Subject Observer Design Pattern

SubjectBase
<Interface>
Register(player)
Unregister(player)
NotifyAll(message)

ObserverBase
Notify(message)

Iterator Design Pattern

Room
<Iterable>
Room(walls, isDragon, isTreasure)
GetNames() : string
DrawRoom(writer, direction)
IsDoor(direction) : bool
IsDragon(): booll
IsTreasure() : bool
GetNextDoorRoomNumber() : number
AttackPlayer(name, dmg) : Info
AttackDragon(dmg) : Info
GetTreasure() : treasure
Register(player)
UntegisterPlayer(player)
NotifyAll(message)
begin() : Iterator
end() : iterator

ObserverPlayer_mt
ObserverPlayer_mt(player)
GetPlayer() : player
GetName() : name
IsEqual(player) : bool
AttackPlayerResponse(dmg) : Info

0,...N

SubjectRoom_mt
SubjectRoom_mt()
GetNames() : string
AttackPlayer(name, dmg) : Info

1

DungeonRoom
DungeonRoom(doors,
isDragon,IsTreasure)
Draw(grid, direction,coordinates (x,y))

For printing

Wall
Wall(isDoor, nextRoom)
IsDoor) : bool
GetNextRoomNumber() : roomNumber

0,....,N

4

Player

Player(name, Writer)
NotifyPlayer(message)
AttackPlayerResponse(dmg number) : dmg back
Text
<Getters>
GetName() : Name
GetRoomNumber() : room number
GetDirection() : Direction
GetLife() : life points
GetAttack() : attack points
isAlive() : bool
<Setters>
SetRoomNumber(Number)
SetDirection(Direction)
SetPlayerGameArgument(string)
SetLife(Number)
SetAttack(Number)

Dungeon_mt

Dungeon_mt()
DrawRoom(Writer, roomNumber, direction)
Walk_mt(room number, direction) : room number
NotifyRoom(roomNumber, message)
IsMonster(roomNumber) : bool
GetNames(roomNumber)
Shout(roomNumber, Message)
RegisterPlayer(player, roomNumber)
UnregisterPlayer(player, roomNumber)
AttackPlayer(roomNumber, attackerInfo,
    attacked info) : Info
AttackDragon(roomNumber, dmg) : Info
GetTreasure(roomNumber)

Grid

Grid(x,y)
Clean()
Print()
For printing

Monster

Monster(life,dmg)
DamageAndAttack(dmg) : backDmg
IsAlive() : bool
GetLifePoints() : number

Dragon

Dragon()

GameExecuter

GameExecuter(Writer, Reader, Dungeon, Player)
Execute()

States Design Pattern

Action
<Interface>

Act(Duengeon,Player) : Response

Strategy Design Pattern

Game

Play()
ServerPlay()
GeneralPlay(Writer,Reader)

Walk

Right

Left

Talk

Shout

Attack

Take

UserParser

ParseUserInput(string) : Command

Reader

ReadLine() : string

Writer

Write(string)

ActionResponse

GetResponse() : Data

NetReader

CoutReader

CoutWriter

NetWeiter

StringActionResponse

Command Design Pattern