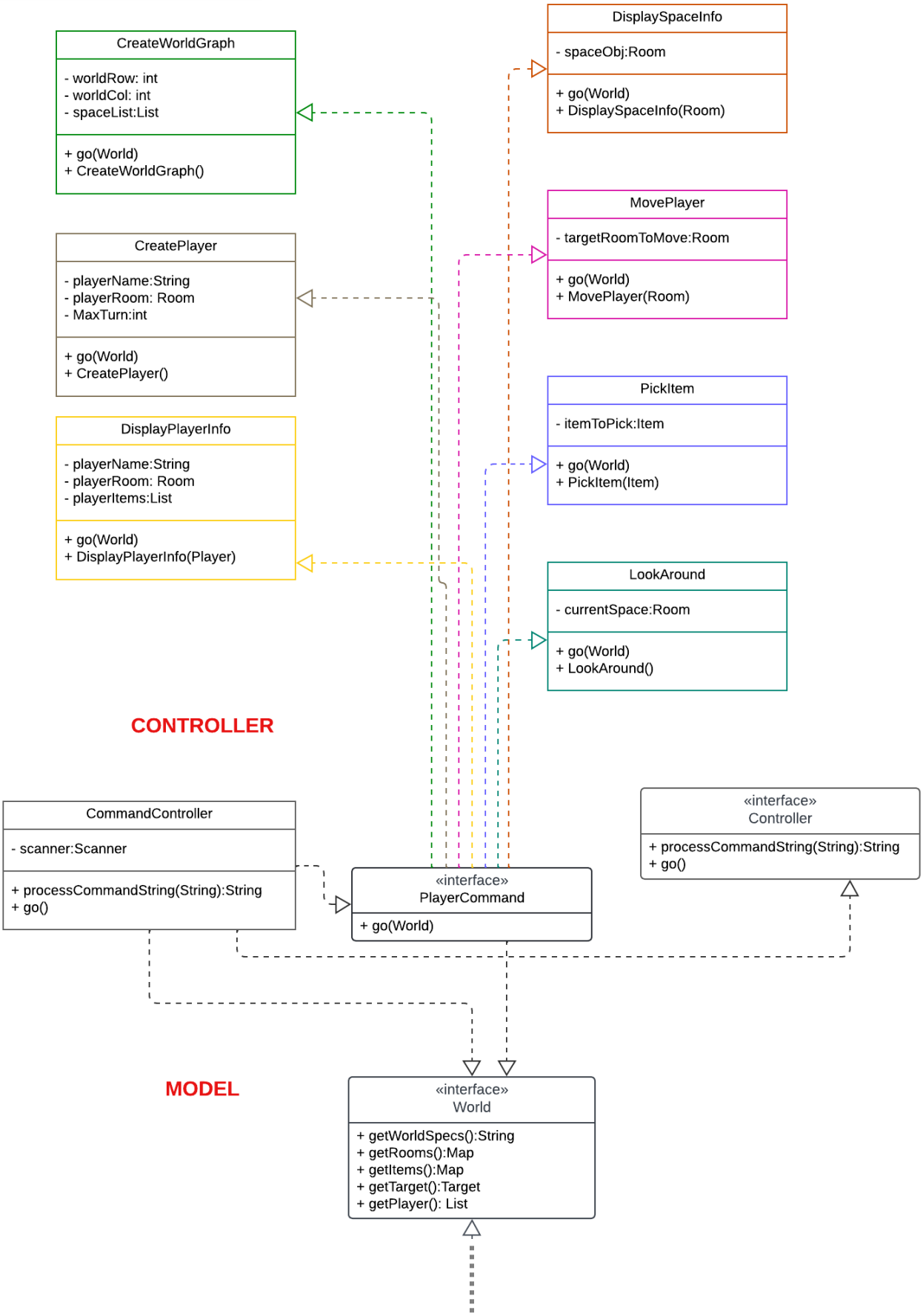
