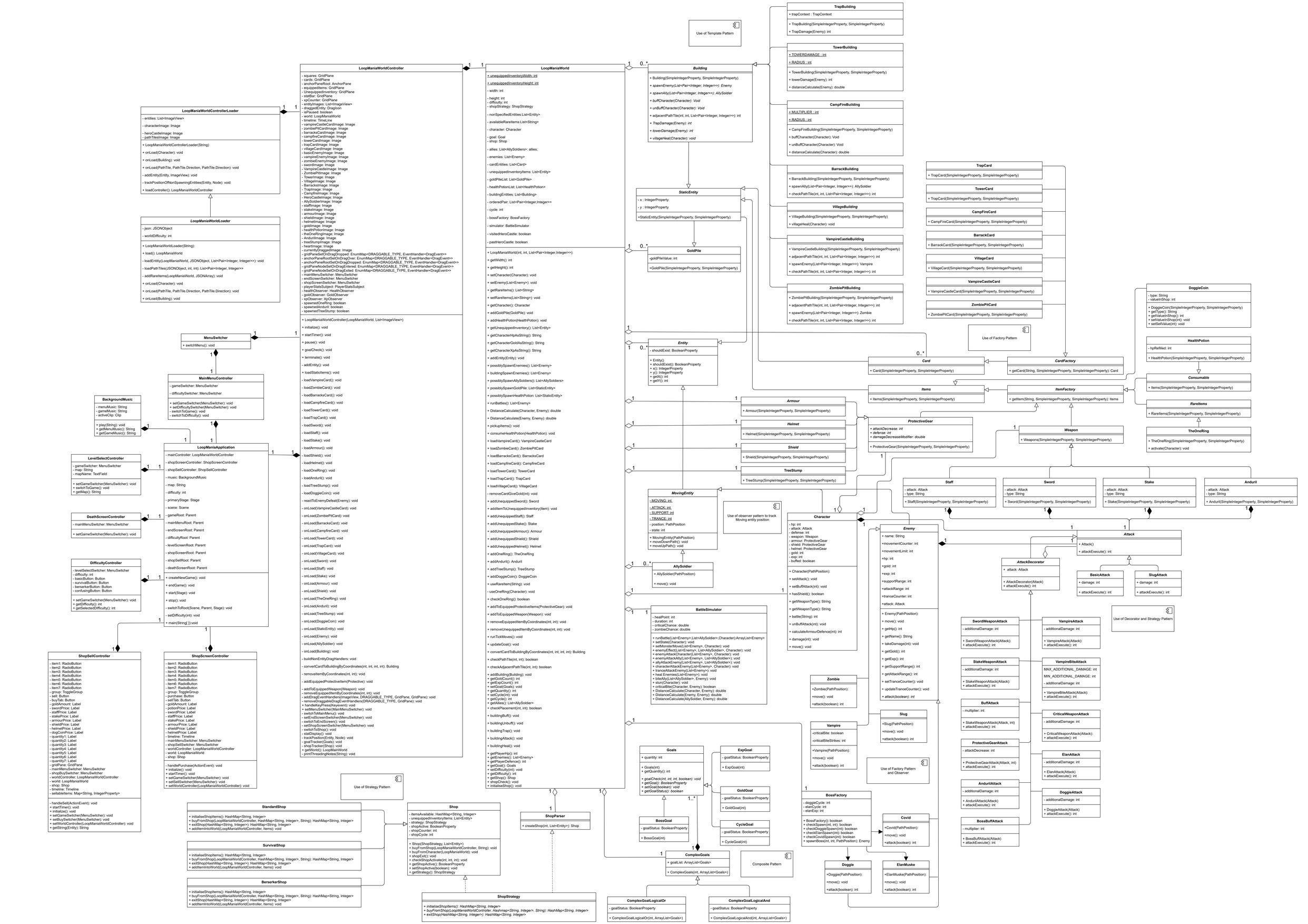
Use of Template Pattern
TrapBuilding
+ trapContext : TrapContext
+ TrapBuilding(SimpleIntegerProperty, SimpleIntegerProperty)
+ TrapDamage(Enemy): int
TowerBuilding
+ TOWERDAMAGE : int
+ RADIUS : int
+ TowerBuilding(SimpleIntegerProperty, SimpleIntegerProperty)
+ towerDamage(Enemy): int
+ distanceCalculate(Enemy): double
CampFireBuilding
+ MULTIPLIER : int
+ RADIUS : int
+ CampFireBuilding(SimpleIntegerProperty, SimpleIntegerProperty)
+ buffCharacter(Character): Void
+ unBuffCharacter(Character): Void
+ distanceCalculate(Character): double
BarrackBuilding
+ BarrackBuilding(SimpleIntegerProperty, SimpleIntegerProperty)
+ spawnAlly(List<Pair<Integer, Integer>>): AllySoldier
+ checkPathTile(int, int, List<Pair<Integer, Integer>>): int
VillageBuilding
+ VillageBuilding(SimpleIntegerProperty, SimpleIntegerProperty)
+ villageHeal(Character): void
VampireCastleBuilding
+ VampireCastleBuilding(SimpleIntegerProperty, SimpleIntegerProperty)
+ adjacentPathTile(int, int, List<Pair<Integer, Integer>>): int
+ spawnEnjoy(List<Pair<Integer, Integer>>): Vampire
+ checkPathTile(int, int, List<Pair<Integer, Integer>>): int
ZombiePitBuilding
+ ZombiePitBuilding(SimpleIntegerProperty, SimpleIntegerProperty)
+ adjacentPathTile(int, int, List<Pair<Integer, Integer>>): int
+ spawnEnemy(List<Pair<Integer, Integer>>): Zombie
+ checkPathTile(int, int, List<Pair<Integer, Integer>>): int
Building
+ Building(SimpleIntegerProperty, SimpleIntegerProperty)
+ spawnEnemy(List<Pair<Integer, Integer>>): Enemy
+ spawnAlly(List<Pair<Integer, Integer>>): AllySoldier
+ buffCharacter(Character): Void
+ unBuffCharacter(Character): Void
+ adjacentPathTile(int, int, List<Pair<Integer, Integer>>): int
+ TrapDamage(Enemy): int
+ towerDamage(Enemy): int
+ villageHeal(Character): void
LoopManiaWorldController
- squares: GridPlane
- cards: GridPlane
- anchorPaneRoot: AnchorPane
- equippedItems: GridPlane
- UnequippedInventory: GridPlane
- statBar: GridPlane
- xpCounter: GridPlane
- entityImages: List<ImageView>
- draggedEntity: DragIcon
- isPaused: boolean
- world: LoopManiaWorld
- timeline: TimeLine
- vampireCastleCardImage: Image
- zombiePitCardImage: Image
- barracksCardImage: Image
- campfireCardImage: Image
- towerCardImage: Image
- trapCardImage: Image
- villageCardImage: Image
- basicEnemyImage: Image
- vampireEnemyImage: Image
- zombieEnemyImage: Image
- swordImage: Image
- VampireCastleImage: Image
- ZombiePitImage: Image
- TowerImage: Image
- VillageImage: Image
- BarracksImage: Image
- TrapImage: Image
- CampfireImage: Image
- HeroCastleImage: Image
- AllySoldierImage: Image
- staffImage: Image
- stakeImage: Image
- armourImage: Image
- shieldImage: Image
- helmetImage: Image
