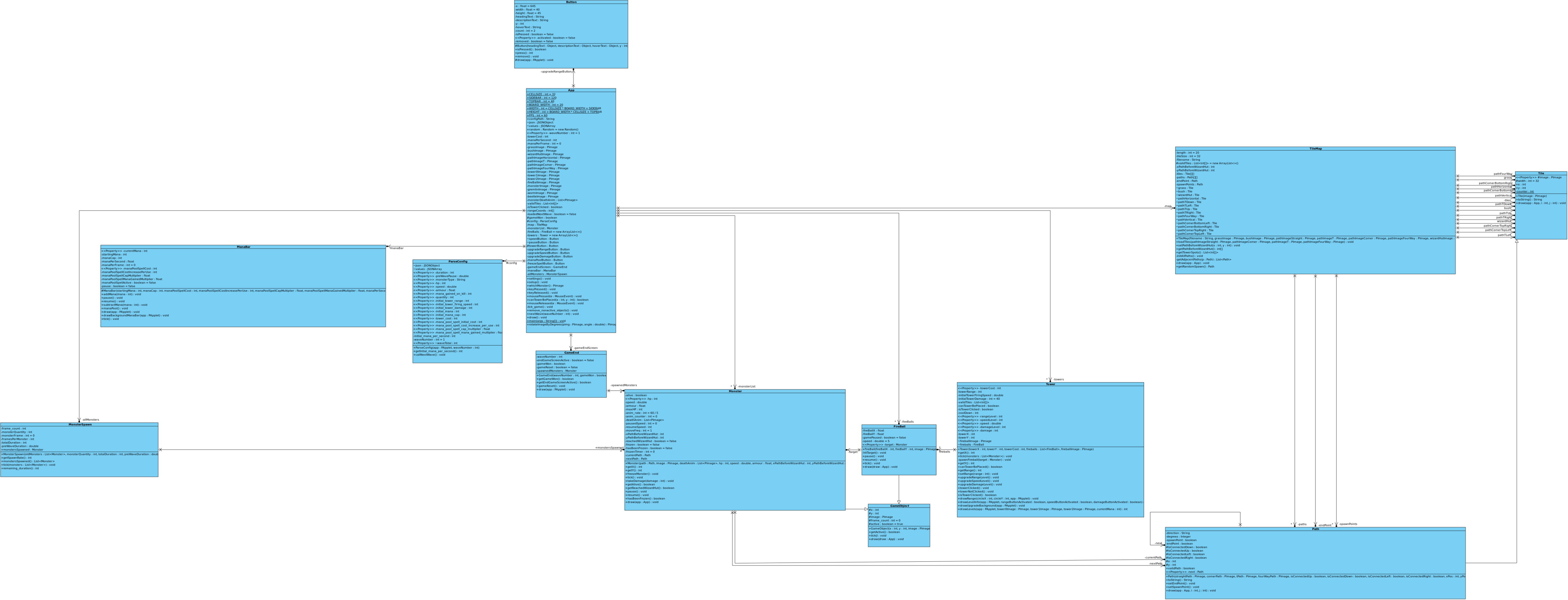
Assignment Report

While completing the assignment I tried to keep my work as object oriented as possible, This was achieved by implementing several classes, including an abstract class, to keep the code DRY (Don't Repeat Yourself) and make it more understandable for those who read it. For the app.java file I tried to code as little game mechanics as possible inside this file and instead, used this file to instantiate other objects which handled game mechanics inside of their own classes. This approach worked well for classes such as buttons, allowing me to avoid code repetition. I created 9 instances of a button class, each with different behaviour based on their activation status. However, they all shared similar design such as design and position.

Other parts of the code which I was able to separate included all the mana calculation and aspects of the game such as Mana Pool. I used a Mana Pool class which then used getters and setters to update the mana accordingly without having to do it all in the draw function. Furthermore, extensive parts of the code such as tower upgrades which required an extensive amount of if statements to decide whether a new tower image should be rendered or not were successfully able to be completed in the tower class.

