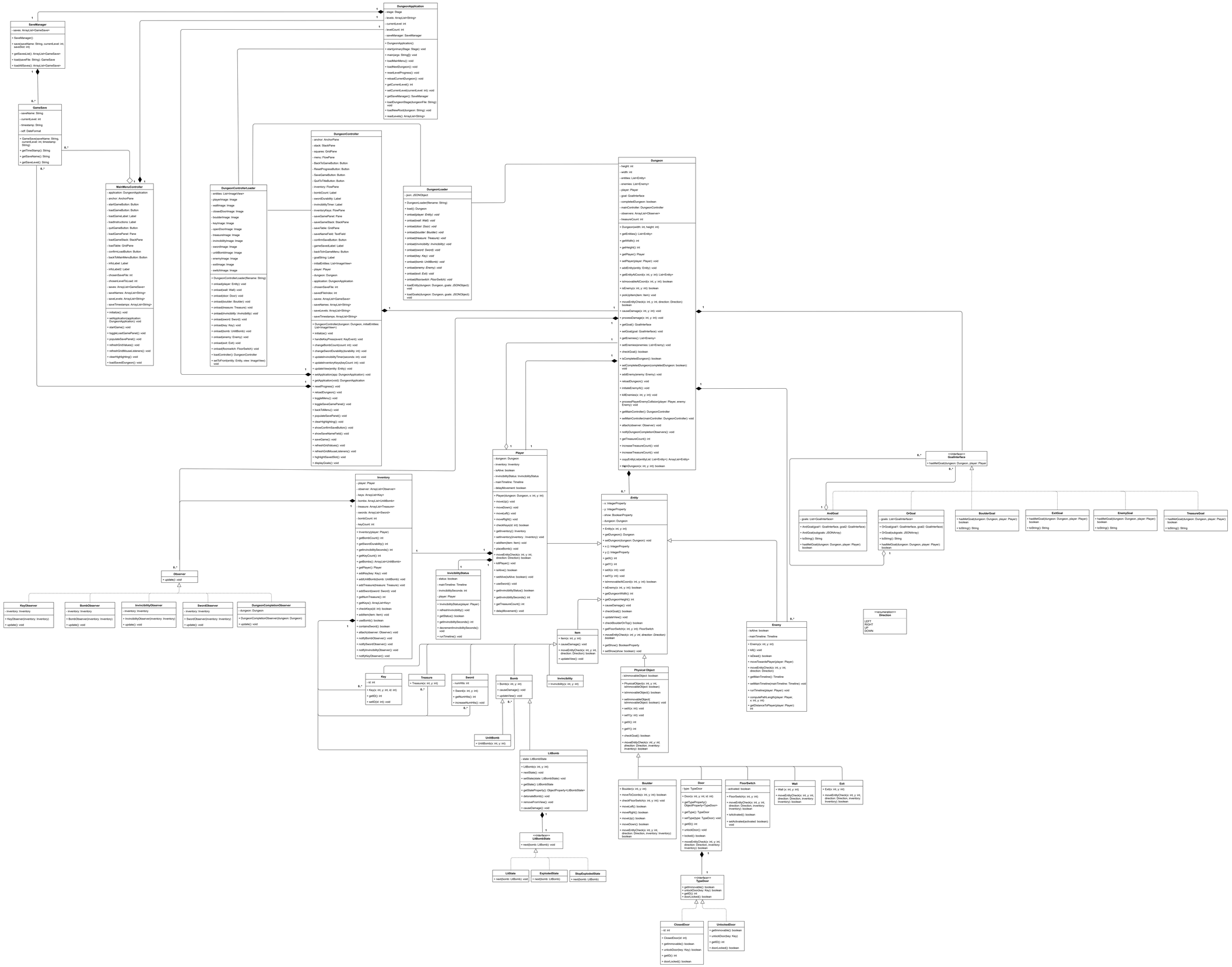
SaveManager
- saves: ArrayList<GameSave>
+ SaveManager()
+ save(saveName: String, currentLevel: int, saveSlot: int)
+ getSavesList(): ArrayList<GameSave>
+ load(saveFile: String): GameSave
+ loadAllSaves(): ArrayList<GameSave>
GameSave
- saveName: String
- currentLevel: int
- timestamp: String
- sdf: DateFormat
+ GameSave(saveName: String, currentLevel: int, timestamp: String)
+ getTimeStamp(): String
+ getSaveName(): String
+ getSaveLevel(): String
DungeonApplication
- stage: Stage
- levels: ArrayList<String>
- currentLevel: int
- levelCount: int
- saveManager: SaveManager
+ DungeonApplication()
+ start(primaryStage: Stage): void
+ main(args: String[]): void
+ loadMainMenu(): void
+ loadNextDungeon(): void
+ resetLevelProgress(): void
+ reloadCurrentDungeon(): void
+ getCurrentLevel(): int
+ setCurrentLevel(currentLevel: int): void
+ getSaveManager(): SaveManager
+ loadDungeonStage(dungeonFile: String): void
+ loadNewRoot(dungeon: String): void
+ readLevels(): ArrayList<String>
MainMenuController
- application: DungeonApplication
- anchor: AnchorPane
- startGameButton: Button
- loadGameButton: Button
- loadGameLabel: Label
- loadInstructions: Label
- quitGameBtn: Button
- loadGamePane: Pane
- loadGameStack: StackPane
- loadTable: GridPane
- confirmLoadBtn: Button
- backToMainMenuBtn: Button
- infoLabel: Label
- infoLabel2: Label
- chosenSaveFile: int
- chosenLevelToLoad: int
- saves: ArrayList<GameSave>
- saveNames: ArrayList<String>
- saveLevels: ArrayList<String>
- saveTimestamps: ArrayList<String>
+ initialize(): void
+ setApplication(application: DungeonApplication): void
+ startGame(): void
+ toggleLoadGamePane(): void
+ populateSavePane(): void
+ refreshGridValues(): void
+ refreshGridMouseListeners(): void
+ clearHighlighting(): void
+ loadSavedDungeon(): void
DungeonControllerLoader
- entities: List<ImageView>
- playerImage: Image
- wallImage: Image
- closedDoorImage: Image
- boulderImage: Image
- keyImage: Image
- openDoorImage: Image
- treasureImage: Image
- invincibilityImage: Image
- swordImage: Image
- unlitBombImage: Image
- enemyImage: Image
- exitImage: Image
- switchImage: Image
+ DungeonControllerLoader(filename: String)
+ onload(player: Entity): void
+ onload(wall: Wall): void
+ onload(door: Door): void
+ onload(boulder: Boulder): void
+ onload(treasure: Treasure): void
+ onload(invincibility: Invincibility): void
+ onload(sword: Sword): void
+ onload(key: Key): void
+ onload(bomb: UnitBomb): void
+ onload(enemy: Enemy): void
+ onload(exit: Exit): void
+ onload(floorswitch: FloorSwitch): void
+ loadController(): DungeonController
+ setToFront(entity: Entity, view: ImageView): void