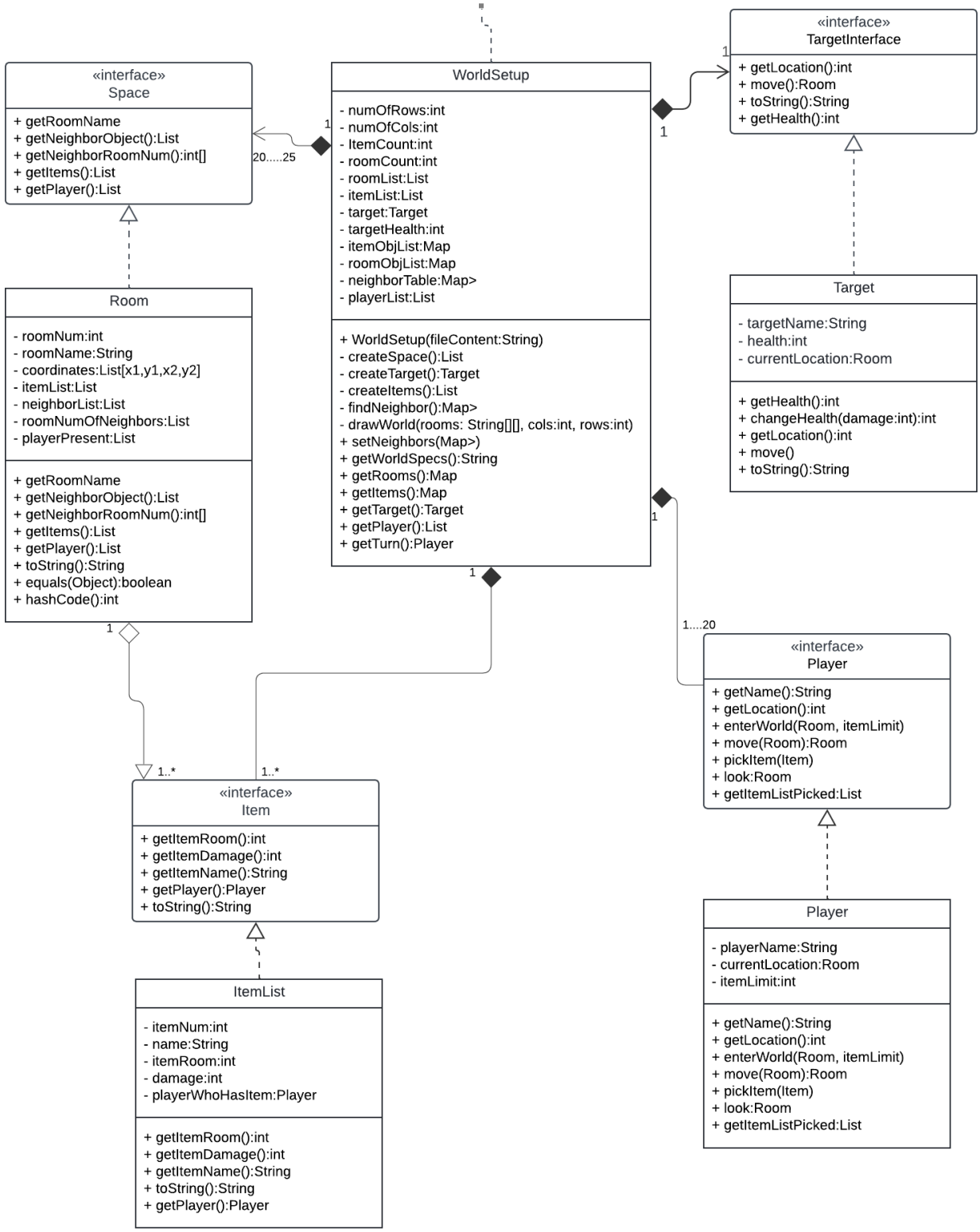
Kill Doctor Lucky UML