- goldImage: Image
- healthPotionImage: Image
- theOneRingImage: Image
- AnduriIImage: Image
- treeStumpImage: Image
- heartImage: Image
- currentlyDraggedImage: Image
- gridPaneSetOnDragDropped: EnumMap<DRAGGABLE_TYPE, EventHandler<DragEvent>>
- anchorPaneRootSetOnDragOver: EnumMap<DRAGGABLE_TYPE, EventHandler<DragEvent>>
- anchorPaneRootSetOnDragDropped: EnumMap<DRAGGABLE_TYPE, EventHandler<DragEvent>>
- gridPaneNodeSetOnDragEntered: EnumMap<DRAGGABLE_TYPE, EventHandler<DragEvent>>
- gridPaneNodeSetOnDragExited: EnumMap<DRAGGABLE_TYPE, EventHandler<DragEvent>>
- mainMenuSwitcher: MenuSwitcher
- endScreenSwitcher: MenuSwitcher
- shopScreenSwitcher: MenuSwitcher
- playerStatsSubject: PlayerStatsSubject
- healthObserver: HealthObserver
- goldObserver: GoldObserver
- xpObserver: XpObserver
- spawnedOneRing: boolean
- spawnedAnduril: boolean
- spawnedTreeStump: boolean
+ LoopManiaWorldController(LoopManiaWorld, List<ImageView>)
+ initialize(): void
+ startTimer(): void
+ pause(): void
+ goalCheck(): void
+ terminate(): void
- addEntity(): void
+ loadStaticItems(): void
+ loadVampireCard(): void
+ loadZombieCard(): void
+ loadBarracksCard(): void
+ loadCampfireCard(): void
+ loadTowerCard(): void
+ loadTrapCard(): void
+ loadSword(): void
+ loadStaff(): void
+ loadStake(): void
+ loadArmour(): void
+ loadShield(): void
+ loadHelmet(): void
+ loadOneRing(): void
+ loadAnduril(): void
+ loadTreeStump(): void
+ loadDoggieCoin(): void
+ reactToEnemyDefeat(Enemy): void
+ onLoad(VampireCastleCard): void
+ onLoad(ZombiePitCard): void
+ onLoad(BarracksCard): void
+ onLoad(CampfireCard): void
+ onLoad(TowerCard): void
+ onLoad(TrapCard): void
+ onLoad(VillageCard): void
+ onLoad(Sword): void
+ onLoad(Staff): void
+ onLoad(Stake): void
+ onLoad(Armour): void
+ onLoad(Shield): void
+ onLoad(TheOneRing): void
+ onLoad(Anduril): void
+ onLoad(TreeStump): void
+ onLoad(DoggieCoin): void
+ onLoad(StaticEntity): void
+ onLoad(Enemy): void
+ onLoad(AllySoldier): void
+ onLoad(Building): void
- buildNonEntityDragHandlers: void
- convertCardToBuildingByCoordinates(int, int, int, int): Building
- removeItemByCoordinates(int, int): void
- addEquippedProtectiveItem(Protective): void
- addToEquippedWeapon(Weapon): void
- removeEquippedItemByCoordinates(int, int): void
- addDragEventHandlers(ImageView, DRAGGABLE_TYPE, GridPane, GridPane): void
- removeDraggableDragEventHandlers(DRAGGABLE_TYPE, GridPane): void
+ handleKeyPress(KeyEvent): void
+ setMenuSwitcher(MainMenuSwitcher): void
+ switchToMainMenu(): void
+ setEndScreenSwitcher(MenuSwitcher): void
+ switchToEndScreen(): void
+ setShopScreenSwitcher(MenuSwitcher): void
+ switchToShop(): void
+ statDisplay(): void
+ trackPosition(Entity, Node): void