DungeonController
- anchor: AnchorPane
- stack: StackPane
- squares: GridPane
- menu: FlowPane
- BackToGameButton: Button
- ResetProgressBtn: Button
- SaveGameButton: Button
- QuiTtoTitleBtn: Button
- inventory: FlowPane
- bombCount: Label
- swordDurability: Label
- invincibilityTimer: Label
- inventoryKeys: FlowPane
- saveGamePanel: Pane
- saveGameStack: StackPane
- saveTable: GridPane
- saveNameFeld: TextField
- confirmSaveBtn: Button
- gameSavedLabel: Label
- backToInGameMenu: Button
- goalString: Label
- initialEntities: List<ImageView>
- player: Player
- dungeon: Dungeon
- application: DungeonApplication
- chosenSaveFile: int
- saveFileIndex: int
- saves: ArrayList<GameSave>
- saveNames: ArrayList<String>
- saveLevels: ArrayList<String>
- saveTimestamps: ArrayList<String>
+ DungeonController(dungeon: Dungeon, initialEntities: List<ImageView>)
+ initialize(): void
+ handleKeyPress(event: KeyEvent): void
+ changeBombCount(count: int): void
+ changeSwordDurability(durability: int): void
+ updateInvincibilityTimer(seconds: int): void
+ updateInventoryKeys(keyCount: int): void
+ updateView(entity: Entity): void
+ setApplication(app: DungeonApplication): void
+ getApplication(void): DungeonApplication
+ resetProgress(): void
+ reloadDungeon(): void
+ toggleMenu(): void
+ toggleSaveGamePane(): void
+ backToMenu(): void
+ populateSavePane(): void
+ clearHighlighting(): void
+ showConfirmSaveButton(): void
+ showSaveNameField(): void
+ saveGame(): void
+ refreshGridValues(): void
+ refreshGridMouseListeners(): void
+ highlightSaveSlot(): void
+ displayGoals(): void
DungeonLoader
- json: JSONObject
+ DungeonLoader(filename: String)
+ load(): Dungeon
+ onload(player: Entity): void
+ onload(wall: Wall): void
+ onload(door: Door): void
+ onload(boulder: Boulder): void
+ onload(treasure: Treasure): void
+ onload(invincibility: InvincibilityStatus): void
+ onload(sword: Sword): void
+ onload(key: Key): void
+ onload(bomb: UnitBomb): void
+ onload(enemy: Enemy): void
+ onload(exit: Exit): void
+ onload(floorswitch: FloorSwitch): void
+ loadEntity(dungeon: Dungeon, goals: JSONObject): void
+ loadGoals(dungeon: Dungeon, goals: JSONObject): void
Dungeon
- height: int
- width: int
- entities: List<Entity>
- enemies: List<Enemy>
- player: Player
- goal: GoalInterface
- completedDungeon: boolean
- mainController: DungeonController
- observers: ArrayList<Observer>
- treasureCount: int
+ Dungeon(width: int, height: int)
+ getEntities(): List<Entity>
+ getWidth(): int
+ getHeight(): int
+ getPlayer(): Player
+ setPlayer(player: Player): void
+ addEntity(entity: Entity): void
+ getEntiyAtCoord(x: int, y: int): List<Entity>
