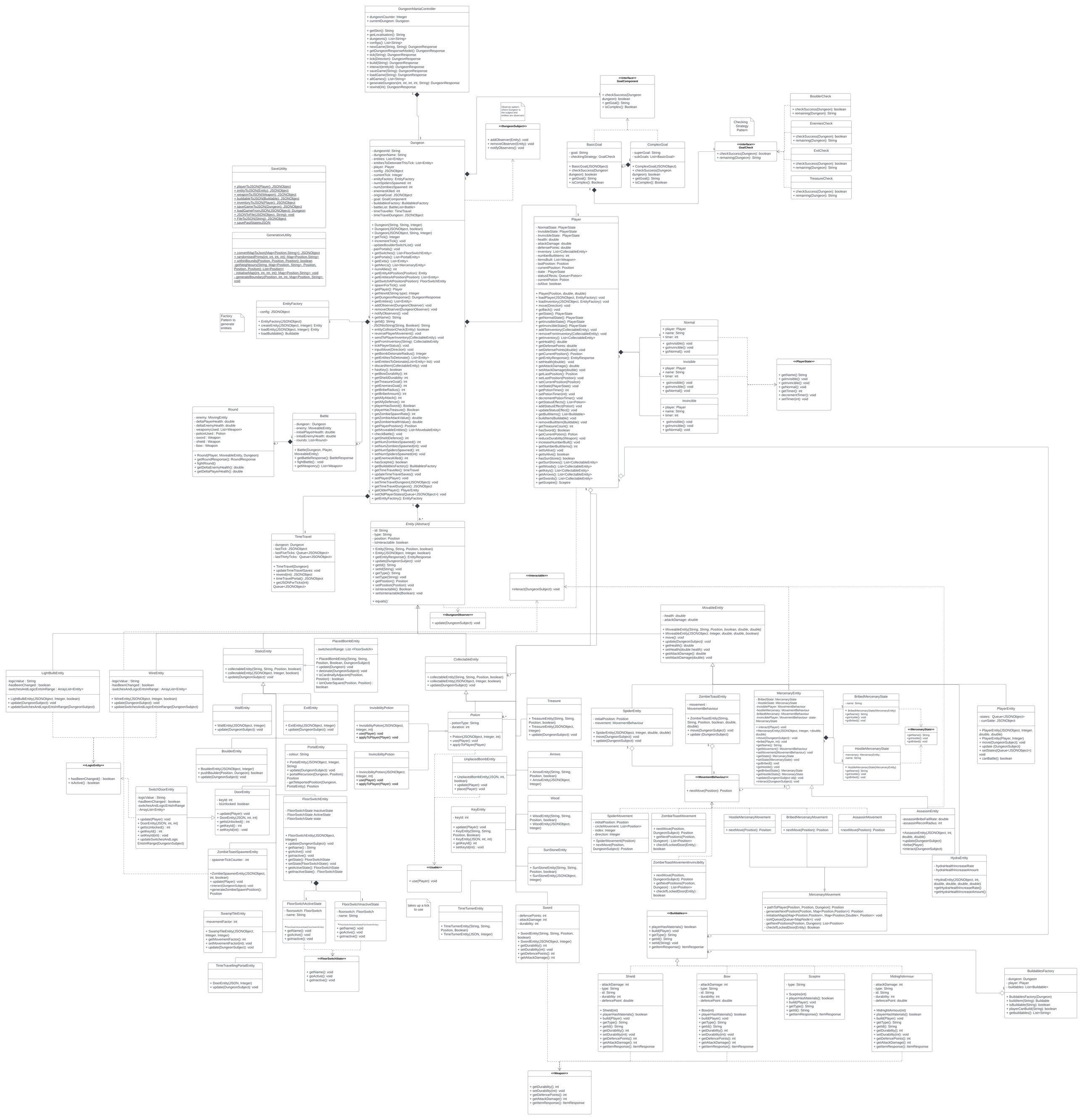
DungeonManiaController
+ dungeonCounter: Integer
+ currentDungeon: Dungeon
+ getSkin(): String
+ getLocalisation(): String
+ dungeons(): List<String>
+ configs(): List<String>
+ newGame(String, String): DungeonResponse
+ getDungeonResponseModel(): DungeonResponse
+ tick(String): DungeonResponse
+ tick(Direction): DungeonResponse
+ build(String): DungeonResponse
+ interact(entityId): DungeonResponse
+ saveGame(String): DungeonResponse
+ loadGame(String): DungeonResponse
+ allGames(): List<String>
+ generateDungeon(int, int, int, int, String): DungeonResponse
+ rewind(int): DungeonResponse
<<interface>>
GoalComponent
+ checkSuccess(Dungeon dungeon): boolean
+ getGoal(): String
+ isComplex(): Boolean
Observer pattern where Dungeon is the subject and entities are observers
<<DungeonSubject>>
+ addObserver(Entity): void
+ removeObserver(Entity): void
+ notifyObservers(): void
BasicGoal
- goal: String
- checkingStrategy: GoalCheck
+ BasicGoal(JSONObject)
+ checkSuccess(Dungeon dungeon): boolean
+ getGoal(): String
+ isComplex(): Boolean
ComplexGoal
- superGoal: String
- subGoals: List<BasicGoal>
+ ComplexGoal(JSONObject)
+ checkSuccess(Dungeon dungeon): boolean
+ getGoal(): String
+ isComplex(): Boolean
Checking Strategy Pattern
<<interface>>
GoalCheck
+ checkSuccess(Dungeon): boolean
+ remaining(Dungeon): String
BoulderCheck
+ checkSuccess(Dungeon): boolean
+ remaining(Dungeon): String
EnemiesCheck
+ checkSuccess(Dungeon): boolean
+ remaining(Dungeon): String
ExitCheck
+ checkSuccess(Dungeon): boolean
+ remaining(Dungeon): String
TreasureCheck
+ checkSuccess(Dungeon): boolean
+ remaining(Dungeon): String
SaveUtility
+ playerToJSON(Player): JSONObject
+ entityToJSON(Entity): JSONObject
+ weaponToJSON(Weapon): JSONObject
+ buildableToJSON(Buildable): JSONObject
+ inventoryToJSON(Player): JSONObject
+ saveGameToJSON(Dungeon): JSONObject
+ loadGameFromJSON(JSONObject): Dungeon
+ JSONToFile(JSONObject, String): void
+ FileToJSON(String): JSONObject
+ savePastStatetoJSON
GenerationUtility
+ convertMapToJson(Map<Position,String>): JSONObject
+ randomisedPrims(int, int, int, int): Map<Position,String>
+ withinBounds(Position, Position, Position): boolean
-getNeighbours(String, Map<Position, String>, Position, Position, Position): List<Position>
- initialiseMap(int, int, int, int): Map<Position,String>: void
- generateBoundary(Position, int, int, Map<Position, String>: void
Dungeon
- dungeonId: String
- dungeonName: String
- entities: List<Entity>
- entitiesToDetonateThisTick: List<Entity>
- player: Player
- config: JSONObject
- currentTick: Integer
- entityFactory: EntityFactory
- numSpidersSpawned: int
- numZombiesSpawned: int
- enemiesKilled: int