Kill Doctor Lucky **UML class diagrams**

This diagram outlines a proposed refined solution to the game to be built ie. Kill Doctor Lucky

Edorh Carlos Semho | edorh.c@northeastern.edu




«interface»
Space
+ getRoomName
+ getNeighborObject():List
+ getNeighborRoomNum():int[]
+ getItems():List
+ getPlayer():List
WorldSetup
- numOfRows:int
- numOfCols:int
- ItemCount:int
- roomCount:int
- roomList:List
- itemList:List
- target:Target
- targetHealth:int
- itemObjList:Map
- roomObjList:Map
- neighborTable:Map>
- playerList:List
+ WorldSetup(fileContent:String)
- createSpace():List
- createTarget():Target
- createItems():List
- findNeighbor():Map>
- drawWorld(rooms: String[][], cols:int, rows:int)
+ setNeighbors(Map>)
+ getWorldSpecs():String
+ getRooms():Map
+ getItems():Map
+ getTarget():Target
+ getPlayer():List
+ getTurn():Player
«interface»
TargetInterface
+ getLocation():int
+ move():Room
+ toString():String
+ getHealth():int
Target
- targetName:String
- health:int
- currentLocation:Room
+ getHealth():int
+ changeHealth(damage:int):int
+ getLocation():int
+ move()
+ toString():String
Room
- roomNum:int
- roomName:String
- coordinates:List[x1,y1,x2,y2]
- itemList:List
- neighborList:List
- roomNumOfNeighbors:List
- playerPresent:List
+ getRoomName
+ getNeighborObject():List
+ getNeighborRoomNum():int[]
+ getItems():List
+ getPlayer():List
+ toString():String
+ equals(Object):boolean
+ hashCode():int
«interface»
Player
+ getName():String
+ getLocation():int
+ enterWorld(Room, itemLimit)
+ move(Room):Room
+ pickItem(Item)
+ look:Room
+ getItemListPicked:List
Player
- playerName:String
- currentLocation:Room
- itemLimit:int
+ getName():String
+ getLocation():int
+ enterWorld(Room, itemLimit)
+ move(Room):Room
+ pickItem(Item)
+ look:Room
+ getItemListPicked:List
«interface»
Item
+ getItemRoom():int
+ getItemDamage():int
+ getItemName():String
+ getPlayer():Player
+ toString():String
ItemList
- itemNum:int
- name:String
- itemRoom:int
- damage:int
- playerWhoHasItem:Player
+ getItemRoom():int
+ getItemDamage():int
+ getItemName():String
+ toString():String
+ getPlayer():Player
1
20.....25
1
1
1
1....20
1
1..*
1..*

Kill Doctor Lucky Test Plan

CONTROLLER TESTS

Test "DisplayPlayerInfo" Interface

Test DisplayPlayerInfo()	Input	Expected output
Get Info of Player 1	Player1.DisplayPlayerInfo()	Player Name = Quick Fox, Player Room = Room2, Player Item = Knife
Get Info of a non-existent player	Player50.DisplayPlayerInfo()	IllegalArgumentException

Test "CreatePlayer" Interface

Test CreatePlayer()	Input	Expected output
Create Player 3	Player details: Name = Quick Fox, First Room to Enter = 4	Player created
Try to create Player who enters non-existent room	Player details: Name = Quick Fox, First Room to Enter = 50	IllegalArgumentException

Test "CreateWorldGraph" Interface

Test CreateWorldGraph()	Input	Expected output
Create world with correct dimensions	Row = 50, Col = 50	World graphical representation created
Wrong world dimensions given	Row = 0, Col = -50	IllegalArgumentException

Test "DisplaySpaceInfo" Interface

Test DisplaySpaceInfo()	Input	Expected output
Display space info of player 5	Player5.DisplaySpaceInfo()	Room = Room5, Items = Item3, Neighbors = [6, 8, 10]
Display space info for a non-existent player	Player50.DisplaySpaceInfo()	IllegalArgumentException

Test "MovePlayer" Interface

Test MovePlayer()	Input	Expected output
Move Player to valid Neighbor Room	Player2.MovePlayer(Room5)	Player2 moved from room4 to room5
Move Player to a room which is not a neighbor	Player5. MovePlayer(Room16)	IllegalArgumentException

Test “PickItem” Interface

Test PickItem()	Input	Expected output
Player 4 tries to pick Item5 when he still has eligibility to pick items.	Player2.PickItem(item5)	Item picked and removed from space
Player tries to pick an item when he already has max number of items allowed to pick.	Player2.PickItem()	IllegalStateException

Test “LookAround” Interface

Test LookAround()	Input	Expected output
Player gets details of neighboring rooms	Player2.LookAround()	Neighbors = [6, 8, 10]
LookAround called for non-existent player	Player50.LookAround()	IllegalArgumentException

MODEL TESTS

Test the “World” Interface

Test WorldSetup() constructor	Input	Expected output
Input valid file path	mansion.txt	Constructor parses through the file
Input invalid file path	world.txt	Throws FileNotFoundException

Test getWorldSpecs()	Input	Expected output
Valid file was input	mansion.txt	File content returned as a string

Test getRooms()	Input	Expected output
Valid file was input with valid roomCount (roomCount>0)	mansion.txt – roomCount = 10	10
Valid file was input with invalid roomCount (roomCount<1)	mansion.txt – roomCount = 0	IllegalArgumentException