+ isImovableAtCoord(x: int, y: int): boolean
+ isEnemy(x: int, y: int): boolean
+ pickUpItem(item: Item): void
+ moveEntityCheck(x: int, y: int, direction: Direction): boolean
+ causeDamage(x: int, y: int): void
+ processDamage(x: int, y: int): void
+ getGoal(): GoalInterface
+ setGoal(goal: GoalInterface): void
+ getEnemies(): List<Enemy>
+ setEnemies(enemies: List<Enemy>): void
+ checkGoal(): boolean
+ isCompletedDungeon(): boolean
+ setCompletedDungeon(completedDungeon: boolean): void
+ addEnemy(anemy: Enemy): void
+ reloadDungeon(): void
+ initialiseEnemyAI(): void
+ killEnemies(x: int, y: int): void
+ processPlayerEnemyCollision(player: Player, enemy: Enemy): void
+ getMainController(): DungeonController
+ setMainController(mainController: DungeonController): void
+ attach(observer: Observer): void
+ notifyDungeonCompletionObservers(): void
+ getTreasureCount(): int
+ increaseTreasureCount(): void
+ increaseTreasureCount(): void
+ copyEntiyLisEntity(Lst: List<Entity>): ArrayList<Entity>
+ flipDungeon(x: int, y: int): boolean
Player
- dungeon: Dungeon
- inventory: Inventory
- isAlive: boolean
- invincibilityStatus: InvincibilityStatus
- mainTimeline: Timeline
- delayMovement: boolean
+ Player(dungeon: Dungeon, x: int, y: int)
+ moveUp(): void
+ moveDown(): void
+ moveLeft(): void
+ moveRight(): void
+ checkKeys(id: int): boolean
+ getInventory(): Inventory
+ setInventory(inventory : Inventory): void
+ addItem(item: Item): void
+ placeBomb(): void
+ moveEntityCheck(x: int, y: int, direction: Direction): boolean
+ killPlayer(): void
+ isAlive(): boolean
+ setAlive(isAlive: boolean): void
+ useSword(): void
+ getInvincibilityStatus(): boolean
+ getInvincibilitySeconds(): int
+ getTreasureCount(): int
+ delayMovement(): void
Entity
- x: IntegerProperty
- y: IntegerProperty
- show: BooleanProperty
- dungeon: Dungeon
+ Entity(x: int, y: int)
+ getDungeon(): Dungeon
+ setDungeon(dungeon: Dungeon): void
+ x (): IntegerProperty
+ y (): IntegerProperty
+ getX(): int
+ getY(): int
+ setX(x: int): void
+ setY(y: int): void
+ isImovableAtCoord(x: int, y: int): boolean
+ isEnemy(x: int, y: int): boolean
+ getDungeonWidth(): int
+ getDungeonHeight(): int
+ causeDamage(): void
+ checkGoal(): boolean
+ updateView(): void
+ checkBoulderOnTop(): boolean
+ getFloorSwitch(x: int, y: int): FloorSwitch
+ moveEntityCheck(x: int, y: int, direction: Direction): boolean
+ getShow(): BooleanProperty
+ setShow(show: boolean): void
Inventory
- player: Player
- observer: Array<Observer>
- keys: ArrayList<Key>
- bombs: ArrayLsit<UnitBomb>
- treasure: ArrayLsit<Treasure>
- swords: ArrayLsit<Sword>
- bombCount: int
- keyCount: int
+ Inventory(player: Player)
+ getBombCount(): int
