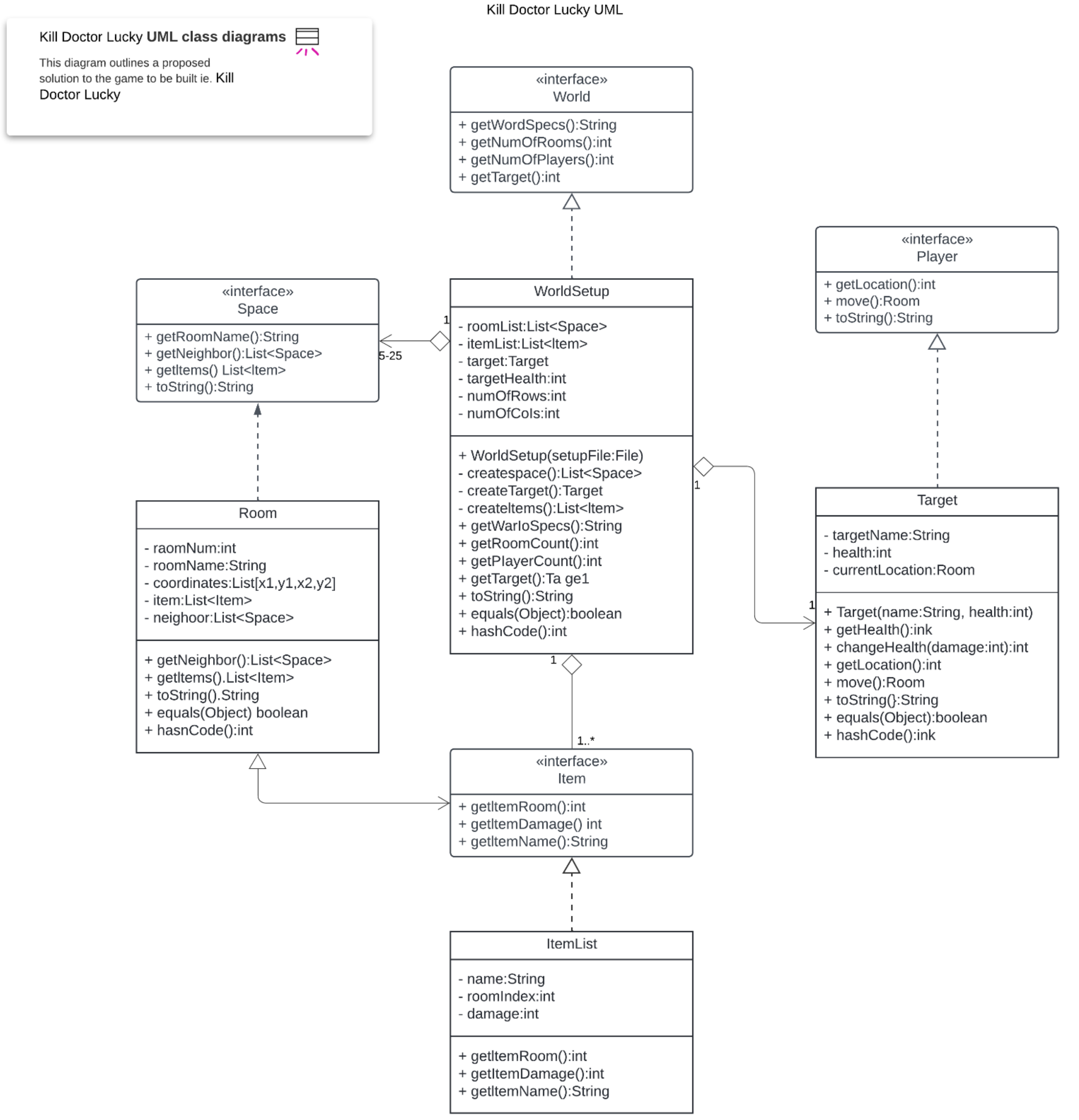
Carlos Semeho Edorh
edorh.c@northeastern.edu