+ goalTracker(Goals): void
+ shopTracker(Shop): void
+ getWorld(): LoopManiaWorld
+ printThreadingNotes(String): void
LoopManiaWorldControllerLoader
- entities: List<ImageView>
- characterImage: Image
- heroCastleImage: Image
- pathTilesImage: Image
+ LoopManiaWorldControllerLoader(String)
+ onLoad(Character): void
+ onLoad(Building): void
+ onLoad(PathTile, PathTile.Direction, PathTile.Direction): void
- addEntity(Entity, ImageView): void
- trackPositionOfNonSpawningEntities(Entity, Node): void
+ loadController(): LoopManiaWorldController
LoopManiaWorldLoader
- json: JSONObject
- worldDifficulty: int
+ LoopManiaWorldLoader(String)
+ load(): LoopManiaWorld
- loadEntity(LoopManiaWorld, JSONObject, List<Pair<Integer, Integer>>): void
- loadPathTiles(JSONObject, int, int): List<Pair<Integer, Integer>>
- addRareItems(LoopManiaWorld, JSONArray): void
+ onLoad(Character): void
+ onLoad(PathTile, PathTile.Direction, PathTile.Direction): void
+ onLoad(Building): void
MenuSwitcher
+ switchMenu(): void
MainMenuController
- gameSwitcher: MenuSwitcher
- difficultySwitcher: MenuSwitcher
+ setGameSwitcher(MenuSwitcher): void
+ setDifficultySwitcher(MenuSwitcher): void
+ switchToGame(): void
+ switchToDifficulty(): void
BackgroundMusic
- menuMusic: String
- gameMusic: String
- activeClip: Clip
+ play(String): void
+ getMenuMusic(): String
+ getGameMusic(): String
LoopManiaApplication
- mainController: LoopManiaWorldController
- shopScreenController: ShopScreenController
- shopSellController: ShopSellController
- music: BackgroundMusic
- map: String
- difficulty: int
- primaryStage: Stage
- scene: Scene
- gameRoot: Parent
- mainMenuRoot: Parent
- endScreenRoot: Parent
- difficultyRoot: Parent
- levelScreenRoot: Parent
- shopScreenRoot: Parent
- shopSellRoot: Parent
- deathScreenRoot: Parent
+ createNewGame(): void
+ endGame(): void
+ start(Stage): void
+ stop(): void
- switchToRoot(Scene, Parent, Stage): void
- setDifficulty(int): void
+ main(String[ ]):void
LevelSelectController
- gameSwitcher: MenuSwitcher
- map: String
- mapName: TextField
+ setGameSwitcher(MenuSwitcher): void
+ switchToGame(): void
+ getMap(): String
DeathScreenController
- mainMenuSwitcher: MenuSwitcher
+ setGameSwitcher(MenuSwitcher): void
DifficultyController
- levelSelectSwitcher: MenuSwitcher
- difficulty: int
- basicButton: Button
- survivalButton: Button
- berserkerButton: Button
- confusingButton: Button
+ setGameSwitcher(MenuSwitcher): void
+ getDifficulty(): int
+ getSelectedDifficulty(): int
ShopSellController
- item1: RadioButton
- item2: RadioButton
- item3: RadioButton
- item4: RadioButton
- item5: RadioButton
- item6: RadioButton
- item7: RadioButton
- group: ToggleGroup
- sell: Button
- buyTab: Button