- originalGoal: JSONObject
- goal: GoalComponent
- buildablesFactory: BuildablesFactory
- battleList: BattleList<Battle>
- timeTraveller: TimeTravel
- timeTravelDungeon: JSONObject
+ Dungeon(String, String, Integer)
+ Dungeon(JSONObject, boolean)
+ Dungeon(JSONObject, String, Integer)
+ getTick(): Integer
+ incrementTick(): void
- updateBoulderSwitchList(): void
- pairPortals(): void
+ getSwitches(): List<FloorSwitchEntity>
+ getPortals(): List<PortalEntity>
+ getExits(): List<Entity>
+ getMercs(): List<MercenaryEntity>
+ numAllies(): int
+ getEntityAtPosition(Position): Entity
+ getEntitiesAtPosition(Position): List<Entity>
+ getSwitchAtPosition(Position): FloorSwitchEntity
+ spawnForTick(): void
+ getPlayer(): Player
+ getNewId(String type): Integer
+ getDungeonResponse(): DungeonResponse
+ getEntities(): List<Entity>
+ addObserver(DungeonObserver): void
+ removeObserver(DungeonObserver): void
+ notifyObservers(): void
+ getName(): String
+ getId(): String
- JSONtoString(String, Boolean): String
+ entityCollisionCheck(Entity): boolean
+ reversePlayerMovement(): void
+ sendToPlayerInventory(CollectableEntity): void
+ getFromInventory(String): CollectableEntity
+ tickPlayerStatus(): void
+ inputMove(Direction): void
+ getBombDetonateRadius(): Integer
+ getEntitiesToDetonate(): List<Entity>
+ setEntitiesToDetonate(List<Entity> list): void
+ discardItem(CollectableEntity): void
+ hasKey(): boolean
+ getBowDurability(): int
+ getShieldDurability: int
+ getTreasureGoal(): int
+ getEnemiesGoal(): int
+ getBribeRadius(): int
+ getBribeAmount(): int
+ getAllyAttack(): int
+ getAllyDefence(): int
+ playerHasSword(): Boolean
+ playerHasTreasure(): Boolean
+ getZombieSpawnRate(): int
+ getZombieAttackValue(): double
+ getZombieHealthValue(): double
+ getPlayerPosition(): Position
+ getMoveableEntities(): List<MovebaleEntity>
+ checkBattle(): void
+ getShieldDefence(): int
+ getNumZombiesSpawned(): int
+ setNumZombiesSpawned(int): void
+ getNumSpidersSpawned(): int
+ setNumSpidersSpawned(int): void
+ getEnemiesKilled(): int
+ hasSceptre(): boolean
+ getBuildablesFactory(): BuildablesFactory
+ getTimeTraveller(): timeTravel
+ updateTimeTravelSaves(): void
+ setPlayer(Player): void
+ setTimeTravelDungeon(JSONObject): void
+ getTimeTravelDungeon(): JSONObject
+ getOlderPlayer(): PlayerEntity
+ setOldPlayerStates(Queue<JSONObject>): void
+ getEntityFactory(): EntityFactory
EntityFactory
- config: JSONObject
+ EntityFactory(JSONObject)
+ createEntity(JSONObject, Integer): Entity
+ loadEntity(JSONObject, Integer): Entity
+ loadBuildable(): Buildable
Factory Pattern to generate entities
Player
- NormalState: PlayerState
- InvisibleState: PlayerState
- InvincibleState: PlayerState
- health: double
