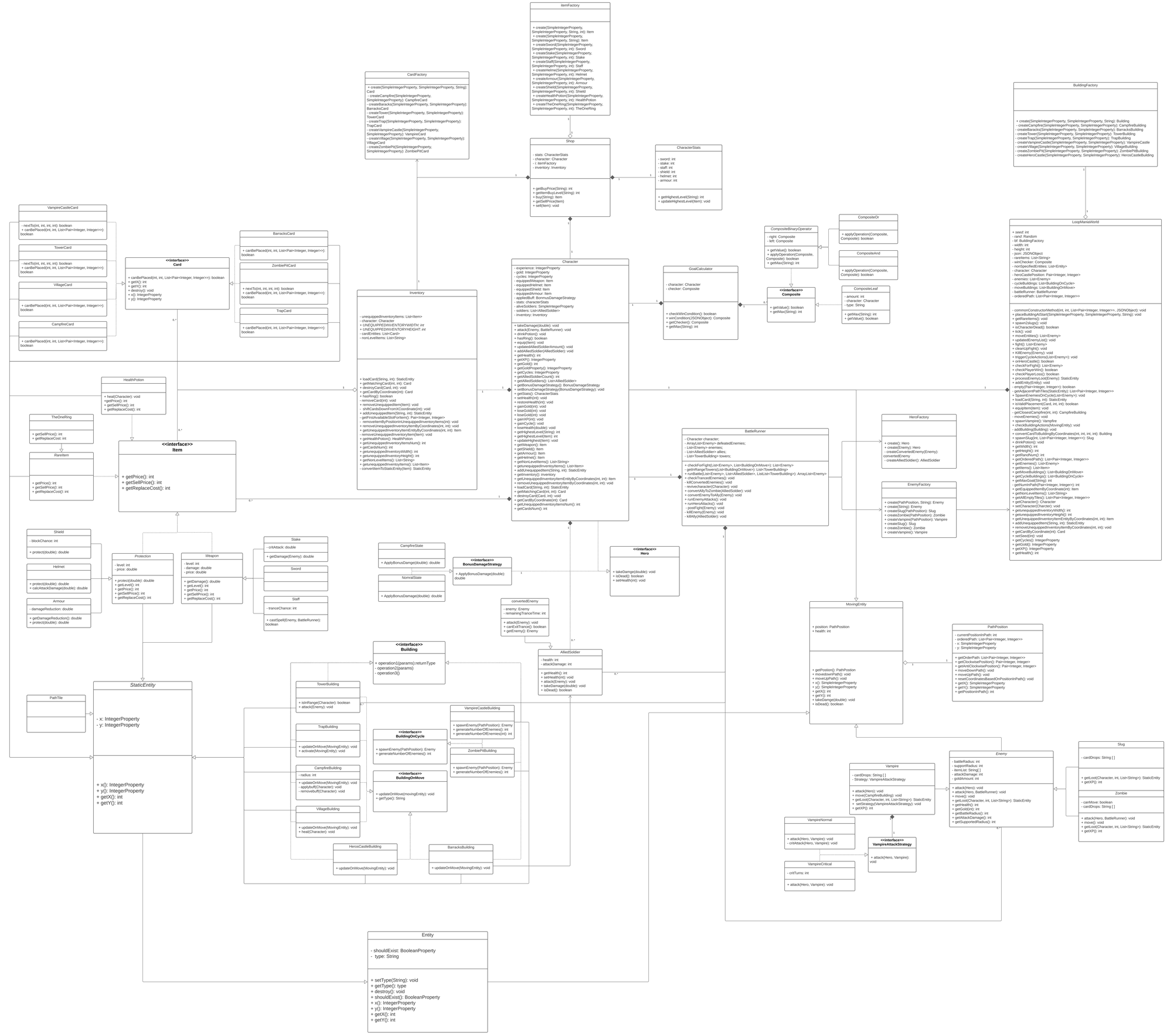
ItemFactory
+ create(SimpleIntegerProperty, SimpleIntegerProperty, String, int): Item
+ create(SimpleIntegerProperty, SimpleIntegerProperty, String): Item
+ createSword(SimpleIntegerProperty, SimpleIntegerProperty, int): Sword
+ createStake(SimpleIntegerProperty, SimpleIntegerProperty, int): Stake
+ createStaff(SimpleIntegerProperty, SimpleIntegerProperty, int): Staff
+ createHelmet(SimpleIntegerProperty, SimpleIntegerProperty, int): Helmet
+ createArmour(SimpleIntegerProperty, SimpleIntegerProperty, int): Armour
+ createShield(SimpleIntegerProperty, SimpleIntegerProperty, int): Shield
+ createHealthPotion(SimpleIntegerProperty, SimpleIntegerProperty, int): HealthPotion
+ createTheOneRing(SimpleIntegerProperty, SimpleIntegerProperty, int): TheOneRing
CardFactory
+ create(SimpleIntegerProperty, SimpleIntegerProperty, String): Card
- createCampfire(SimpleIntegerProperty, SimpleIntegerProperty): CampfireCard
- createBarracks(SimpleIntegerProperty, SimpleIntegerProperty): BarracksCard
- createTower(SimpleIntegerProperty, SimpleIntegerProperty): TowerCard
- createTrap(SimpleIntegerProperty, SimpleIntegerProperty): TrapCard
- createVampireCastle(SimpleIntegerProperty, SimpleIntegerProperty): VampireCard
- createVillage(SimpleIntegerProperty, SimpleIntegerProperty): VillageCard
- createZombiePit(SimpleIntegerProperty, SimpleIntegerProperty): ZombiePitCard
Shop
- stats: CharacterStats
- character: Character
- i: itemFactory
- inventory: Inventory
+ getBuyPrice(String): int
+ getItemBuyLevel(String): int
+ buy(String): Item
+ getSellPrice(Item)
+ sell(Item): void
CharacterStats
- sword: int
- stake: int
- staff: int
- shield: int
- helmet: int
- armour: int
+ getHighestLevel(String): int
+ updateHighestLevel(Item): void
BuildingFactory
+ create(SimpleIntegerProperty, SimpleIntegerProperty, String): Building
- createCampfire(SimpleIntegerProperty, SimpleIntegerProperty): CampfireBuilding
- createBarracks(SimpleIntegerProperty, SimpleIntegerProperty): BarracksBuilding
- createTower(SimpleIntegerProperty, SimpleIntegerProperty): TowerBuilding
- createTrap(SimpleIntegerProperty, SimpleIntegerProperty): TrapBuilding
- createVampireCastle(SimpleIntegerProperty, SimpleIntegerProperty): VampireCastle
- createVillage(SimpleIntegerProperty, SimpleIntegerProperty): VillageBuilding
- createZombiePit(SimpleIntegerProperty, SimpleIntegerProperty): ZombiePitBuilding
- createHeroCastle(SimpleIntegerProperty, SimpleIntegerProperty): HerosCastleBuilding
LoopManiaWorld
+ seed: int
- rand: Random
- bf: BuildingFactory
- width: int
- height: int
- json: JSONObject
- rareItems: List<String>
- winChecker: Composite
- nonSpecifiedEntities: List<Entity>
- character: Character
- heroCastlePosition: Pair<Integer, Integer>
- enemies: List<Enemy>
- cycleBuildings: List<BuildingOnCycle>
- moveBuildings: List<BuildingOnMove>
