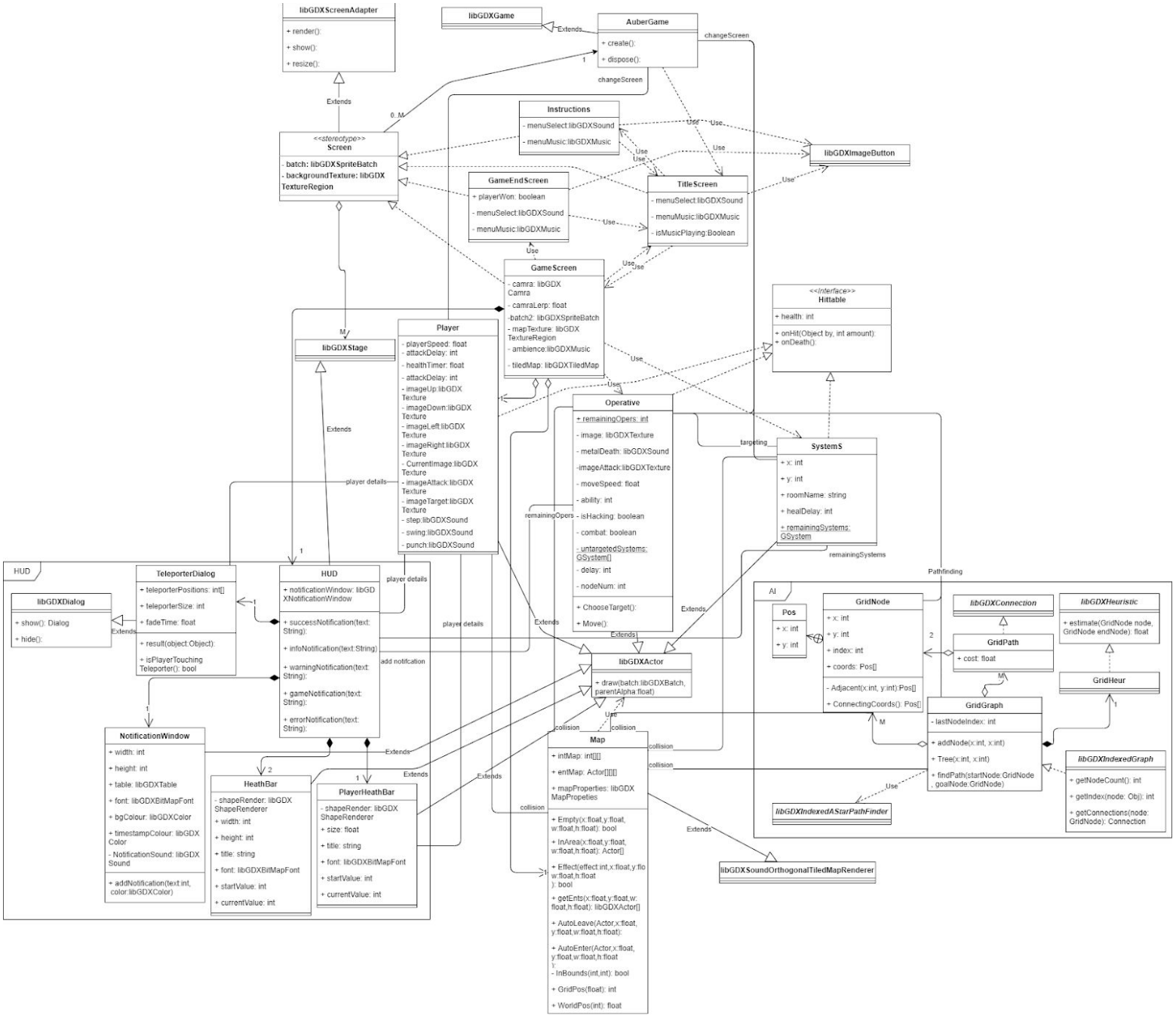
The concrete architecture follows the abstract one pretty closely other than:

- ai() and draw() being merged into one (as that's the easiest way to do it using libGDX actors)
- The minimap being replaced with a notification system (as the minimap was unnecessary for the brief and programmatically challenging).
- The drawing of the map being moved into the Map class (we changed to use a tile renderer with multiple layers, moving it to its own class helped simplify GameScreen)

Other than those some classes got more fleshed out as we realised extra requirements/useful methods for them.

The concrete representation has libGDX types included as basic types (i.e. as attribute types) only including them (without irrelevant details) when they have been inherited from or implemented. The game screens also implement a stereotype of Screen showing the common features/relationships of all the screens.