- attackDamage: double
- defensePoints: double
- inventory: List<CollectableEntity>
- numberBuiltItems: int
- itemsBuilt: List<Weapon>
- lastPosition: Position
- currentPosition: Position
- state : PlayerState
- statusEffects: Queue<Potion>
- currentPotion: Potion
- isAlive: boolean
+ Player(Position, double, double)
+ loadPlayer(JSONObject, EntityFactory): void
+ loadInventory(JSONObject, EntityFactory): void
+ move(Direction): void
+ goBack(): void
+ getState(): PlayerState
+ getNormalState(): PlayerState
+ getInvisibleState(): PlayerState
+ getInvincibleState(): PlayerState
+ addToInventory(CollectableEntity): void
+ removeFromInventory(CollectableEntity): void
+ getInventory(): List<CollectableEntity>
+ getHealth(): double
+ getDefensePoints: double
+ setDefensePoints(double): void
+ getCurrentPosition(): Position
+ getEntityResponse(): EntityResponse
+ setHealth(double): void
+ getAttackDamage(): double
+ setAttackDamage(double): void
+ getLastPosition(): Position
+ setLastPosition(Position): void
+ setCurrentPosition(Posiiton)
+ setState(PlayerState): void
+ getPotionTimer(): int
+ setPotionTimer(int): void
+ decrementPotionTimer(): void
+ getStatusEffects(): List<Potion>
+ addStatusEffect(Potion): void
+ updateStatusEffect(): void
+ getBuiltItems(): List<Buildable>
+ buildItem(Buildable): void
+ removeBuiltItem(Buildable): void
+ getTreasureCount(): int
+ hasSword(): Boolean
+ getCurrentPotion(): Potion
+ reduceDurability(Weapon): void
+ increaseNumberBuilt(): void
+ getNumberBuiltItems(): int
+ setIsAlive(): void
+ getIsAlive(): boolean
+ hasSunStone(): boolean
+ getSunStones(): List<CollectableEntity>
+ getWoods(): List<CollectableEntity>
+ getKeys(): List<CollectableEntity>
+ getArrows(): List<CollectableEntity>
+ getSwords(): List<CollectableEntity>
+ getSceptre(): Sceptre
Normal
+ player: Player
+ name: String
+ timer: int
+ goInvisible(): void
+ goInvincible(): void
+ goNormal(): void
Invisible
+ player: Player
+ name: String
+ timer: int
+ goInvisible(): void
+ goInvincible(): void
+ goNormal(): void
Invincible
+ player: Player
+ name: String
+ timer: int
+ goInvisible(): void
+ goInvincible(): void
+ goNormal(): void
<<PlayerState>>
+ getName() String
+ goInvisible(): void
+ goInvincible(): void
+ goNormal(): void
+ getTimer(): int
+ decrementTimer(): void
+ setTimer(int): void
Round
- enemy: MovingEntity
- deltaPlayerHealth: double
- deltaEnemyHealth: double
- weaponryUsed: List<Weapon>
- potionUsed : Potion
- sword : Weapon
- shield : Weapon
- bow : Weapon
+ Round(Player, MoveableEntity, Dungeon)
+ getRoundResponse(): RoundResponse
+ fightRound()
+ getDeltaEnemyHealth(): double
+ getDeltaPlayerHealth(): double
Battle
- dungeon : Dungeon
- enemy: MoveableEntity
