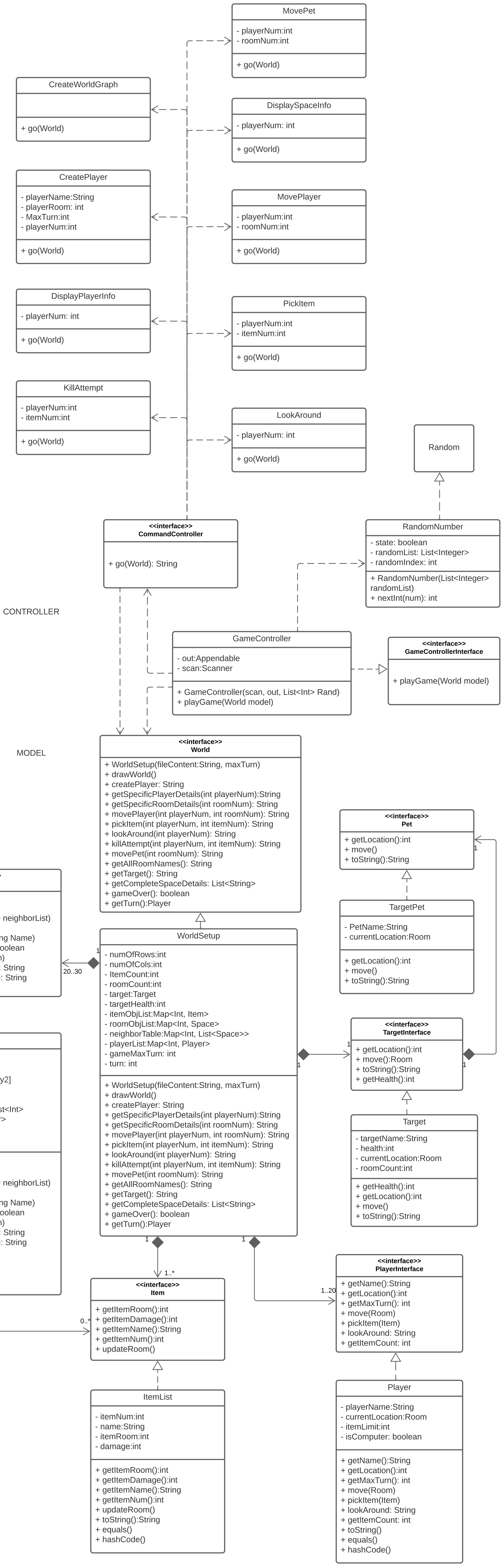
Rakshika Raju || raju.r@northeastern.edu

Kill Doctor Lucky UML



Kill Doctor Lucky Test Plan

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 itemCount (roomCount>0)	mansion.txt – itemCount = 15	15
Valid file was input with invalid itemCount (roomCount<1)	mansion.txt – itemCount = 0	IllegalArgumentException

Test pickItem()	Input	Expected output
Valid PlayerName and Item ID given: Player present in room with item ID	Player 1 pick item 2: pickItem(1,2)	Picked successfully
Valid PlayerName and Item ID given: Player not in room with item ID	Player 1 pick item 5: pickItem(1,5)	Item not present in Room, try again.
Invalid PlayerName or Item ID given	Player 10 picks item 90: pickItem(10,90)	IllegalArgumentException

Valid PlayerName and Item ID given: Player already reached maxItem count	Player 1 has max item count of 2 and tries to pick 3rd item: pickItem(1,5) pickItem(1,4) pickItem(1,10)	Max item count reached, can't pick item.

Test move()	**Input**	**Expected output**
Valid PlayerName and room ID given: roomID is a neighbor of present room	Player 1 move to room 5: move(1,5)	Moved successfully
Valid PlayerName and room ID given: roomID not a neighbor of present room.	Player 1 move to room 10: move(1,10)	Room not a neighbor, lose turn.
Invalid PlayerName or room ID given	Player 10 move to room 90: move(10,90)	IllegalArgumentException

Test lookAround()	**Input**	**Expected output**
Valid PlayerNum given	lookAround(1)	Correct look around details provided.
Invalid PlayerName	lookAround(10)	IllegalArgumentException

Test killAttempt()	**Input**	**Expected output**
Valid PlayerName given: Target in not in same room as player.	Player 1 attempts kill: killAttempt(1)	Attempt failed.
Valid PlayerName given: Target in same space as player, but another player see's the attack	Player 2 attempts to kill: killAttempt(2)	Player 1 say you attempt to kill, kill failed.
Invalid PlayerName given	Player 10 attempts to kill: killAttempt(10)	IllegalArgumentException
Valid PlayerName given: Target in same space as player, and no player see's the attack	Player 3 attempts to kill: killAttempt(3)	Attack successful.

Test movePet()	**Input**	**Expected output**
Valid PlayerName and room ID given: roomID is a neighbor of present room	Player 1 move pet to room 5: movePet(1,5)	Moved successfully
Valid PlayerName and room ID given: roomID not a neighbor	Player 1 move to room 10: movePet(1,10)	Moved successfully
Invalid PlayerName or room ID given	Player 10 move to room 90: movePet(10,90)	IllegalArgumentException

Test getSpecificPlayerDetails()	**Input**	**Expected output**
Valid Player number	getSpecificPlayerDetails(3)	Player 3 details provided.