- goldAmount: Label
- swordPrice: Label
- staffPrice: Label
- stakePrice: Label
- armourPrice: Label
- shieldPrice: Label
- helmetPrice: Label
- dogCoinPrice: Label
- quantity1: Label
- quantity2: Label
- quantity3: Label
- quantity4: Label
- quantity5: Label
- quantity6: Label
- quantity7: Label
- gridPane: GridPane
- mainMenuSwitcher: MenuSwitcher
- shopBuySwitcher: MenuSwitcher
- worldController: LoopManiaWorldController
- world: LoopManiaWorld
- shop: Shop
- timeline: Timeline
- sellableItems: Map<String, IntegerProperty>
- handleSell(ActionEvent): void
+ startTimer(): void
+ initialize(): void
+ setGameSwitcher(MenuSwitcher): void
+ setBuySwitcher(MenuSwitcher): void
+ setWorldController(LoopManiaWorldController): void
+ getString(Entity): String
ShopScreenController
- item1: RadioButton
- item2: RadioButton
- item3: RadioButton
- item4: RadioButton
- item5: RadioButton
- item6: RadioButton
- item7: RadioButton
- group: ToggleGroup
- purchase: Button
- sellTab: Button
- goldAmount: Label
- potionPrice: Label
- swordPrice: Label
- staffPrice: Label
- stakePrice: Label
- armourPrice: Label
- shieldPrice: Label
- helmetPrice: Label
- timeline: Timeline
- mainMenuSwitcher: MenuSwitcher
- shopSellSwitcher: MenuSwitcher
- worldController: LoopManiaWorldController
- world: LoopManiaWorld
- shop: Shop
- handlePurchase(ActionEvent): void
+ initialize(): void
+ startTimer(): void
+ setGameSwitcher(MenuSwitcher): void
+ setSellSwitcher(MenuSwitcher): void
+ setWorldController(LoopManiaWorldController): void
LoopManiaWorld
+ unequippedInventoryWidth: int
+ unequippedInventoryHeight: int
- width: int
- height: int
- difficulty: int
- shopStrategy: ShopStrategy
- nonSpecifiedEntities:List<Entity>
- availableRareItems:List<String>
- character: Character
- goal: Goal
- shop: Shop
- allies: List<AllySoldiers>: allies;
- enemies: List<Enemy>
- cardEntities: List<Card>
- unequippedInventoryItems: List<Entity>
- goldPileList: List<GoldPile>
- healthPotionList: List<HealthPotion>
- buildingEntities: List<Building>
- orderedPair: List<Pair<Integer,Integer>>
- cycle: int
- bossFactory: BossFactory
- simulator: BattleSimulator
- visitedHeroCastle: boolean
- pastHeroCastle: boolean
+ LoopManiaWorld(int, int, List<Pair<Integer,Integer>>):
+ getWidth(): int
+ getHeight(): int
+ setCharacter(Character): void
+ setEnemy(List<Enemy>): void
+ getRareItems(): List<String>
+ setRareItems(List<String>): void
+ getCharacter(): Character
+ addGoldPile(GoldPile): void
+ addHealthPotion(HealthPotion): void
+ getUnequippedInventory(): List<Entity>
+ getCharacterHpAsString(): String
+ getCharacterGoldAsString(): String
+ getCharacterXpAsString(): String
+ addEntity(Entity): void
+ possiblySpawnEnemies(): List<Enemy>