Kill Doctor Lucky UML Class Diagram

Edorh Carlos Semho | edorh.c@northeastern.edu

Kill Doctor Lucky UML



Kill Doctor Lucky – Milestone 1 Test Plan

TEST THE **WORLD** INTERFACE

Test ***WorldSetup()*** constructor

- **Purpose**: Ensure the constructor parses the setup file correctly and handles missing files

Input	Expected output
Valid file path mansion.txt	Constructor parses through the file
Invalid file path `world.txt	Throws FileNotFoundException

Test **getWorldSpecs*()***

- **Purpose**: Ensure that the **getWorldSpecs()** method correctly returns file content.

Input	Expected output
Valid file mansion.txt	Returns file content as a string

Test **getRoomCount*()***

- **Purpose**: Ensure room count is correctly returned and handled when outside the valid range (5 <= roomCount <= 25).

Input	Expected output
Valid file `mansion.txt` with roomCount = 10	Returns 10
Valid file `mansion.txt` with roomCount = 4	Throws IllegalArgumentException
Valid file `mansion.txt` with roomCount = 26	Throws IllegalArgumentException
Valid file `mansion.txt` with roomCount = -2	Throws IllegalArgumentException

Test **getTarget*()***

- **Purpose**: Ensure the target from the file is correctly identified and handled.

Input	Expected output
Valid file `mansion.txt` with Target = "Doctor Lucky"	Returns `"Doctor Lucky"
Valid file `mansion.txt` with Target = ""	Throws IllegalArgumentException
Valid file `mansion.txt` with Target = `123`	Throws IllegalArgumentException

TEST THE “**PLAYER**” INTERFACE AND “**TARGET**” CLASS

Test **Target**(name: String, health: int) constructor

- **Purpose**: Ensure that the `Target` class constructor handles valid and invalid inputs.

Input	**Expected output**
Valid Target `"Doctor Lucky"` with health `50`	Target object created
Invalid Target	Throws IllegalArgumentException
Invalid Health `-5` or `0`	Throws IllegalArgumentException

Test **changeHealth**(damage: int)

- **Purpose**: Ensure the `changeHealth()` method properly reduces health and handles invalid inputs.

Input	**Expected output**
Valid Damage `3`	Returns `currentHealth - 3`
| Invalid Damage `-1`	Throws `IllegalArgumentException`
Damage equals current health (currentHealth = 3, Damage = 3	Target dies, Doctor Lucky is killed

Test **move()**

- **Purpose**: Ensure that the **move()** method properly moves the `Target` between rooms.

Input	**Expected output**
CurrentRoom = 1	Target moves to room 2
CurrentRoom = roomCount	Target moves to room 1

TEST THE **SPACE** INTERFACE

Test **getNeighbor()**

- **Purpose**: Ensure that **getNeighbor()** correctly returns neighbors for rooms.

Input	**Expected output**
Room 1	Returns [Room2, Room4, Room5]
Room 8	Returns [Room7, Room16]
Nonexistent Room 50	Throws NullPointerException

Test **getItems()**

- **Purpose**: Ensure **getItems()** properly returns the list of items in a room.

Input	**Expected output**
Room with 1 item	Returns [Item17]
Room with 0 items	Returns []
Room with >1 item	Returns [Item7, Item12]
Nonexistent Room 50	Throws NullPointerException

TEST THE "**ITEM**" INTERFACE

Test **getItemRoom()**

- **Purpose**: Ensure **getItemRoom()** correctly identifies the room an item belongs to.

Input	**Expected output**
Item2	Returns 4
Nonexistent Item40	Throws NullPointerException

Test **getItemDamage()**

- **Purpose**: Ensure **getItemDamage()** correctly returns the damage value for an item.

Input	**Expected output**
Item2	Returns 2
Nonexistent Item40	Throws NullPointerException

Test **getItemName()**

- **Purpose**: Ensure **getItemName()** correctly returns the name of an item.

Input	**Expected output**
Item2	Returns "Letter Opener"
Nonexistent Item40	Throws NullPointerException