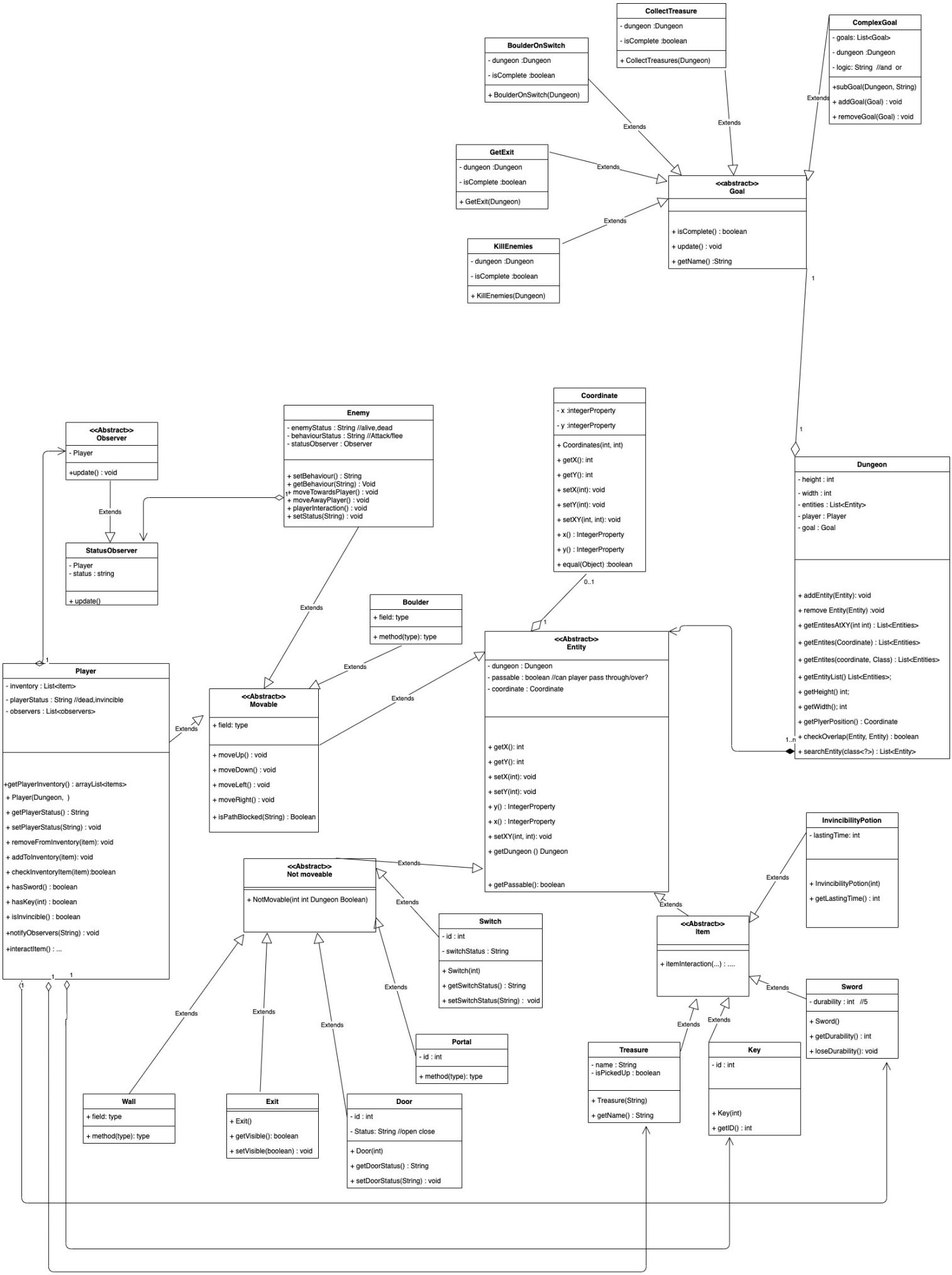
CollectTreasure
- dungeon :Dungeon
- isComplete :boolean
+ CollectTreasures(Dungeon)
ComplexGoal
- goals: List<Goal>
- dungeon :Dungeon
- logic: String //and or
+subGoal(Dungeon, String)
+ addGoal(Goal) : void
+ removeGoal(Goal) : void
BoulderOnSwitch
- dungeon :Dungeon
- isComplete :boolean
+ BoulderOnSwitch(Dungeon)
GetExit
- dungeon :Dungeon
- isComplete :boolean
+ GetExit(Dungeon)
KillEnemies
- dungeon :Dungeon
- isComplete :boolean
+ KillEnemies(Dungeon)
<<abstract>>
Goal
+ isComplete() : boolean
+ update() : void
+ getName() :String
Extends
1
Coordinate
- x :integerProperty
- y :integerProperty
+ Coordinates(int, int)
+ getX(): int
+ getY(): int
+ setX(int): void
+ setY(int): void
+ setXY(int, int): void
+ x() : IntegerProperty
+ y() : IntegerProperty
+ equal(Object) :boolean
Enemy
- enemyStatus : String //alive,dead
- behaviourStatus : String //Attack/flee
- statusObserver : Observer
+ setBehaviour() : String
+ getBehaviour(String) : Void
+ moveTowardsPlayer() : void
+ moveAwayPlayer() : void
+ playerInteraction() : void
+ setStatus(String) : void
<<Abstract>>
Observer
- Player
+update() : void
StatusObserver
- Player
- status : string
+ update()
Dungeon
- height : int
- width : int
- entities : List<Entity>
- player : Player
- goal : Goal
+ addEntity(Entity): void
+ remove Entity(Entity) :void
+ getEntitesAtXY(int int) : List<Entities>
+ getEntites(Coordinate) : List<Entities>
+ getEntites(coordinate, Class) : List<Entities>
+ getEntityList() List<Entities>;
+ getHeight() int;
+ getWidth(); int
+ getPlyerPosition() : Coordinate
+ checkOverlap(Entity, Entity) : boolean
+ searchEntity(class<?>) : List<Entity>
1..n
0..1
Boulder
+ field: type