- battleRunner: BattleRunner
- orderedPath: List<Pair<Integer, Integer>>
- commonConstructorMethod(int, int, List<Pair<Integer, Integer>>, JSONObject): void
+ placeBuildingAtStart(SimpleIntegerProperty, SimpleIntegerProperty, String): void
+ getRareItems(): void
+ spawn2slugs(): void
+ isCharacterDead(): boolean
+ tick(): void
+ moveEntities(): List<Enemy>
+ updateEnemyList(): void
+ fight(): List<Enemy>
+ clearUpFight(): void
+ KillEnemy(Enemy): void
+ triggerCycleActions(List<Enemy>): void
+ onHeroCastle(): boolean
+ checkForFight(): List<Enemy>
+ checkPlayerWin(): boolean
+ checkPlayerLose(): boolean
+ processEnemyLoot(Enemy): StaticEntity
+ addEntity(Entity): void
- empty(Pair<Integer, Integer>): boolean
- getAdjacentPathTiles(StaticEntity): List<Pair<Integer, Integer>>
+ SpawnEnemiesOnCycle(List<Enemy>): void
+ loadCard(String, int): StaticEntity
+ isValidPlacement(Card, int, int): boolean
+ equipItem(Item): void
- getClosestCampfire(int, int): CampfireBuilding
- moveEnemies(): void
+ spawnVampire(): Vampire
+ checkBuildingActions(MovingEntity): void
- addBuilding(Building): void
+ convertCardToBuildingByCoordinates(int, int, int, int): Building
+ spawnSlug(Int, List<Pair<Integer, Integer>>): Slug
+ drinkPotion(): void
+ getWidth(): int
+ getHeight(): int
+ getRandNum(): int
+ getOrderedPath(): List<Pair<Integer, Integer>>
+ getEnemies(): List<Enemy>
+ getItems(): List<Item>
+ getMoveBuildings(): List<BuildingOnMove>
+ getCycleBuildings(): List<BuildingOnCycle>
+ getMaxGoal(String): int
- getNumInPath(Pair<Integer, Integer>): int
+ getEquippedItemByCoordinate(int): Item
+ getNonLevelItems(): List<String>
+ getAllEmptyTiles(): List<Pair<Integer, Integer>>
+ getCharacter(): Character
+ setCharacter(Character): void
+ getunequippedinventoryWidth(): int
+ getunequippedInventoryHeight(): int
+ getUnequippedInventoryItemEntityByCoordinates(int, int): Item
+ addUnequippedItem(String, int): StaticEntity
+ removeUnequippedInventoryItemByCoordinates(int, int): void
+ getCardByCoordinate(int): Card
+ setSeed(int): void
+ getCycles(): IntegerProperty
+ getGold(): IntegerProperty
+ getXP(): IntegerProperty
+ getHealth(): int
VampireCastleCard
- nextTo(int, int, int, int): boolean
+ canBePlaced(int, int, List<Pair<Integer, Integer>>): boolean
TowerCard
- nextTo(int, int, int, int): boolean
+ canBePlaced(int, int, List<Pair<Integer, Integer>>): boolean
VillageCard
+ canBePlaced(int, int, List<Pair<Integer, Integer>>): boolean
CampfireCard
+ canBePlaced(int, int, List<Pair<Integer, Integer>>): boolean
<<interface>>
Card
+ canBePlaced(int, int, List<Pair<Integer, Integer>>): boolean
+ getX(): int
+ getY(): int
+ destroy(): void
+ x(): IntegerProperty
+ y(): IntegerProperty
BarracksCard
+ canBePlaced(int, int, List<Pair<Integer, Integer>>): boolean
ZombiePitCard
- nextTo(int, int, int, int): boolean
+ canBePlaced(int, int, List<Pair<Integer, Integer>>): boolean