- initialPlayerHealth: double
- initialEnemyHealth: double
- rounds: List<Round>
+ Battle(Dungeon, Player, MoveableEntity)
+ getBattleResponse(): BattleResponse
+ fightBattle() : void
+ getWeaponry(): List<Weapon>
TimeTravel
- dungeon: Dungeon
- lastTick: JSONObject
- lastFiveTicks: Queue<JSONObject>
- lastThirtyTicks: Queue<JSONObject>
+ TimeTravel(Dungeon)
+ updateTimeTravelSaves: void
+ rewind(int): JSONObject
+ timeTravelPortal(): JSONObject
+ getJSONForTicks(int): Queue<JSONObject>
Entity (Abstract)
- id: String
- type: String
- position: Position
- isInteractable: boolean
+ Entity(String, String, Position, boolean)
+ Entity(JSONObject, Integer, boolean)
+ getEntityResponse(): EntityResponse
+ update(DungeonSubject): void
+ getId(): String
+ setId(String): void
+ getType(): String
+ setType(String): void
+ getPosition(): Position
+ setPosition(Position): void
+ isInteractable(): Boolean
+ setIsInteractable(Boolean): void
+ equals():
<<Interactable>>
+interact(DungeonSubject): void
<<DungeonObserver>>
+ update(DungeonSubject): void
MovableEntity
- health: double
- attackDamage: double
+ MoveableEntity(String, String, Position, boolean, double, double)
+ MoveableEntity(JSONObject, Integer, double, double, boolean)
+ move(): void
+ update(DungeonSubject): void
+ getHealth(): double
+ setHealth(double health): void
+ getAttackDamage(): double
+ setAttackDamage(double): void
StaticEntity
+ collectableEntity(String, String, Position, boolean)
+ collectableEntity(JSONObject, Integer, boolean)
+ update(DungeonSubject): void
PlacedBombEntity
- switchesInRange: List <FloorSwitch>
+ PlacedBombEntity(String, String, Position, Boolean, DungeonSubject)
+ update(Dungeon): void
+ detonate(DungeonSubject): void
+ isCardinallyAdjacent(Position, Position) : boolean
+ isInOuterSquare(Position, Position) : boolean
CollectableEntity
+ collectableEntity(String, String, Position, boolean)
+ collectableEntity(JSONObject, Integer, boolean)
+ update(DungeonSubject): void
LightBulbEntity
-logicValue : String
-hasBeenChanged : boolean
-switchesAndLogicEntsInRange : ArrayList<Entity>
+ LightBulbEntity(JSONObject, Integer, boolean)
+ update(DungeonSubject): void
+ updateSwitchesAndLogicEntsInRange(DungeonSubject)
WireEntity
-logicValue : String
-hasBeenChanged : boolean
-switchesAndLogicEntsInRange : ArrayList<Entity>
+ WireEntity(JSONObject, Integer, boolean)
+ update(DungeonSubject): void
+ updateSwitchesAndLogicEntsInRange(DungeonSubject)
WallEntity
+ WallEntity(JSONObject, Integer)
+ update(DungeonSubject): void
ExitEntity
+ ExitEntity(JSONObject, Integer)
+ update(DungeonSubject): void
InvisibilityPotion
+ InvisibilityPotion(JSONObject, Integer, int)
+ use(Player): void
+ applyToPlayer(Player): void
Potion
