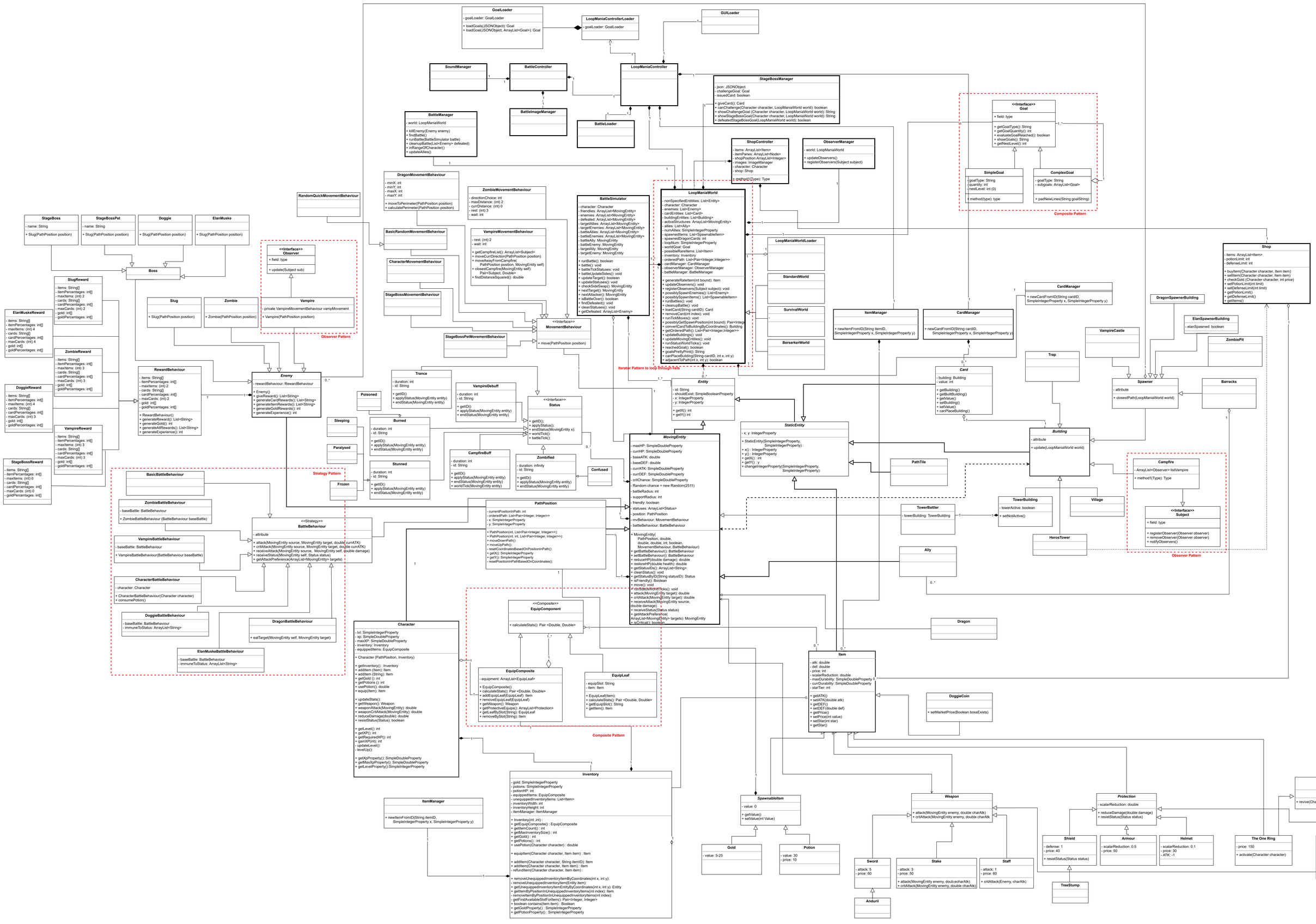
GoalLoader
- goalLoader: GoalLoader
+ loadGoals(JSONObject): Goal
+ loadGoal(JSONObject, ArrayList<Goal>): Goal
LoopManiaControllerLoader
- goalLoader: GoalLoader
GUILoader
SoundManager
BattleController
LoopManiaController
StageBossManager
- json: JSONObject
- challengeGoal: Goal
- issuedCard: boolean
+ giveCard(): Card
+ canChallenge(Character character, LoopManiaWorld world): boolean
+ showChallengeGoal (Character character, LoopManiaWorld world): String
+ showStageBossGoal(Character character, LoopManiaWorld world): String
+ defeatedStageBossGoal(LoopManiaWorld world): boolean
BattleImageManager
BattleLoader
BattleManager
- world: LoopManiaWorld
+ killEnemy(Enemy enemy)
+ findBattle()
+ runBattle(BattleSimulator battle)
+ cleanupBattle(List<Enemy> defeated)
+ inRangeOfCharacter()
+ updateAllies()
<<Interface>>
Goal
+ field: type
+ getGoalType(): String
+ getGoalQuantity(): int
+ evaluateGoalReached(): boolean
+ showGoals(): String
+ getNestLevel(): int
SimpleGoal
- goalType: String
- quantity: int
- nestLevel: int (0)
+ method(type): type
ComplexGoal
- goalType: String
- subgoals: ArrayList<Goal>
+ padNewLines(String goalString)
Composite Pattern
ShopController
- items: ArrayList<Item>
- itemPanes: ArrayList<Node>
- shopPosition:ArrayList<Integer>
- images: ImageManager
- character: Character
- shop: Shop
+ method1(Type): Type
ObserverManager
- world: LoopManiaWorld
+ updateObservers()
+ registerObservers(Subject subject)
DragonMovementBehaviour
- minX: int
- minY: int
- maxX: int
- maxY: int
+ moveToPerimeter(PathPosition position)
+ calculatePerimeter(PathPosition position)
ZombieMovementBehaviour
- directionChoice: int
- currDistance: (int) 2
- curr Distance: (int) 0
- rest: (int) 3
- wait: int
BattleSimulator
- character: Character
- friendlies: ArrayList<MovingEntity>
- enemies: ArrayList<MovingEntity>
- defeated: ArrayList<MovingEntity>
- targetAllies: ArrayList<MovingEntity>
- targetEnemies: ArrayList<MovingEntity>
- battleAllies: ArrayList<MovingEntity>
- battleEnemies: ArrayList<MovingEntity>
- battleAlly: MovingEntity
- battleEnemy: MovingEntity
- targetAlly: MovingEntity
- targetEnemy: MovingEntity
+ runBattle(): boolean
+ battle() : void
+ battleTickStatuses: void
+ battleUpdateSides(): void
+ updateTarget(): boolean
+ updateStatuses(): void
+ checkSideSwap(): MovingEntity
+ nextTarget(): MovingEntity
+ nextAttacker(): MovingEntity
+ isBattleOver(): boolean
+ findDefeated(): void
+ cleanStatuses(): void
+ getDefeated: ArrayList<Enemy>
LoopManiaWorld
- nonSpecifiedEntities: List<Entity>
- character: Character
- enemies: List<Enemy>
- cardEntities: List<Card>
- buildingEntities: List<Building>
- activeStructures: ArrayLists<MovingEntity>
- allies: List<Ally>
- numAllies: SimpleIntegerProperty
- spawnedItems: List<SpawnableItem>
- spawnedDragonCards: int
- loopNum: SimpleIntegerProperty
- worldGoal: Goal
- possibleRareItems: List<Item>
- inventory: Inventory
- orderedPath: List<Pair<Integer,Integer>>
- cardManager: CardManager
- observerManager: ObserverManager