TrapCard
+ canBePlaced(int, int, List<Pair<Integer, Integer>>): boolean
Inventory
- unequippedInventoryItems: List<Item>
- character: Character
+ UNEQUIPPEDINVENTORYWIDTH: int
+ UNEQUIPPEDINVENTORYHEIGHT: int
- cardEntities: List<Card>
- nonLevelItems: List<String>
+ loadCard(String, int): StaticEntity
+ getMatchingCard(int, int): Card
+ destroyCard(Card, int): void
+ getCardByCoordinate(int): Card
+ hasRing(): boolean
- removeCard(int): void
+ removeUnequippedItem(Item): void
- shiftCardsDownFromXCoordinate(int): void
+ addUnequippedItem(String, int): StaticEntity
- getFirstAvailableSlotForItem(): Pair<Integer, Integer>
- removeItemByPositionInUnequippedInventoryItems(int): void
+ removeUnequippedInventoryItemByCoordinates(int, int): void
+ getunequippedInventoryItemEntityByCoordinates(int, int): Item
- removeUnequippedInventoryItem(Item): void
+ getHealthPotion(): HealthPotion
+ getUnequippedInventoryItemsNum(): int
+ getCardsNum(): int
+ getunequippedInventoryWidth(): int
+ getunequippedInventoryHeight(): int
+ getNonLevelItems(): List<String>
+ getunequippedInventoryItems(): List<Item>
- convertItemToStaticEntity(Item): StaticEntity
Character
- experience: IntegerProperty
- gold: IntegerProperty
- cycles: IntegerProperty
- equippedWeapon: Item
- equippedHelmet: Item
- equippedShield: Item
- equippedArmour: Item
- appliedBuff: BonnusDamageStrategy
- stats: characterStats
- aliveSoldiers: SimpleIntegerProperty
- soldiers: List<AlliedSoldier>
- inventory: Inventory
+ takeDamage(double): void
+ attack(Enemy, BattleRunner): void
+ drinkPotion(): void
+ hasRing(): boolean
+ equip(Item): void
+ updatedAlliedSoldierAmount(): void
+ addAlliedSoldier(AlliedSoldier): void
+ getHealth(): int
+ getXP(): IntegerProperty
+ getGold(): int
+ getGoldProperty(): IntegerProperty
+ getCycles: IntegerProperty
+ getAlliedSoldierCount(): int
+ getAlliedSoldiers(): List<AlliedSoldier>
+ getBonusDamageStrategy(): BonusDamageStrategy
+ setBonusDamageStrategy(BonusDamageStrategy): void
+ getStats(): CharacterStats
+ setHealth(int): void
+ restoreHealth(int): void
+ gainGold(int): void
+ loseGold(int): void
+ loseGold(int): void
+ gainXP(int): void
+ gainCycle(): void
+ loseHealth(double): void
+ getHighestLevel(String): int
+ getHighestLevel(Item): int
+ updateHighest(Item): void
+ getWeapon(): Item
+ getShield(): Item
+ getArmour(): Item
+ getHelmet(): Item
GoalCalculator
- character: Character
- checker: Composite
+ checkWinCondition(): boolean
+ winCondition(JSONObject): Composite
+ getChecker(): Composite
+ getMax(String): int
CompositeBinaryOperator
- right: Composite
- left: Composite
+ getValue(): boolean
+ applyOperation(Composite, Composite): boolean
+ getMax(String): int
CompositeOr
+ applyOperation(Composite, Composite): boolean
CompositeAnd
+ applyOperation(Composite, Composite): boolean
<<interface>>
Composite
+ getValue(): boolean