- potionType: String
- duration: int
+ Potion(JSONObject, Integer, int)
+ use(Player): void
+ applyToPlayer(Player)
Treasure
+ TreasureEntity(String, String, Position, boolean)
+ TreasureEntity(JSONObject, Integer)
+ update(DungeonSubject): void
SpiderEntity
- initialPosition: Position
- movement: MovementBehaviour
+ SpiderEntity(JSONObject, Integer, double, double)
+ move(DungeonSubject): void
+ update(DungeonSubject): void
ZombieToastEntity
- movement : MovementBehaviour
+ ZombieToastEntity(String, String, Position, boolean, double, double)
+ move (DungeonSubject)
+ update (DungeonSubject)
MercenaryEntity
- BribedState: MercenaryState
- HostileState: MercenaryState
-invisiblePlayer: MovementBehaviour
-hostileMercenary: MovementBehaviour
-bribedMercenary: MovementBehaviour
-invinciblePlayer: MovementBehaviour- state: MercenaryState
+ interact(Player): void
+MercenaryEntity(JSONObject, Integer, +double, double)
+move(DungeonSubject) : void
+bribe(Player, int): void
+getName(): String
+getMovement(): MovementBehaviour
+setMovement(MovementBehaviour): void
+getState(): MercenaryState
+setState(MercenaryState): void
+goBribed(): void
+goHostile(): void
+getBribedState(): MercenaryState
+getHostileState(): MercenaryState
+update(DungeonSubject obj): void
+interact(DungeonSubject): void
BribedMercenaryState
- name: String
+ BribedMercenaryState(MercenaryEntity)
+getName(): String
+goHostile(): void
+goBribed(): void
<<MercenaryState>>
+getName(): String
+goHostile(): void
+goBribed(): void
HostileMercenaryState
-mercenary: MercenaryEntity;
-name: String
+ HostileMercenaryState(MercenaryEntity)
+getName(): String
+goHostile(): void
+goBribed(): void
PlayerEntity
-states: Queue<JSONObject>
- currSatte: JSONObject
+ PlayerEntity(JSONObject, Integer, double, double)
+ PlayerEntity(Player, Integer)
+ move(DungeonSubject): void
+ update (DungeonSubject)
+ setStates(Queue<JSONObject>): void
+ canBattle(): boolean
<<LogicEntity>>
+ hasBeenChanged() : boolean
+ isActive() : boolean
SwitchDoorEntity
-logicValue : String
-hasBeenChanged : boolean
-switchesAndLogicEntsInRange : ArrayList<Entity>
+ update(Player): void
+ DoorEntity(JSON, int, int)
+ getIsUnlocked() : int
+ getKeyId() : int
+ setKeyId(int) : void
+ updateSwitchesAndLogicEntsInRange(DungeonSubject)
BoulderEntity
+ BoulderEntity(JSONObject, Integer)
+ pushBoulder(Position, Dungeon): boolean
+ update(DungeonSubject): void
PortalEntity
- colour: String
+ PortalEntity(JSONObject, Integer, String)
+ update(DungeonSubject): void
+ portalRecursion(Dungeon, Position): Position
- getTeleportedPosition(Dungeon, PortalEntity): Position
DoorEntity
- keyId: int
- IsUnlocked: boolean
+ update(Player): void
+ DoorEntity(JSON, int, int)