+ buildingSpawnEnemies(): List<Enemy>
+ possiblySpawnAllySoldiers(): List<AllySoldiers>
+ possiblySpawnGoldPile: List<StaticEntity>
+ possiblySpawnHealthPotion: List<StaticEntity>
+ runBattles(): List<Enemy>
+ DistanceCalculate(Character, Enemy): double
+ DistanceCalculate(Enemy, Enemy): double
+ pickupItems(): void
+ consumeHealthPotion(HealthPotion): void
+ loadVampireCard(): VampireCastleCard
+ loadZombieCard(): ZombiePitCard
+ loadBarracksCard(): BarracksCard
+ loadCampfireCard(): CampfireCard
+ loadTowerCard(): TowerCard
+ loadTrapCard(): TrapCard
+ loadVillageCard(): VillageCard
+ removeCardGiveGold(int): void
+ addUnequippedSword(): Sword
+ addItemToUnequippedInventory(Item): void
+ addUnequippedStaff(): Staff
+ addUnequippedStake(): Stake
+ addUnequippedArmour(): Armour
+ addUnequippedShield(): Shield
+ addUnequippedHelmet(): Helmet
+ addOneRing(): TheOneRing
+ addAnduril(): Anduril
+ addTreeStump(): TreeStump
+ addDoggieCoin(): DoggieCoin
+ useRareItem(String): void
- useOneRing(Character): void
+ checkOneRing(): boolean
+ addToEquippedProtectiveItems(ProtectiveGear): void
+ addToEquippedWeapon(Weapon): void
+ removeEquippedItemByCoordinates(int, int): void
+ removeUnequippedItemByCoordinates(int, int): void
+ runTickMoves(): void
+ updateGoal(): void
+ convertCardToBuildingByCoordinates(int, int, int, int): Building
+ checkPathTile(int, int): boolean
+ checkAdjacentPathTile(int, int): boolean
+ addBuilding(Building): void
+ getGoldCount(): int
+ getExpCount(): int
+ setGoal(Goals): void
+ getQuantity(): int
+ setCycle(int): void
+ getCycle(): int
+ getAllies(): List<AllySoldier>
+ checkPlacement(int, int): boolean
+ buildingBuff(): void
+ buildingUnbuff(): void
+ buildingTrap(): void
+ buildingAttack(): void
+ buildingHeal(): void
+ getPlayerHp(): int
+ getEnemies(): List<Enemy>
+ getPlayerDefence(): int
+ getGoal(): Goals
+ setDifficulty(int): void
+ getDifficulty(): int
+ getShop(): Shop
+ shopCheck(): void
+ initialiseShop(): void
StaticEntity
- x : IntegerProperty
- y : IntegerProperty
+StaticEntity(SimpleIntegerProperty, SimpleIntegerProperty)
GoldPile
-goldPileValue: int
+GoldPile(SimpleIntegerProperty, SimpleIntegerProperty)
Entity
- shouldExist: BooleanProperty
+ Entity()
+ shouldExist(): BooleanProperty
+ x(): IntegerProperty
+ y(): IntegerProperty
+ getX(): int
+ getY(): int
Use of Factory Pattern
Card
+ Card(SimpleIntegerProperty, SimpleIntegerProperty)
CardFactory
+ getCard(String, SimpleIntegerProperty, SimpleIntegerProperty): Card
Items
+ Items(SimpleIntegerProperty, SimpleIntegerProperty)
ItemFactory
+ getItem(String, SimpleIntegerProperty, SimpleIntegerProperty): Items
TrapCard
+ TrapCard(SimpleIntegerProperty, SimpleIntegerProperty)
TowerCard