- battleManager: BattleManager
+ generateRateItem(int bound): Item
+ updateObservers(): void
+ registerObservers(Subject subject): void
+ possiblySpawnEnemies(): List<Enemy>
+ possiblySpawnItems(): List<SpawnableItem>
+ runBattles(): void
+ cleanupBattle(): void
+ loadCard(String cardID): Card
+ removeCard(int index): void
+ runTickMoves(): void
+ possiblyGetSpawnPosition(int bound): Pair<Integ
+ convertCardToBuildingByCoordinates(): Building
+ getOrderedPath(): List<Pair<Integer,Integer>>
+ updateBuildings(): void
+ updateMovingEntities(): void
+ runStatusWorldTicks(): void
+ reachedGoal(): boolean
+ goalsPrettyPrint(): String
+ canPlaceBuilding(String cardID, int x, int y)
+ adjacentToPath(int x, int y): boolean
Iterater Pattern to loop through lists
LoopManiaWorldLoader
StandardWorld
SurvivalWorld
BerserkWorld
Shop
- items: ArrayList<Item>
- potionLimit: int
- defenseLimit: int
+ buyItem(Character character, Item item)
+ sellItem(Character character, Item item)
+ checkGold (Character character, int price)
+ setPotionLimit(int limit)
+ setDefenseLimit(int limit)
+ getPotionLimit()
+ getDefenseLimit()
+ getItems()
RandomQuickMovementBehaviour
StageBoss
- name: String
+ Slug(PathPosition position)
StageBossPet
- name: String
+ Slug(PathPosition position)
Doggie
+ Slug(PathPosition position)
ElanMuske
+ Slug(PathPosition position)
Boss
BasicRandomMovementBehaviour
VampireMovementBehaviour
- rest: (int) 2
- wait: int
+ getCampfireList(): ArrayList<Subject>
+ moveCurrDirection(PathPosition position)
+ moveAwayFromCampfire(
PathPosition position, MovingEntity self)
+ closestCampfire(MovingEntity self):
Pair<Subject, Double>
+ findDistanceSquared(): double
CharacterMovementBehaviour
StageBossMovementBehaviour
StageBossPetMovementBehaviour
<<Interface>>
Observer
+ field: type
+ update(Subject sub)
Vampire
- private VampireMovementBehaviour vampMovement
+ Vampire(PathPosition position)
Observer Pattern
Slug
+ Slug(PathPosition position)
Zombie
+ Zombie(PathPosition position)
<<Interface>>
MovementBehaviour
+ move(PathPosition position)
SlugReward
- items: String[]
- itemPercentages: int[]
- maxItems: (int) 2
- cards: String[]
- cardPercentages: int[]
- maxCards: (int) 2
- gold: int[]
- goldPercentages: int[]
ElanMuskeReward
- items: String[]
- itemPercentages: int[]
- maxItems: (int) 4
- cards: String[]
- cardPercentages: int[]
- maxCards: (int) 4
- gold: int[]
- goldPercentages: int[]
ZombieReward
- items: String[]
- itemPercentages: int[]
- maxItems: (int) 3
- cards: String[]
- cardPercentages: int[]
- maxCards: (int) 3
- gold: int[]
- goldPercentages: int[]
DoggieReward
- items: String[]
- itemPercentages: int[]
- maxItems: (int) 4
- cards: String[]
- cardPercentages: int[]
- maxCards: (int) 3
- gold: int[]
- goldPercentages: int[]
VampireReward
- items: String[]
- itemPercentages: int[]
- maxItems: (int) 3
- cards: String[]
- cardPercentages: int[]
- maxCards: (int) 3
- gold: int[]
- goldPercentages: int[]
StageBossReward
- items: String[]
- itemPercentages: int[]
- maxItems: (int) 0
- cards: String[]