+ getIsUnlocked() : int
+ getKeyId() : int
+ setKeyId(int) : void
FloorSwitchEntity
- FloorSwitchState InactiveState
- FloorSwitchState ActiveState
- FloorSwitchState state
+ FloorSwitchEntity(JSONObject, Integer)
+ update(DungeonSubject): void
+ getName() : String
+ goActive(): void
+ goInactive(): void
+ getState(): FloorSwitchState
+ setState(FloorSwitchState): void
+ getActiveState(): FloorSwitchState
+ getInactiveState() : FloorSwitchState
ZombieToastSpawnerEntity
- spawnerTickCounter : int
+ZombieSpawnerEntity(JSONObject, int, boolean)
+ update(Player): void
+interact(DungeonSubject): void
+generateZombieSpawnPosition(): Position
SwampTileEntity
-movementFactor: int
+ SwampTileEntity(JSONObject, Integer, Integer)
+ getMovementFactor(): int
+ setMovementFactor(int): void
+ update(DungeonSubject): void
TimeTravellingPortalEntity
+ DoorEntity(JSON, Integer)
+ update(DungeonSubject): void
FloorSwitchActiveState
- floorswitch: FloorSwitch
- name: String
+FloorSwitchActiveState(FloorSwitchEntity)
+ getName(): void
+ goActive(): void
+ goInactive(): void
FloorSwitchInactiveState
- floorswitch: FloorSwitch
- name: String
+FloorSwitchInactiveState(FloorSwitchEntity)
+ getName(): void
+ goActive(): void
+ goInactive(): void
<<FloorSwitchState>>
+ getName(): void
+ goActive(): void
+ goInactive(): void
UnplacedBombEntity
+ UnplacedBombEntity(JSON, int, boolean)
+ update(Player): void
+ place(Player): void
KeyEntity
- keyId: int
+ update(Player): void
+ KeyEntity(String, String, Position, Boolean)
+ KeyEntity(JSON, int, int)
+ getKeyId(): int
+ setKeyId(int) : void
<<Usable>>
+ use(Player): void
takes up a tick to use
TimeTurnerEntity
+ TimeTurnerEntity(String, String, Position, Boolean)
+ TimeTurnerEntity(JSON, Integer)
Arrows
+ ArrowEntity(String, String, Position, boolean)
+ ArrowEntity(JSONObject, Integer)
Wood
+ WoodEntity(String, String, Position, boolean)
+ WoodEntity(JSONObject, Integer)
SunStoneEntity
+ SunStoneEntity(String, String, Position, boolean)
+ SunStoneEntity(JSONObject, Integer)
Sword
- defencePoints: int
- attackDamage: int
- durability: int
+ SwordEntity(String, String, Position, boolean)
+ SwordEntity(JSONObject, Integer)
+ getDurability(): int
+ setDurability(int): void
+ getDefencePoints(): int
+ getAttackDamage(): int
<<MovementBehaviour>>
+ nextMove(Position): Position
SpiderMovement
- initialPosition: Position
- circleMovement: List<Position>
- index: Integer
- direction: Integer
+ SpiderMovement(Position)
+ nextMove(Position, DungeonSubject): Position
ZombieToastMovement
+ nextMove(Position, DungeonSubject): Position
+ getNextPositions(Position, Dungeon) : List<Position>
+ checkIfLockedDoor(Entity) : boolean
ZombieToastMovementInvincibility