+ TrapCard(SimpleIntegerProperty, SimpleIntegerProperty)
CampFireCard
+ CampFireCard(SimpleIntegerProperty, SimpleIntegerProperty)
BarrackCard
+ BarrackCard(SimpleIntegerProperty, SimpleIntegerProperty)
VillageCard
+ VillageCard(SimpleIntegerProperty, SimpleIntegerProperty)
VampireCastleCard
+ VampireCastleCard(SimpleIntegerProperty, SimpleIntegerProperty)
ZombiePitCard
+ ZombiePitCard(SimpleIntegerProperty, SimpleIntegerProperty)
DoggieCoin
- type: String
- valueInShop: int
+ DoggieCoin(SimpleIntegerProperty, SimpleIntegerProperty)
+ getType(): String
+ getValueInShop(): int
+ setValueInShop(int): void
+ setSellValue(int): void
HealthPotion
- hpRefilled: int
+ HealthPotion(SimpleIntegerProperty, SimpleIntegerProperty)
Consumable
+ Items(SimpleIntegerProperty, SimpleIntegerProperty)
RareItems
+ RareItems(SimpleIntegerProperty, SimpleIntegerProperty)
TheOneRing
+ TheOneRing(SimpleIntegerProperty, SimpleIntegerProperty)
+ activate(Character): void
Armour
+ Armour(SimpleIntegerProperty, SimpleIntegerProperty)
Helmet
+ Helmet(SimpleIntegerProperty, SimpleIntegerProperty)
Shield
+ Shield(SimpleIntegerProperty, SimpleIntegerProperty)
TreeStump
+ TreeStump(SimpleIntegerProperty, SimpleIntegerProperty)
ProtectiveGear
+ attackDecrease: int
+ defense: int
+ damageDecreaseModifier: double
+ ProtectiveGear(SimpleIntegerProperty, SimpleIntegerProperty)
Weapon
+ Weapons(SimpleIntegerProperty, SimpleIntegerProperty)
Staff
- attack: Attack
- type: String
+ Staff(SimpleIntegerProperty, SimpleIntegerProperty)
Sword
- attack: Attack
- type: String
+ Sword(SimpleIntegerProperty, SimpleIntegerProperty)
Stake
- attack: Attack
- type: String
+ Stake(SimpleIntegerProperty, SimpleIntegerProperty)
Anduril
- attack: Attack
- type: String
+ Anduril(SimpleIntegerProperty, SimpleIntegerProperty)
MovingEntity
- MOVING: int
- ATTACK: int
- SUPPORT: int
- TRANCE: int
- position: PathPosition
- state: int
+ MovingEntity(PathPosition)
+ moveDownPath(): void
+ moveUpPath(): void
Use of observer pattern to track Moving entity position
Character
- hp: int
- attack: Attack
- defense: int
- weapon: Weapon
- armour: ProtectiveGear
- shield: ProtectiveGear
- helmet: ProtectiveGear
- gold: int
- exp: int
- buffed: boolean
+ Character(PathPosition):
+ setAttack(): void
+ setBuffAttack(int): void
+ hasShield(): boolean
+ getWeaponType(): String
+ getWeaponType(): String
+ battle(String): int
+ unBuffAttack(int): void
+ calculateArmourDefence(int): int
+ damage(int): void
+ move(): void
Enemy
+ name: String
+movementCounter: int
+movementLimit: int
+hp: int
+gold: int
+exp: int
+supportRange: int
+attackRange: int
+tranceCounter: int
+attack: Attack
+ Enemy(PathPosition):
+ move(): void
+ getHp(): int
+ getName(): String