+ getSwordDurability(): int
+ getInvincibilitySeconds(): int
+ getKeyCount(): int
+ getBombs(): ArrayLsit<UnitBomb>
+ getPlayer(): Player
+ addKey(key: Key): void
+ addUnitBomb(bomb: UnitBomb): void
+ addTreasure(treasure: Treasure): void
+ addSword(sword: Sword): void
+ getNumTreasure(): int
+ getKeys(): ArrayLsit<Key>
+ checkKeys(id: int): boolean
+ addItem(item: Item): void
+ useBomb(): boolean
+ containsSword(): boolean
+ attach(observer: Observer): void
+ notifyBombObserver(): void
+ notifySwordObserver(): void
+ notifyInvincibilityObserver(): void
+ notifyKeyObserver(): void
InvincibilityStatus
- status: boolean
- mainTimeline: Timeline
- invincibilySeconds: int
- player: Player
+ InvincibilityStatus(player: Player)
+ refreshInvincibility(): void
+ getStatus(): boolean
+ getInvincibilySeconds(): int
+ decrementInvincibilySeconds(): void
+ runTimeline(): void
Observer
+ update(): void
KeyObserver
- inventory: Inventory
+ KeyObserver(inventory: Inventory)
+ update(): void
BombObserver
- inventory: Inventory
+ BombObserver(inventory: Inventory)
+ update(): void
InvincibilityObserver
- inventory: Inventory
+ InvincibilityObserver(inventory: Inventory)
+ update(): void
SwordObserver
- inventory: Inventory
+ SwordObserver(inventory: Inventory)
+ update(): void
DungeonCompletionObserver
- dungeon: Dungeon
+ DungeonCompletionObserver(dungeon: Dungeon)
+ update(): void
Item
+ Item(x: int, y: int)
+ causeDamage(): void
+ moveEntityCheck(x: int, y: int, direction: Direction): boolean
+ updateView(): void
Key
- id: int
+ Key(x: int, y: int, id: int)
+ getID(): int
+ setID(id: int): void
Treasure
+ Treasure(x: int, y: int)
Sword
- numHits: int
+ Sword(x: int, y: int)
+ getNumHits(): int
+ increaseNumHits(): void
Bomb
+ Bomb(x: int, y: int)
+ causeDamage(): void
+ updateView(): void
Invincibility
+ Invincibility(x: int, y: int)
UnitBomb
+ UnitBomb(x: int, y: int)
LitBomb
- state: LitBombState
+ LitBomb(x: int, y: int)
+ nextState(): void
+ setState(state: LitBombState): void
+ getState(): LitBombState
+ getStateProperty(): ObjectProperty<LitBombState>
+ detonateBomb(): void
+ removeFromView(): void
+ causeDamage(): void
<<Interface>>
LitBombState
+ next(bomb: LitBomb): void
LitState
+ next(bomb: LitBomb): void
ExplodedState
+ next(bomb: LitBomb)
StopExplodedState
+ next(bomb: LitBomb)
Physical Object
- isImmovable:Object: boolean
+ PhysicalObject(x: int, y: int, isImmovable:Object: boolean)
+ isImmovable(Object): boolean
+ setImmovable(Object, isImmovable:Object: boolean): void
+ setX(x: int): void
+ setY(y: int): void
+ getX(): int
+ getY(): int
+ checkGoal(): boolean
+ moveEntityCheck(x: int, y: int, direction: Direction, inventory: Inventory): boolean