Invalid Player number	getSpecificPlayerDetails(20)	IllegalArgumentException

Test getSpecificSpaceDetails()	**Input**	**Expected output**
Valid space number	getSpecificSpaceDetails(4)	Space = 4, Space Name = foyer, Items = [2], Players = {}
Valid space number with players in Space	getSpecificSpaceDetails(3)	Space = 3, Space Name = kitchen, Items = [3, 6], Players = {player1, Player2}
Valid space number with Target	getSpecificSpaceDetails(3)	Space = 3, Space Name = kitchen, Items = [3, 6], Players = {player1, Target}
Valid space number with Pet	getSpecificSpaceDetails(5)	Space = 5, Space Name = piazza, Items = [], Players = {player5, pet}
Invalid space number	getSpecificSpaceDetails(100)	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
Game started and 3 turns completed.	Target.getLocation()	Returned value = 4

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

Test getHealth()	**Input**	**Expected output**
Player attacks target	Player 1 attacks target successfully: Player1.killAttempt()	Health = 47
Player attacks target	Player 2 attack on target not a success: Player2.killAttempt()	Health = 50

Test "Space" Interface

Test getNeighbor()	Input	Expected output
Get neighbors of Room 1	Room1.getNeighbor()	[Room2, Room4, Room5]
Get neighbors of Room 1 when pet is in room 4 which is a neighbor	Room1.getNeighbor()	[Room2, 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 hasPet()	Input	Expected output
Space has the pet	Room2.hasPet()	true
Space does not have pet	Room3.hasPet()	false

Test hasItem()	Input	Expected output
Space has the item	Room1.hasItem(2)	true
Space does not have item. Item could be non-existent item as well.	Room1.hasItem(89)	false

Test hasTarget()	Input	Expected output
Space has the Target	Room5.hasTarget()	true
Space does not have the Target	Room12.hasTarget()	false

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 tries to move to non-neighboring space	Player1.move(room8)	Not a neighbor, move failed.
Player moves to a invalid room	Player1.move(room100)	IllegalArgumentException

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

Test pickItem()	Input	Expected output

Player tries to pick item: Player present in room with item ID	Player1.pickItem(2)	Picked successfully
Player tris to pick item: Player not in room with item ID	Player1.pickItem(5)	Item not present in Room, try again.
Invalid PlayerName or Item ID given	Player.pickItem(90)	IllegalArgumentException
Valid PlayerName and Item ID given: Player already reached maxItem count	Player 1 has max item count of 2 Player1.pickItem(5) Player1.pickItem(4) Player1.pickItem(10)	Max item count reached, can't pick item.

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

Test "Pet" Interface

Test getName()	**Input**	**Expected output**
Get Pet Name	Pet.getName()	Suzzy the cat
Get Pet name when pet is not created	Pet.getName()	NullPointerException

Test getLocation()	**Input**	**Expected output**
Get location of a Pet	Pet.getLocation()	Room2
Get location if Pet is null.	Pet.getLocation()	NullPointerException

Test move()	**Input**	**Expected output**
Player tries to move pet to valid location	Pet.move(room5)	Success
Player tries to move pet to non-existent location	Pet.move(room109)	IllegalArgumentException

Controller Tests with Mock Model

Test "DisplayPlayerInfo" Command

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" Command

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" Command

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" Command

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" Command

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" Command

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 4 tries to pick Item6 when he still has eligibility to pick items, but Item5 is not present in the current space	Player2.PickItem(item6)	Item not present in space.
Player 4 tries to pick item when max item count is reached.	MaxItem count = 1. Player2.PickItem(item1) Player2.PickItem(item5)	Max item count reached.

Player tries to pick an item when he already has max number of items allowed to pick.	Player2.PickItem()	IllegalStateException

Test "LookAround" Command

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

Test "KillAttempt" Command

Test KillAttempt ()	Input	Expected output
Player 4 tries to attack Target when target is in same space.	Player4.killAttempt()	Target attacked.
Player 4 tries to attack Target when target is not in same space.	Player4.killAttempt()	Target not in same space. Failed to attack
Player 4 tries to attack Target when target is in same space an other player see's attack	Player4.killAttempt()	Kill witnessed, attack failed.

Test "MovePet" Command

Test MovePet()	Input	Expected output
Computer Tries to move to neighboring room	Player6.MovePet(Room3)	Pet moved to room 3
Player 6 tries to move pet to valid room when pet not in same room	Player6.MovePet(Room3)	Cannot move Pet
Player 6 tries to move pet to invalid room	Player6.MovePet(Room63)	IllegalArgumentException

Computer Controlled Player

Test MovePlayer()	Input	Expected output
Computer Player tries to move to a valid Neighbor	Comp.MovePlayer(Room5)	Computer moved from room4 to room5
Computer Player tries to move to a room which is not a neighbor	Comp.MovePlayer(Room16)	IllegalArgumentException

Test PickupItem()	**Input**	**Expected output**
Computer player tries to pick Item5 when he still has eligibility to pick items.	Comp.PickItem(item5)	Item picked and removed from space
Computer player tries to pick Item6 when he still has eligibility to pick items, but Item5 is not present in the current space	Comp.PickItem(item6)	Item not present in space.
Computer player tries to pick item when max item count is reached.	MaxItem count = 1. Comp.PickItem(item1) Comp.PickItem(item5)	Max item count reached.

Test KillAttempt()	**Input**	**Expected output**
Computer Player tries to attack Target when target is in same space.	Comp.KillAttempt()	Target attacked.
Computer Player tries to attack Target when target is not in same space.	Comp.KillAttempt()	Target not in same space. Failed to attack
Computer Player tries to attack Target when target is in same space an other player see's attack	Comp.KillAttempt()	Kill witnessed, attack failed.