+ getMax(String): int
CompositeLeaf
- amount: int
- character: Character
- type: String
+ getMax(String): int
+ getValue(): boolean
HealthPotion
+ heal(Character): void
+ getPrice(): int
+ getSellPrice(): int
+ getReplaceCost(): int
TheOneRing
+ getSellPrice(): int
+ getReplaceCost: int
Rareitem
+ getPrice(): int
+ getSellPrice(): int
+ getReplaceCost(): int
<<interface>>
Item
+ getPrice(): int
+ getSellPrice(): int
+ getReplaceCost(): int
HeroFactory
+ create(): Hero
+ create(Enemy): Hero
- createConvertedEnemy(Enemy): convertedEnemy
- createAlliedSoldier(): AlliedSoldier
EnemyFactory
+ create(PathPosition, String): Enemy
+ create(String): Enemy
+ createSlug(PathPosition): Slug
+ createZombie(PathPosition): Zombie
+ createVampire(PathPosition): Vampire
+ createSlug(): Slug
+ createZombie(): Zombie
+ createVampire(): Vampire
BattleRunner
- Character character;
- ArrayList<Enemy> defeatedEnemies;
- List<Enemy> enemies;
- List<AlliedSoldier> allies;
- List<TowerBuilding> towers;
+ checkForFight(List<Enemy>, List<BuildingOnMove>): List<Enemy>
- getInRangeTowers(List<BuildingOnMove>): List<TowerBuilding>
+ runBattles(List<Enemy>, List<AlliedSoldier>, List<TowerBuilding>): ArrayList<Enemy>
+ checkTrancedEnemies(): void
- killConvertedEnemies(): void
- reviveCharacter(Character): void
+ convertAllyToZombie(AlliedSoldier): void
+ convertEnemyToAlly(Enemy): void
+ runEnemyAttacks(): void
+ runHeroAttacks(): void
- posFight(Enemy): void
- killEnemy(Enemy): void
- killAlly(AlliedSolder): void
Shield
- blockChance: int
+ protect(double): double
Helmet
+ protect(double): double
+ calcAttackDamage(double): double
Armour
- damageReduction: double
+ getDamageReduction(): double
+ protect(double): double
Protection
- level: int
- price: double
+ protect(double): double
+ getLevel(): int
+ getPrice(): int
+ getSellPrice(): int
+ getReplaceCost(): int
Weapon
- level: int
- damage: double
- price: double
+ getDamage(): double
+ getLevel(): int
+ getPrice(): int
+ getSellPrice(): int
+ getReplaceCost(): int
Stake
- critAttack: double
+ getDamage(Enemy): double
Sword
Staff
- tranceChance: int
+ castSpell(Enemy, BattleRunner): boolean
CampfireState
+ ApplyBonusDamage(double): double
NormalState
+ ApplyBonusDamage(double): double
<<interface>>
BonusDamageStrategy
+ ApplyBonusDamage(double): double
<<interface>>
Hero
+ takeDamage(double): void
+ isDead(): boolean
+ setHealth(int): void
convertedEnemy
- enemy: Enemy
- remainingTranceTime: int
+ attack(Enemy): void
+ canExitTrance(): boolean
+ getEnemy(): Enemy
AlliedSoldier
- health: int
- attackDamage: int
+ getHealth(): int
+ setHealth(int): void
+ attack(Enemy): void
+ takeDamage(double): void
+ isDead(): boolean
MovingEntity
+ position: PathPosition
+ health: int
+ getPostion(): PathPostion
+ moveDownPath(): void
+ moveUpPath(): void
+ x(): SimpleIntegerProperty
+ y(): SimpleIntegerProperty
+ getX(): int