Test getTarget()	Input	Expected output
Valid file was input with Target as String	mansion.txt – Target = Doctor Lucky	Doctor Lucky
Valid file was input with Target as non-string or no target	mansion.txt – Target = “”	IllegalArgumentException
Valid file was input with Target as non-string or no target	mansion.txt – Target = 123	IllegalArgumentException

Test getItems()	Input	Expected output
Valid file was input with valid roomCount (roomCount>0)	mansion.txt – roomCount = 10	10
Valid file was input with invalid roomCount (roomCount<1)	mansion.txt – roomCount = 0	IllegalArgumentException

Test the "Target" Interface and "Target" Class

Test Target(name:String, health:int) constructor	Input	Expected output
Valid Target String	mansion.txt – Target = Doctor Lucky	Target object created
Valid Health	mansion.txt – health = 50	Target object created
Invalid Target String	mansion.txt – Target = ""	IllegalArgumentException
Invalid Health	mansion.txt – health = -5 or 0	IllegalArgumentException

Test getLocation	Input	Expected output
At start of game	Target.getLocation()	Returned value = 1

Test move()	Input	Expected output
currentRoom = 1	currentRoom = 1	Target moved to room 2
currentLocation = roomCount	currentLocation = roomCount	Target moved to room 1

Test "Space" Interface

Test getNeighbor()	Input	Expected output
Get neighbors of Room 1	Room1.getNeighbor()	[Room2, Room4, Room5]
Get neighbors of Room 8	Room8.getNeighbor()	[Room7, Room16]
Get neighbors of nonexistent room	Room50.getNeighbor()	IllegalArgumentException

Test getItems()	Input	Expected output
Get items of Room with 1 item	Room11.getItems()	[Item17]
Get items of Room with 0 item	Room3.getItems()	[]
Get items of Room with >1 items	Room2.getItems()	[Item7, Item12]
Get Items of nonexistent room	Room50.getItems()	IllegalArgumentException

Test getPlayer()	Input	Expected output
Get players in a room with 1 player	Room2.getPlayer()	[Player3]
Get Players in a room with 2 players	Room3.getItems()	[Player1, Player 5]

Get players in a room with 0 players	Room5.getItems()	[]
Get Player in nonexistent room	Room50.getItems()	IllegalArgumentException

Test “Item” Interface

Test getItemRoom()	**Input**	**Expected output**
Get Room where particular Item is located	Item2.getItemRoom()	4
Get Room for null Items	Item40.getItemRoom()	IllegalArgumentException

Test getItemDamage()	**Input**	**Expected output**
Get damage for Item	Item2.getItemDamage()	2
Get damage for null Items	Item40.getItemRoom()	IllegalArgumentException

Test getItemName()	**Input**	**Expected output**
Get name for of Item	Item2.getItemName()	Letter Opener
Get name for null Items	Item40.getItemRoom()	IllegalArgumentException

Test getPlayerWhoHasItem()	**Input**	**Expected output**
Get Player who has Item 3	Item2.getPlayerWhoHasItem()	[Player 3]
When item is not used by any Player	Item5.getPlayerWhoHasItem()	[]
Get Player for null Items	Item40.getPlayerWhoHasItem()	IllegalArgumentException

Test “Player” Interface

Test getName()	**Input**	**Expected output**
Get Player name	Player2.getName()	Quick Fox
Get Player Name for Nonexistent Player	Player50.getName()	IllegalArgumentException

Test getLocation()	**Input**	**Expected output**
Get location of a Player	Player3.getLocation()	Room2
Get location of a non-existent Player	Player50.getLocation()	IllegalArgumentException

Test enterWorld()	**Input**	**Expected output**
Player enters room 1 at start of game	Player1.enterWorld(Room1)	Successfully Entered.
Player tries to enter nonexistent Space	Player2.enterWorld(Room50)	IllegalArgumentException

Test move()	**Input**	**Expected output**
Player moves to neighboring room	Player1.move(room5)	Success

Player moves to a room which is not a neighbor of current room	Player1.move(room10)	IllegalArgumentException

Test look()	**Input**	**Expected output**
Player looks around room	Player2.look()	[Room1 – has Items 2,3 and Player 3, Room3 has Items 15]

Test getItemListPicked()	**Input**	**Expected output**
Player has some items	Player5.getItemListPicked	[Item1, Item17]
Player has no items	Player2.getItemListPicked	[]