+ nextMove(Position, DungeonSubject): Position
+ getNextPositions(Position, Dungeon) : List<Position>
+ checkIfLockedDoor(Entity) : boolean
HostileMercenaryMovement
+ nextMove(Position): Position
BribedMercenaryMovement
+ nextMove(Position): Position
AssassinMovement
+nextMove(Position): Position
AssassinEntity
-assassinBribeFailRate: double
-assassinReconRadius: int
+AssassinEntity(JSONObject, int, double, double)
+update(DungeonSubject)
+bribe(Player)
+interact(DungeonSubject)
HydraEntity
- hydraHealthIncreaseRate
- hydraHealthIncreaseAmount
+HydraEntity(JSONObject, int, double, double, double, double)
+getHydraHealthIncreaseRate()
+getHydraHealthIncreaseAmount()
MercenaryMovement
+ pathToPlayer(Position, Position, Dungeon): Position
- generateNextPosition(Position, Map<Position,Position>): Position
- initialiseMaps(Map<Position,Position>, Map<Position,Double>, Position>: void
- sortQueue(Queue<MapNode>): void
- getNextPositions(Position, Dungeon): List<Position>
- checkIfLockedDoor(Entity): Boolean
<<Buildables>>
+ playerHasMaterials(): boolean
+ build(Player): void
+ getType(): String
+ getId(): String
+ setId(String): void
+ getItemResponse(): ItemResponse
Shield
- attackDamage: int
- type: String
- id: String
- durability: int
- defencePoint: double
+ Shield(int)
+ playerHasMaterials(): boolean
+ build(Player): void
+ getType(): String
+ getId(): String
+ getDurability(): int
+ setDurability(int): void
+ getDefencePoints(): int
+ getAttackDamage(): int
+ getItemResponse(): ItemResponse
Bow
- attackDamage: int
- type: String
- id: String
- durability: int
- defencePoint: double
+ Bow(int)
+ playerHasMaterials(): boolean
+ build(Player): void
+ getType(): String
+ getId(): String
+ getDurability(): int
+ setDurability(int): void
+ getDefencePoints(): int
+ getAttackDamage(): int
+ getItemResponse(): ItemResponse
Sceptre
- type: String
+ Sceptre(int)
+ playerHasMaterials(): boolean
+ build(Player): void
+ getType(): String
+ getId(): String
+ getItemResponse(): ItemResponse
MidnightArmour
- attackDamage: int
- type: String
- id: String
- durability: int
- defencePoint: double
+ MidnightArmour(int)
+ playerHasMaterials(): boolean
+ build(Player): void
+ getType(): String
+ getId(): String
+ getDurability(): int
+ setDurability(int): void
+ getDefencePoints(): int
+ getAttackDamage(): int
+ getItemResponse(): ItemResponse
BuildablesFactory
- dungeon: Dungeon
- player: Player
- buildables: List<Buildable>
+ BuildablesFactory(Dungeon)
+ buildItem(String): Buildable
+ isBuildable(String): boolean
+ playerCanBuild(String): boolean
+ getbuildables(): List<String>
<<Weapon>>
+ getDurability(): int
+ setDurability(int): void
+ getDefencePoints(): int
+ getAttackDamage(): int
+ getItemResponse(): ItemResponse