- cardPercentages: int[]
- maxCards: (int) 0
- goldPercentages: int[]
RewardBehaviour
- items: String[]
- itemPercentages: int[]
- maxItems: (int) 2
- cards: String[]
- cardPercentages: int[]
- maxCards: (int) 2
- gold: int[]
- goldPercentages: int[]
+ RewardBehaviour()
+ generateReward(): List<String>
+ generateGold(): int
+ generateAllRewards(): List<String>
+ generateExperience(): int
Enemy
- rewardBehaviour: RewardBehaviour
+ Enemy()
+ giveReward(): List<String>
+ generateCardRewards(): List<String>
+ generateItemRewards(): List<String>
+ generateGoldRewards(): int
+ generateExperience(): int
Trance
- duration: int
- id: String
+ getID()
+ applyStatus(MovingEntity entity)
+ endStatus(MovingEntity entity)
VampireDebuff
- duration: int
- id: String
+ getID()
+ applyStatus(MovingEntity entity)
+ endStatus(MovingEntity entity)
Poisoned
Sleeping
Paralysed
Frozen
Burned
- duration: int
- id: String
+ getID()
+ applyStatus(MovingEntity entity)
+ endStatus(MovingEntity entity)
Stunned
- duration: int
- id: String
+ getID()
+ applyStatus(MovingEntity entity)
+ endStatus(MovingEntity entity)
<<Interface>>
Status
+ getID();
+ applyStatus();
+ endStatus(MovingEntity x);
+ worldTick();
+ battleTick();
CampfireBuff
- duration: int
- id: String
+ getID()
+ applyStatus(MovingEntity entity)
+ endStatus(MovingEntity entity)
+ worldTick(MovingEntity entity)
Zombified
- duration: infinity
- id: String
+ getID()
+ applyStatus(MovingEntity entity)
+ endStatus(MovingEntity entity)
Confused
Entity
- id: String
- shouldExist: SimpleBooleanProperty
- x: IntegerProperty
- y: IntegerProperty
+ getX(): int
+ getY() int
MovingEntity
- maxHP: SimpleDoubleProperty
- currHP: SimpleDoubleProperty
- baseATK: double
- baseDEF: double
- currATK: SimpleDoubleProperty
- currDEF: SimpleDoubleProperty
- critChance: SimpleDoubleProperty
- Random chance = new Random(2511)
- battleRadius: int
- supportRadius: int
- friendly: boolean
- statuses: ArrayList<Status>
- position: PathPosition
- mvBehaviour: MovementBehaviour
- battleBehaviour: BattleBehaviour
+ MovingEntity(
PathPosition, double,
double, double, int, boolean,
MovementBehaviour, BattleBehaviour)
+ getBattleBehaviour(): BattleBehaviour
+ setBattleBehaviour(): BattleBehaviour
+ reduceHP(double damage): double
+ restoreHP(double health): double
+ getStatusIDs(): ArrayList<String>
+ cleanStatus(): void
+ getStatusByID(String statusID): Status
+ isFriendly(): Boolean
+ move(): void
+ runStatusWorldTicks(): void
+ attack(MovingEntity target): double
+ critAttack(MovingEntity target): double
+ receiveAttack(MovingEntity source,
double damage)
+ receiveStatus(Status status)
+ getAttackPreference(
ArrayList<MovingEntity> targets): MovingEntity
+ isCritical(): boolean
StaticEntity
- x, y: IntegerProperty
+ StaticEntity(SimpleIntegerProperty,
SimpleIntegerProperty) :
+ x() : IntegerProperty
+ y() : IntegerProperty
+ getX() : int
+ getY() : y
+ changeIntegerProperty(SimpleIntegerProperty,
SimpleIntegerProperty):
Card
- building: Building
- value: int
+ getBuilding()
+ getBuiltBuilding()
+ getValue()