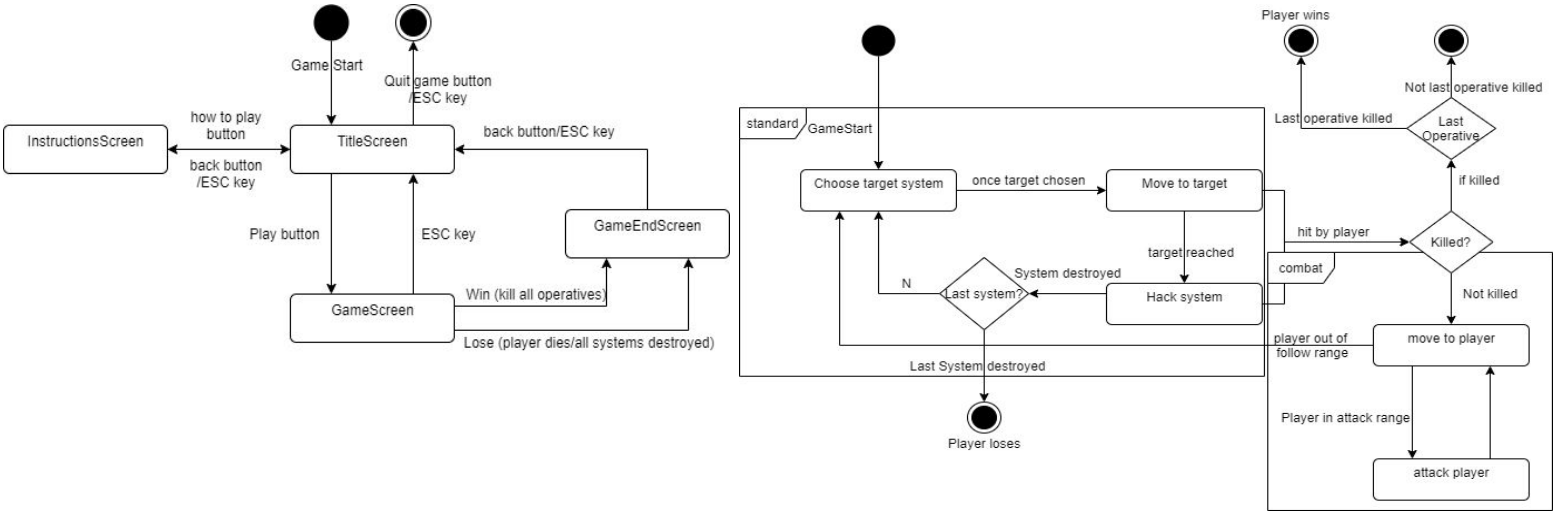
These were done to minimize the already too large diagram and to increase legibility. Concrete ClassUML.drawio.



A copy of this can be found at: https://drive.google.com/drive/folders/1eBlSsvp-kbzssY28SMd0jNNHQn3ll2rZ?usp=sharing

Below are the behaviour diagrams (UML state machine) of how the screen system and operative AI should work:

Arch1.pdf - Architecture



The abstract representation allows us to create a representation of the architecture in a low level of detail to describe the relationship between the game's classes without it being influenced by the game engine. The concrete representation however takes each aspect of the abstract representation and modifies it so that it includes the classes from the game engine and allows us to relate our own classes to them meaning the concrete representation is in a much higher level of detail and will be identical to the structure of the end product.

Function Requirement ID	Area of Architecture	Justification
FR_GAME_WIN	GameEndScreen-playerWon	If Operative.remainingOperatives = 0 then the game is won
FR_GAME_LOSE	GameEndScreen. playerWon	If the Player.onDeath() is called the game is over.
FR_GAME_REAL TIME	Player.draw()	When the player is drawn they will move according to their key presses.
FR_OPER_NUM	Operatives.remaining Operatives GameScreen.show()	Inside the GameScreen.show() 8 instances of the Operative class are created, Operatives.remainingOperatives will be equal to the number of operatives that haven't been arrested.
FR_OPER_ABILIT Y	Operative.ability	In the draw() method of operatives it checks the value of .ability and causes the effects of it
FR_OPER_SYST EM_ATTACK	Systems implements Hittable	When operatives are on a system and 'attack' it the onHit() of the system is called, which will reduce the health of the system.
FR_PLAYER_HEA LTH	Player.health Player implements	The player's onHit() can be called by operatives

	Hittable	
FR_PLAYER_ARREST	Operative implements Hittable	When the onDeath() of an operative is called they are no longer drawn and counted as arrested.
FR_PLAYER_HEAL	player.healthTimer	As long as the player is in the infirmary the healthTimer will allow their health to increase.
FR_PLAYER_TELEPORT	TeleporterDialog isPlayerTouchingTeleporter	The player is able to interact with the teleporter by pressing a key as long as they are touching the teleporter
FR_PLAYER_NOTIFIED	NotiifcationWindow.addNotification	HUD.warningNotification is called whenever an operative begins sabotaging a system.
FR_PLAYER_SYSTEM_LOCATION	System.x, System.y	Each system has a location, which allows operatives to find a path to them.
FR_MAP_SYSTEMS	System	Each system on the map an instance of the System class
FR_MAP_ROOMS	System.roomName	Each room has its own system.
FR_MAP_LAYOUT	Map.intMap	The intMap defines the collisions of the walls of the map
FR_MAP_TELEPORTERS	TeleporterDialog.teleporterPositions	There is an array of teleporter positions and TeleporterDialog.isPlayerTouchingTeleporter to decide whether the player is currently able to teleport.
FR_MAP_INFIRM	Map.effect Map.intMap	In the Map.intMap the area of the infirmary is marked with 2
FR_MAP_INFIRM_TELEPORTER	TeleporterDialog.teleporterPositions	There is a teleporter in the infirmary.
FR_MAP_INFIRM_HEAL	Map.effect(2, Player) player.healthTimer	If the Player's current position is 2 in the intMap then the player is inside the infirmary and will heal.
FR_UI_SCALABLE	GameScreen.show()	The height and width of the player's screen is used to scale the game and the UI accordingly.