While these classes were successful, one problem I discovered was that by having several classes even these classes can repeat some of their code within other classes. This led me to use an abstract class which created foundational methods and variables that I could call in other classes without having to repeat myself. A successful example of this is the game object class. This game object was for any parts of the game which share methods such as “get active”, “tick” and “draw”. This was used for objects which move in the game and are drawn based on whether they are active or not. I found this to work successfully with the monster and fireball class Although they serve different functions, in the game, both the monster and fireball classes share common behaviours. They move throughout the map when active and are removed from the game when inactive. Therefore, making them eligible for an Abstract Class.

As part of the extension, I implemented a freeze spell that creates an ice block around monsters when activated. The ice covers the monster and begins to violently shake while it surrounds the monster. After a few seconds the monster can break through, it then becomes angry and turns red. The monster also moves at a speed slightly higher than its original speed. The freeze spell cost a small amount of mana due to its double-edged sword nature. When the freeze spell is applied it freezers all current monsters on the map.


Button
-x : float = 645
-width : float = 40
-height : float = 45
-headingText : String
-descriptionText : String
-y : int
-hoverText : String
-count : int = 2
-isPressed : boolean = false
<<Property>> -activated : boolean = false
-removed : boolean = false
#Button(headingText : Object, descriptionText : Object, hoverText : Object, y : int
+isPressed() : boolean
+press() : int
+remove() : void
#draw(app : PApplet) : void
-upgradeRangeButton
App
+CELLSIZE : int = 32
+SIDEBAR : int = 120
+TOPBAR : int = 40
+BOARD_WIDTH : int = 20
+WIDTH : int = CELLSIZE * BOARD_WIDTH + SIDEBAR
+HEIGHT : int = BOARD_WIDTH * CELLSIZE + TOPBAR
+FPS : int = 60
+configPath : String
~json : JSONObject
~values : JSONArray
+random : Random = new Random()
<<Property>> -waveNumber : int = 1
-towerCost : int
-manaPerSecond : int
-manaPerFrame : int = 0
-grassImage : PImage
-bushImage : PImage
-wizardHutImage : PImage
-pathImageHorizontal : PImage
-pathImageT : PImage
-pathImageCorner : PImage
-pathImageFourWay : PImage
-tower0Image : PImage
-tower1Image : PImage
-tower2Image : PImage
-fireBallImage : PImage
-monsterImage : PImage
-gremlinImage : PImage
-wormImage : PImage
-beetleImage : PImage
-monsterDeathAnim : List<PImage>
-validTiles : List<int[]>
-isTowerClicked : boolean
-rangeCoords : int[]
-loadedNextWave : boolean = false
#gameWon : boolean
#config : ParseConfig
-map : TileMap
-monsterList : Monster
-fireBalls : FireBall = new ArrayList<>()
-towers : Tower = new ArrayList<>()
~speedButton : Button
~pauseButton : Button
#towerButton : Button
-upgradeRangeButton : Button
-upgradeSpeedButton : Button
-upgradeDamageButton : Button
-manaPoolButton : Button
-freezeSpellButton : Button
-gameEndScreen : GameEnd
-manaBar : ManaBar
-allMonsters : MonsterSpawn
+settings() : void
+setup() : void
+whichMonster() : PImage
+keyPressed() : void
+keyReleased() : void
+mousePressed(e : MouseEvent) : void
+canTowerBePlaced(x : int, y : int) : boolean
+mouseReleased(e : MouseEvent) : void
-tick_game() : void
+remove_nonactive_objects() : void
+nextWave(waveNumber : int) : void
+draw() : void
+main(args : String[]) : void