+ setBuilding()
+ setValue()
+ canPlaceBuilding()
ItemManager
+ newItemFromID(String itemID,
SimpleIntegerProperty x, SimpleIntegerProperty y)
CardManager
+ newCardFromID(String cardID,
SimpleIntegerProperty x, SimpleIntegerProperty y)
CardManager
+ newCardFromID(String cardID,
SimpleIntegerProperty x, SimpleIntegerProperty y)
DragonSpawnerBuilding
ElanSpawnerBuilding
- elanSpawned: boolean
VampireCastle
ZombiePit
Trap
Spawner
- attribute
+ closestPath(LoopManiaWorld world)
Barracks
Building
- attribute
+ update(LoopManiaWorld world)
PathTile
Campfire
- ArrayList<Observer> listVampire
+ method1(Type): Type
<<Interface>>
Subject
+ field: type
+ registerObserver(Observer observer)
+ removeObserver(Observer observer)
+ notifyObservers()
Observer Pattern
TowerBattler
- towerBuilding: TowerBuilding
TowerBuilding
- towerActive: boolean
+ setNotActive()
Village
HerosTower
Ally
Strategy Pattern
BasicBattleBehaviour
ZombieBattleBehaviour
- baseBattle: BattleBehaviour
+ ZombieBattleBehaviour (BattleBehaviour baseBattle)
<<Strategy>>
BattleBehaviour
- attribute
+ attack(MovingEntity source, MovingEntity target, double currATK)
+ critAttack(MovingEntity source, MovingEntity target, double currATK)
+ receiveAttack(MovingEntity source, MovingEntity self, double damage)
+ receiveStatus(MovingEntity self, Status status)
+ getAttackPreference(ArrayList<MovingEntity> targets)
VampireBattleBehaviour
- baseBattle: BattleBehaviour
+ VampireBattleBehaviour(BattleBehaviour baseBattle)
CharacterBattleBehaviour
- character: Character
+ CharacterBattleBehaviour(Character character)
+ consumePotion()
DoggieBattleBehaviour
- baseBattle: BattleBehaviour
- immuneToStatus: ArrayList<String>
DragonBattleBehaviour
+ eatTarget(MovingEntity self, MovingEntity target)
ElanMuskeBattleBehaviour
- baseBattle: BattleBehaviour
- immuneToStatus: ArrayList<String>
PathPosition
- currentPositionInPath: int
- orderedPath: List<Pair<Integer, Integer>>
- x: SimpleIntegerProperty
- y: SimpleIntegerProperty
+ PathPosition(int, List<Pair<Integer, Integer>>):
+ PathPosition(int, int, List<Pair<Integer, Integer>>):
+ moveDownPath():
+ moveUpPath():
+ resetCoordinatesBaseOnPositionInPath():
+ getX(): SimpleIntegerProperty
+ getY() : SimpleIntegerProperty
+ resetPositionInPathBaseOnCoordinates():
<<Composite>>
EquipComponent
+ calculateStats(): Pair <Double, Double>
Character
- lvl: SimpleIntegerProperty
- xp: SimpleDoubleProperty
- maxXP: SimpleDoubleProperty
- inventory: Inventory
- equippedItems: EquipComposite
+ Character (PathPosition, Inventory)
+ getInventory() : Inventory
+ addItem (Item): Item
+ addItem (String): Item
+ getGold () : int
+ getPotions () : int
+ usePotion(): double
+ equipItem(): Item
+ updateStats():
+ getWeapon(): Weapon
+ weaponAttack(MovingEntity): double
+ weaponCritAttack(MovingEntity): double
+ reduceDamage(double): double
+ resistStatus(Status): boolean
+ getLevel(): int
+ getXP() : int
+ getRequiredXP(): int
+ gainXP(int): int
+ updateLevel():
- levelUp():
+ getXpProperty(): SimpleDoubleProperty
+ getMaxXpProperty(): SimpleDoubleProperty