+ takeDamage(int): void
+ getGold(): int
+ getExp(): int
+ getSupportRange(): int
+ getAttackRange(): int
+ setTranceCounter(): void
+ updateTranceCounter(): void
+ attack(boolean): int
Attack
+ Attack()
+ attackExecute(): int
AttackDecorator
+ attack: Attack
+ AttackDecorator(Attack)
+ attackExecute(): int
BasicAttack
+ damage: int
+ attackExecute(): int
SlugAttack
+ damage: int
+ attackExecute(): int
Use of Decorator and Strategy Pattern
AllySoldier
+ AllySoldier(PathPosition)
+ move(): void
BattleSimulator
- healPoint: int
- duration: int
- criticalChance: double
- zombieChance: double
+ runBattle(List<Enemy>,List<AllySoldier>,Character):ArrayList<Enemy>
+ setState(Character): void
+ setMonsterMove(List<Enemy>, Character): void
+ enemyEffect(List<Enemy>, List<AllySoldier>, Character): void
+ enemyAttackCharacter(List<Enemy>, Character): void
+ enemyAttackAlly(List<Enemy>, List<AllySoldier>): void
+ allyAttackEnemy(List<Enemy>, List<AllySoldier>): void
+ characterAttackEnemy(List<Enemy>, Character): void
+ tranceAttackEnemy(List<Enemy>): void
+ heal Enemies(List<Enemy>): void
+ biteAlly(List<AllySoldier>, Enemy): void
+ stun(Character): void
+ criticalBite(Character, Enemy): boolean
+ DistanceCalculate(Character, Enemy): double
+ DistanceCalculate(Enemy, Enemy): double
+ DistanceCalculate(AllySoldier, Enemy): double
Zombie
+Zombie(PathPosition):
+move(): void
+attack(boolean): int
Vampire
-criticalBite: boolean
-criticalBiteStrikes: int
+Vampire(PathPosition):
+move(): void
+attack(boolean): int
Slug
+Slug(PathPosition):
+move(): void
+attack(boolean): int
SwordWeaponAttack
- additionalDamage: int
+ SwordWeaponAttack(Attack):
+ attackExecute(): int
VampireAttack
- additionalDamage: int
+ VampireAttack(Attack):
+ attackExecute(): int
StakeWeaponAttack
- additionalDamage: int
+ StakeWeaponAttack(Attack):
+ attackExecute(): int
VampireBiteAttack
MAX_ADDITIONAL_DAMAGE: int
MIN_ADDITIONAL_DAMAGE: int
- additionalDamage: int
+ VampireBiteAttack(Attack):
+ attackExecute(): int
BuffAttack
- multiplier: int
+ StakeWeaponAttack(Attack, int)
+ attackExecute(): int
CriticalWeaponAttack
- additionalDamage: int
+ CriticalWeaponAttack(Attack):
+ attackExecute(): int
ProtectiveGearAttack
- attackDecrease: int
+ ProtectiveGearAttack(Attack, int)
+ attackExecute(): int
ElanAttack
- additionalDamage: int
+ ElanAttack(Attack)
+ attackExecute(): int
AnduriIAttack
- additionalDamage: int
+ AnduriIAttack(Attack)
+ attackExecute(): int
DoggieAttack
- additionalDamage: int
+ DoggieAttack(Attack)
+ attackExecute(): int
BossBuffAttack
- multiplier: int
+ BossBuffAttack(Attack)
+ attackExecute(): int
Use of Factory Pattern and Observer
BossFactory
- doggieCycle: int
- elanCycle: int
- elanExp: int