+rotateImageByDegrees(pimg : PImage, angle : double) : PImag
-manaBar
ManaBar
<<Property>> -currentMana : int
-startingMana : int
-manaCap : int
-manaPerSecond : float
-manaPerFrame : int = 0
<<Property>> -manaPoolSpellCost : int
-manaPoolSpellCostIncreasePerUse : int
-manaPoolSpellCapMultiplier : float
-manaPoolSpellManaGainedMultiplier : float
-manaPoolSpellActive : boolean = false
-pause : boolean = false
#ManaBar(startingMana : int, manaCap : int, manaPoolSpellCost : int, manaPoolSpellCostIncreasePerUse : int, manaPoolSpellCapMultiplier : float, manaPoolSpellManaGainedMultiplier : float, manaPerSeco
+addMana(mana : int) : void
+pause() : void
+resume() : void
+subtractMana(mana : int) : void
+manaPool() : void
+draw(app : PApplet) : void
+drawBackgroundManaBar(app : PApplet) : void
+tick() : void
#config
ParseConfig
~json : JSONObject
~values : JSONArray
<<Property>> -duration : int
<<Property>> -preWavePause : double
<<Property>> -monsterType : String
<<Property>> -hp : int
<<Property>> -speed : double
<<Property>> -armour : float
<<Property>> -mana_gained_on_kill : int
<<Property>> -quantity : int
<<Property>> -initial_tower_range : int
<<Property>> -initial_tower_firing_speed : int
<<Property>> -initial_tower_damage : int
<<Property>> -initial_mana : int
<<Property>> -initial_mana_cap : int
<<Property>> -tower_cost : int
<<Property>> -mana_pool_spell_initial_cost : int
<<Property>> -mana_pool_spell_cost_increase_per_use : int
<<Property>> -mana_pool_spell_cap_multiplier : float
<<Property>> -mana_pool_spell_mana_gained_multiplier : floa
-initial_mana_per_second : int
-waveNumber : int = 1
<<Property>> ~waveTotal : int
+ParseConfig(app : PApplet, waveNumber : int)
+getIntial_mana_per_second() : int
+callNextWave() : void
-gameEndScreen
GameEnd
-waveNumber : int
-endGameScreenActive : boolean = false
-gameWon : boolean
-gameReset : boolean = false
-spawnedMonsters : Monster
+GameEnd(waveNumber : int, gameWon : boolea
+getGameWon() : boolean
+getEndGameScreenActive() : boolean
+gameReset() : void
+draw(app : PApplet) : void
-spawnedMonsters
*
*
-monsterList
Monster
-alive : boolean
<<Property>> -hp : int
-speed : double
-armour : float
-maxHP : int
-anim_rate : int = 60 / 5
-anim_counter : int = 0
-deathAnim : List<PImage>
-pausedSpeed : int = 0
-resumeSpeed : int
-moveFreq : int = 1
-xPathBeforeWizardHut : int
-yPathBeforeWizardHut : int
-reachedWizardHut : boolean = false
-frozen : boolean = false
-hasBeenFrozen : boolean = false
-frozenTimer : int = 0
-currentPath : Path
-nextPath : Path
+Monster(path : Path, image : PImage, deathAnim : List<PImage>, hp : int, speed : double, armour : float, xPathBeforeWizardHut : int, yPathBeforeWizardHut :
+getX() : int
+getY() : int
+freezeMonster() : void
+tick() : void
+takeDamage(damage : int) : void
+getAlive() : boolean
+getReachedWizardHut() : boolean
+pause() : void
+resume() : void
+hasBeenFrozen() : boolean
+draw(app : App) : void
-allMonsters
MonsterSpawn
-frame_count : int
-monsterQuantity : int
-monsterFrame : int = 0
-framesPerMonster : int
-totalDuration : int
-preWaveDuration : double
+monstersSpawned : Monster
+MonsterSpawn(oldMonsters : List<Monster>, monsterQuantity : int, totalDuration : int, preWaveDuration : doub
+getSpawnRate() : int
+monstersSpawned() : List<Monster>
+tick(monsters : List<Monster>) : void
+remaining_duration() : int
+monstersSpawned
*
*
-fireBalls
FireBall
-fireBallX : float
-fireBallY : float
-gamePaused : boolean = false
-speed : double = 5
<<Property>> -target : Monster
+FireBall(fireBallX : int, fireBallY : int, image : PImage
-hitTarget() : void
+pause() : void
+resume() : void