Boulder
+ Boulder(x: int, y: int)
+ moveToCoord(x: int, y: int): boolean
+ checkFloorSwitch(x: int, y: int): void
+ moveLeft(): boolean
+ moveRight(): boolean
+ moveUp(): boolean
+ moveDown(): boolean
+ moveEntityCheck(x: int, y: int, direction: Direction, inventory: Inventory): boolean
Door
- type: TypeDoor
+ Door(x: int, y: int, id: int)
+ getTypeProperty(): ObjectProperty<TypeDoor>
+ getType(): TypeDoor
+ setType(type: TypeDoor): void
+ getID(): int
+ unlockDoor(): void
+ locked(): boolean
+ moveEntityCheck(x: int, y: int, direction: Direction, inventory: Inventory): boolean
FloorSwitch
- activated: boolean
+ FloorSwitch(x: int, y: int)
+ moveEntityCheck(x: int, y: int, direction: Direction, inventory: Inventory): boolean
+ isActivated(): boolean
+ setActivated(activated: boolean): void
Wall
+ Wall (x: int, y: int)
+ moveEntityCheck(x: int, y: int, direction: Direction, inventory: Inventory): boolean
Exit
+ Exit(x: int, y: int)
+ moveEntityCheck(x: int, y: int, direction: Direction, inventory: Inventory): boolean
<<Interface>>
TypeDoor
+ getImmovable(): boolean
+ unlockDoor(key: Key): boolean
+ getID(): int
+ doorLocked(): boolean
ClosedDoor
- id: int
+ ClosedDoor(id: int)
+ getImmovable(): boolean
+ unlockDoor(key: Key): boolean
+ getID(): int
+ doorLocked(): boolean
UnlockedDoor
+ getImmovable(): boolean
+ unlockDoor(key: Key)
+ getID(): int
+ doorLocked(): boolean
Enemy
- isAlive: boolean
- mainTimeline: Timeline
+ Enemy(x: int, y: int)
+ kill(): void
+ isDead(): boolean
+ moveTowardsPlayer(player: Player)
+ moveEntityCheck(x: int, y: int, direction: Direction)
+ getMainTimeline(): Timeline
+ setMainTimeline(mainTimeline: Timeline): void
+ runTimeline(player: Player): void
+ computePathLength(player: Player, x: int, y: int)
+ getDistanceToPlayer(player: Player): int
<<Interface>>
GoalInterface
+ hasMetGoal(dungeon: Dungeon, player: Player
AndGoal
- goals: List<GoalInterface>
+ AndGoal(goal1: GoalInterface, goal2: GoalInterface)
+ AndGoal(subgoals: JSONArray)
+ toString(): String
+ hasMetGoal(dungeon: Dungeon, player: Player): boolean
OrGoal
- goals: List<GoalInterface>
+ OrGoal(goal1: GoalInterface, goal2: GoalInterface)
+ OrGoal(subgoals: JSONArray)
+ toString(): String
+ hasMetGoal(dungeon: Dungeon, player: Player): boolean
BoulderGoal
+ hasMetGoal(dungeon: Dungeon, player: Player): boolean
+ toString(): String
ExitGoal
+ hasMetGoal(dungeon: Dungeon, player: Player): boolean
+ toString(): String
EnemyGoal
+ hasMetGoal(dungeon: Dungeon, player: Player): boolean
+ toString(): String
TreasureGoal
+ hasMetGoal(dungeon: Dungeon, player: Player): boolean
+ toString(): String
<<enumeration>>
Direction
LEFT
RIGHT
UP
DOWN