+ getY(): int
+ takeDamage(double): void
+ isDead(): boolean
PathPosition
- currentPositioninPath: int
- orderedPath: List<Pair<Integer, Integer>>
- x: SimpleIntegerProperty
- y: SimpleIntegerProperty
+ getOrderPath: List<Pair<Integer, Integer>>
+ getClockwisePosition(): Pair<Integer, Integer>
+ getAntiClockwisePosition(): Pair<Integer, Integer>
+ moveDownPath(): void
+ moveUpPath(): void
+ resetCoordinatesBasedOnPossitionInPath(): void
+ getX(): SimpleIntegerProperty
+ getY(): SimpleIntegerProperty
+ getPositioninPath(): int
<<interface>>
Building
+ operation1(params):returnType
- operation2(params)
- operation3()
StaticEntity
- x: IntegerProperty
- y: IntegerProperty
+ x(): IntegerProperty
+ y(): IntegerProperty
+ getX(): int
+ getY(): int
PathTile
TowerBuilding
+ isInRange(Character): boolean
+ attack(Enemy): void
TrapBuilding
+ updateOnMove(MovingEntity): void
+ activate(MovingEntity): void
CampfireBuilding
- radius: int
+ updateOnMove(MovingEntity): void
- applybuff(Character): void
- removebuff(Character): void
VillageBuilding
+ updateOnMove(MovingEntity): void
+ heal(Character): void
VampireCastleBuilding
+ spawnEnemy(PathPosition): Enemy
+ generateNumberOfEnemies(): int
+ generateNumberOfEnemies(int): int
<<interface>>
BuildingOnCycle
+ spawnEnemy(PathPosition): Enemy
+ generateNumberOfEnemies(): int
ZombiePitBuilding
+ spawnEnemy(PathPosition): Enemy
+ generateNumberOfEnemies(): int
<<interface>>
BuildingOnMove
+ updateOnMove(movingEntity): void
+ getType(): String
HerosCastleBuilding
+ updateOnMove(MovingEntity): void
BarracksBuilding
+ updateOnMove(MovingEntity): void
Enemy
- battleRadius: int
- supportRadius: int
- itemList: String[ ]
- attackDamage: int
- goldAmount: int
+ attack(Hero): void
+ attack(Hero, BattleRunner): void
+ move(): void
+ getLoot(Character, int, List<String>): StaticEntity
+ getHealth(): int
+ getGold(int): int
+ getBattleRadius(): int
+ getAttackDamage(): int
+ getSupportedRadius(): int
Vampire
- cardDrops: String [ ]
- Strategy: VampireAttackStrategy
+ attack(Hero): void
+ move(CampfireBuilding): void
+ getLoot(Character, int, List<String>): StaticEntity
+ setStrategy(VampireAttackStrategy): void
+ getXP(): int
VampireNormal
+ attack(Hero, Vampire): void
- critAttack(Hero, Vampire): void
VampireCritical
- critTurns: int
+ attack(Hero, Vampire): void
<<interface>>
VampireAttackStrategy
+ attack(Hero, Vampire): void
Slug
- cardDrops: String [ ]
+ getLoot(Character, int, List<String>): StaticEntity
+ getXP(): int
Zombie
- cantMove: boolean
- cardDrops: String [ ]
+ attack(Hero, BattleRunner): void
+ move(): void
+ getLoot(Character, int, List<String>): StaticEntity
+ getXP(): int
Entity
- shouldExist: BooleanProperty
- type: String
+ setType(String): void
+ getType(): type
+ destroy(): void
+ shouldExist(): BooleanProperty
+ x(): IntegerProperty
+ y(): IntegerProperty
+ getX(): int
+ getY(): int