+tick() : void
+draw(draw : App) : void
-target
*
fireballs
GameObject
#x : int
#y : int
#image : PImage
#frame_count : int = 0
#active : boolean = true
+GameObject(x : int, y : int, image : PImage
+getActive() : boolean
+tick() : void
+draw(draw : App) : void
*
-towers
Tower
<<Property>> -towerCost : int
-towerRange : int
-initialTowerFiringSpeed : double
-initialTowerDamage : int = 40
-validTiles : List<int[]>
-canTowerBePlaced : boolean
-isTowerClicked : boolean
-coolDown : int
<<Property>> -rangeLevel : int
<<Property>> -speedLevel : int
<<Property>> -speed : double
<<Property>> -damageLevel : int
<<Property>> -damage : int
-towerX : int
-towerY : int
~fireballImage : PImage
~fireballs : FireBall
+Tower(towerX : int, towerY : int, towerCost : int, fireballs : List<FireBall>, fireballImage : PImage)
+getX() : int
+tick(monsters : List<Monster>) : void
-spawnFireball(target : Monster) : void
+getY() : int
+canTowerBePlaced() : boolean
+getRange() : int
+setRange(range : int) : void
+upgradeRangeLevel() : void
+upgradeSpeedLevel() : void
+upgradeDamageLevel() : void
+towerClicked() : void
+towerNotClicked() : void
+isTowerClicked() : boolean
+drawRange(circleX : int, circleY : int, app : PApplet) : void
+drawLevelInfo(app : PApplet, rangeButtonActivated : boolean, speedButtonActivated : boolean, damageButtonActivated : boolean) :
+drawUpgradeBackground(app : PApplet) : void
+drawLevels(app : PApplet, tower0Image : PImage, tower1Image : PImage, tower2Image : PImage, currentMana : int) : int
-map
TileMap
-length : int = 20
-tileSize : int = 32
-filename : String
#validTiles : List<int[]> = new ArrayList<>()
-xPathBeforeWizardHut : int
-yPathBeforeWizardHut : int
-tiles : Tile[][]
-paths : Path[][]
-endPoint : Path
-spawnPoints : Path
~grass : Tile
~bush : Tile
~wizardHut : Tile
~pathHorizontal : Tile
~pathTDown : Tile
~pathTLeft : Tile
~pathTUp : Tile
~pathTRight : Tile
~pathFourWay : Tile
~pathVertical : Tile
~pathCornerBottomLeft : Tile
~pathCornerBottomRight : Tile
~pathCornerTopRight : Tile
~pathCornerTopLeft : Tile
+TileMap(filename : String, grassImage : PImage, bushImage : PImage, pathImageStraight : PImage, pathImageT : PImage, pathImageCorner : PImage, pathImageFourWay : PImage, wizardHutImage :
+loadTiles(pathImageStraight : PImage, pathImageCorner : PImage, pathImageT : PImage, pathImageFourWay : PImage) : void
+setPathBeforeWizardHut(x : int, y : int) : void
+getPathBeforeWizardHut() : int[]
+getTowerSpots() : List<int[]>
-initAllPaths() : void
-getAdjacentPaths(p : Path) : List<Path>
+draw(app : App) : void
+getRandomSpawn() : Path
pathFourWay
grass
pathCornerBottomRight
pathHorizontal
pathCornerBottomLeft
pathVertical
-tiles
pathTDown
bush
pathTUp
pathTRight
wizardHut
pathCornerTopRight
pathCornerTopLeft
pathTLeft
Tile
<<Property>> #image : PImage
#width : int = 32
+x : int
+y : int
-counter : int
+Tile(image : PImage)
+toString() : String
+draw(app : App, i : int, j : int) : void
*
-paths
-endPoint
*
-spawnPoints
-next
-currentPath
-nextPath
Path
-direction : String
-degrees : Integer
-spawnPoint : boolean
-endPoint : boolean
#isConnectedDown : boolean
#isConnectedUp : boolean
#isConnectedLeft : boolean
#isConnectedRight : boolean
#x : int
#y : int
+validPath : boolean
<<Property>> -next : Path
+Path(straightPath : PImage, cornerPath : PImage, tPath : PImage, fourWayPath : PImage, isConnectedUp : boolean, isConnectedDown : boolean, isConnectedLeft : boolean, isConnectedRight : boolean, xPos : int, yPo
+toString() : String
+setEndPoint() : void
+setSpawnPoint() : void
+draw(app : App, i : int, j : int) : void