+ BossFactory(): boolean
+ checkSpawn(int, int): boolean
+ checkDoggieSpawn(int): boolean
+ checkElanSpawn(int): boolean
+ checkCovidSpawn(int): boolean
+ spawnBoss(int, int, PathPosition): Enemy
Covid
+Covid(PathPosition):
+move(): void
+attack(boolean): int
Doggie
+Doggie(PathPosition):
+move(): void
+attack(boolean): int
ElanMuske
+ElanMuske(PathPosition):
+move(): void
+attack(boolean): int
Goals
+ quantity: int
+ Goals(int)
+ getQuantity(): int
+ goalCheck(int, int, int, boolean): void
+ getGoal(): BooleanProperty
+ setGoal(boolean): void
+ getGoalStatus(): boolean
ExpGoal
- goalStatus: BooleanProperty
+ ExpGoal(int)
GoldGoal
- goalStatus: BooleanProperty
+ GoldGoal(int)
CycleGoal
- goalStatus: BooleanProperty
+ CycleGoal(int)
BossGoal
- goalStatus: BooleanProperty
+ BossGoal(int)
ComplexGoals
+ goalList: ArrayList<Goals>
+ ComplexGoals(int, ArrayList<Goals>)
Composite Pattern
ComplexGoalLogicalOr
- goalStatus: BooleanProperty
+ ComplexGoalLogicalOr(int, ArrayList<Goals>)
ComplexGoalLogicalAnd
- goalStatus: BooleanProperty
+ ComplexGoalLogicalAnd(int, ArrayList<Goals>)
Use of Strategy Pattern
StandardShop
+ initialiseShopItems(): HashMap<String, Integer>
+ buyFromShop(LoopManiaWorldController, HashMap<String, Integer>, String): HashMap<String, Integer>
+ exitShop(HashMap<String, Integer>): HashMap<String, Integer>
+ additemIntoWorld(LoopManiaWorldController, Items): void
SurvivalShop
+ initialiseShopItems(): HashMap<String, Integer>
+ buyFromShop(LoopManiaWorldController, HashMap<String, Integer>, String): HashMap<String, Integer>
+ exitShop(HashMap<String, Integer>): HashMap<String, Integer>
+ additemIntoWorld(LoopManiaWorldController, Items): void
BerserkerShop
+ initialiseShopItems(): HashMap<String, Integer>
+ buyFromShop(LoopManiaWorldController, HashMap<String, Integer>, String): HashMap<String, Integer>
+ exitShop(HashMap<String, Integer>): HashMap<String, Integer>
+ additemIntoWorld(LoopManiaWorldController, Items): void
Shop
- itemsAvailable: HashMap<String, Integer>
- unequippedInventoryItems: List<Entity>
- strategy: ShopStrategy
- shopActive: BooleanProperty
- shopCounter: int
- shopCycle: int
+ Shop(ShopStrategy, List<Entity>)
+ buyFromShop(LoopManiaWorldController, String): void
+ buyFromCharacter(LoopManiaWorld): void
+ shopExit(): void
+ checkShopActivate(int, int, int): void
+ getShopActive(): BooleanProperty
+ setShopActive(boolean) void
+ getStrategy(): ShopStrategy
ShopParser
+ createShop(int, List<Entity>): Shop
ShopStrategy
+ initialiseShopItems(): HashMap<String, Integer>
+ buyFromShop(LoopManiaWorldController, Hashmap<String, Integer>, String): HashMap<String, Integer>
+ exitShop(HashMap<String, Integer>): HashMap<String, Integer>
1
0..*
1..*