+ method(type): type
<<Abstract>>
Entity
- dungeon : Dungeon
- passable : boolean //can player pass through/over?
- coordinate : Coordinate
+ getX(): int
+ getY(): int
+ setX(int): void
+ setY(int): void
+ y() : IntegerProperty
+ x() : IntegerProperty
+ setXY(int, int): void
+ getDungeon () Dungeon
+ getPassable(): boolean
Player
- inventory : List<Item>
- playerStatus : String //dead,invincible
- observers : List<observers>
+getPlayerInventory() : arrayList<items>
+ Player(Dungeon,)
+ getPlayerStatus() : String
+ setPlayerStatus(String) : void
+ removeFromInventory(item): void
+ addToInventory(item): void
+ checkInventoryItem(item):boolean
+ hasSword() : boolean
+ hasKey(int) : boolean
+ isInvincible() : boolean
+notifyObservers(String) : void
+interactItem() : ...
<<Abstract>>
Movable
+ field: type
+ moveUp() : void
+ moveDown() : void
+ moveLeft() : void
+ moveRight() : void
+ isPathBlocked(String) : Boolean
InvincibilityPotion
- lastingTime: int
+ InvincibilityPotion(int)
+ getLastingTime() : int
<<Abstract>>
Not moveable
+ NotMovable(int int Dungeon Boolean)
Switch
- id : int
- switchStatus : String
+ Switch(int)
+ getSwitchStatus() : String
+ setSwitchStatus(String) : void
<<Abstract>>
Item
+ itemInteraction(...) : ....
Sword
- durability : int //5
+ Sword()
+ getDurability() : int
+ loseDurability(): void
Treasure
- name : String
- isPickedUp : boolean
+ Treasure(String)
+ getName() : String
Key
- id : int
+ Key(int)
+ getID() : int
Portal
- id : int
+ method(type): type
Wall
+ field: type
+ method(type): type
Exit
+ Exit()
+ getVisible(): boolean
+ setVisible(boolean) : void
Door
- id : int
- Status: String //open close
+ Door(int)
+ getDoorStatus() : String
+ setDoorStatus(String) : void