+ getLevelProperty():SimpleIntegerProperty
EquipComposite
- equipment: ArrayList<EquipLeaf>
+ EquipComposite():
+ calculateStats() : Pair <Double, Double>
+ addEquipLeaf(EquipLeaf): Item
+ removeEquipLeaf(EquipLeaf)
+ getWeapon(): Weapon
+ getProtectiveEquips() : ArrayList<Protection>
+ getLeafBySlot(String): EquipLeaf
+ removeBySlot(String): Item
EquipLeaf
- equipSlot: String
- item: Item
+ EquipLeaf(Item):
+ calculateStats() : Pair <Double, Double>
+ getEquipSlot(): String
+ getItem(): Item
Composite Pattern
Item
- atk: double
- def: double
- price: int
- scalarReduction: double
- maxDurability: SimpleDoubleProperty
- currDurability: SimpleDoubleProperty
- starTier: int
+ getATK()
+ setATK(double atk)
+ getDEF()
+ setDEF(double def)
+ getPrice()
+ setPrice(int value)
+ setStar(int star)
+ getStar()
DoggieCoin
+ setMarketPrice(Boolean bossExists)
Dragon
Inventory
- gold: SimpleIntegerProperty
- potions: SimpleIntegerProperty
- potionHP: int
- equippedItems: EquipComposite
- unequippedInventoryItems: List<Item>
- inventoryWidth: int
- inventoryHeight: int
- itemManager: ItemManager
+ Inventory(int ,int) :
+ getEquipComposite() : EquipComposite
+ getItemCount() : int
+ getMaxInventorySize() : int
+ getGold() : int
+ getPotions() : int
+ usePotion(Character character) : double
+ equipItem(Character character, Item item) : Item
+ addItem(Character character, String itemID): Item
+ addItem(Character character, Item item) : Item
+ refundItem(Character character, Item item) :
+ removeUnequippedInventoryItemByCoordinates(int x, int y):
+ removeUnequippedInventoryItem(Entity item):
+ getUnequippedInventoryItemEntityByCoordinates(int x, int y): Entity
+ getItemByPositionInUnequippedInventoryItems(int index): Item
+ removeItemByPositionInUnequippedInventoryItems(int index):
+ getFirstAvailableSlotForItem():: Pair<Integer, Integer>
+ boolean contains(Item item) : Boolean
+ getGoldProperty() : SimpleIntegerProperty
+ getPotionProperty() : SimpleIntegerProperty
ItemManager
+ newItemFromID(String itemID,
SimpleIntegerProperty x, SimpleIntegerProperty y)
SpawnableItem
- value: 0
+ getValue()
+ setValue(int Value)
Gold
- value: 5-25
Potion
- value: 30
- price: 10
Weapon
+ attack(MovingEntity enemy, double charAtk)
+ critAttack(MovingEntity enemy, double charAtk
Sword
- attack: 5
- price: 60
Anduril
Stake
- attack: 3
- price: 50
+ attack(MovingEntity enemy, doub;echarAtk)
+ critAttack(MovingEntity enemy, double charAtk)
Staff
- attack: 1
- price: 60
+ critAttack(Enemy, charAtk)
Protection
- scalarReduction: double
+ reduceDamage(double damage)
+ resistStatus(Status status)
Shield
- defense: 1
- price: 40
+ resistStatus(Status status)
Armour
- scalarReduction: 0.5
- price: 50
Helmet
- scalarReduction: 0.1
- price: 30
- ATK: -1
TreeStump
<<Interface>>
Reviver
+ revive(Character character): booelean)
The One Ring
- price: 150
+ activate(Character character)
ConfusedItem
- worldRareItems: ArrayList<String>
- base: String
- extra: String
- anduril: Anduril
- treeStump: TreeStump
- theOneRing: TheOneRing
+ createCombination()