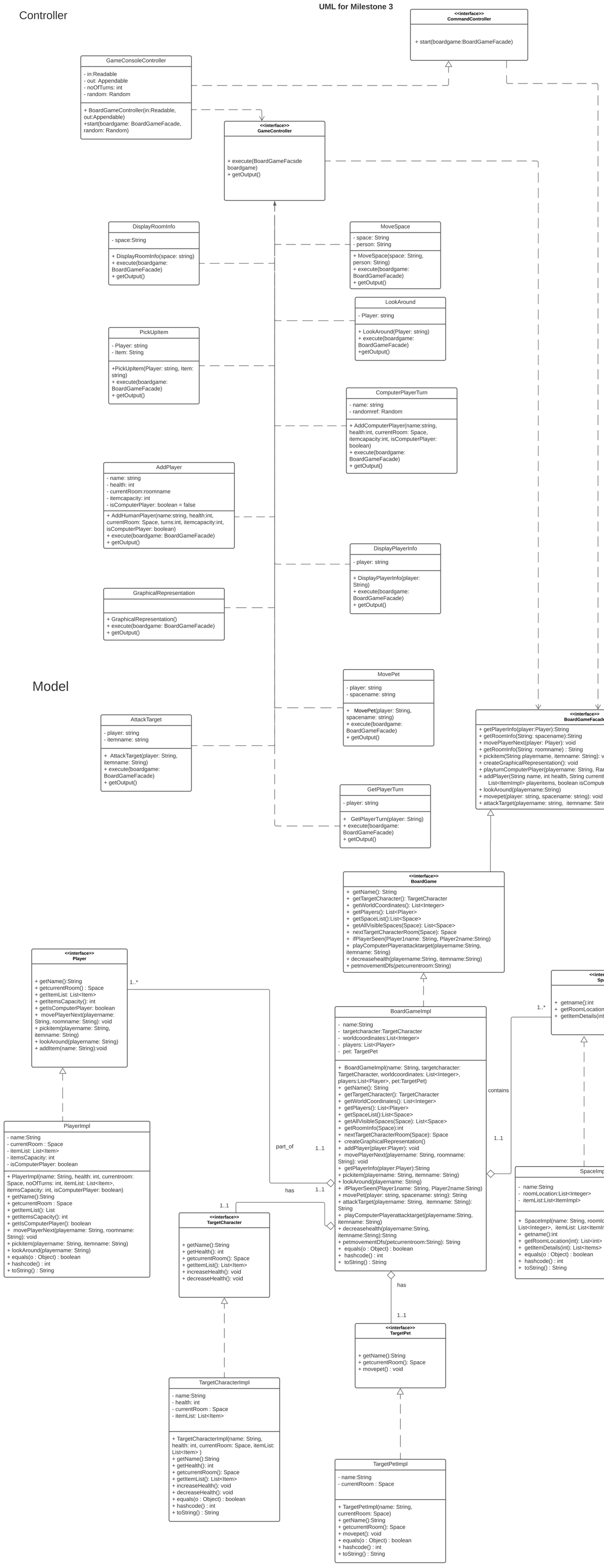
UML for Milestone 3

Controller



Model

Testing Plan for Milestone 3

Testing Model:

The test cases for the additional features of Milestone 3 are represented as a different color.

BoardGameImpl Test:

Testing	Input	Expected Output
Fetching Name	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).getName()	Doctor Lucky’s Mansion
Fetching World Coordinates	List<Integer> worldcoordinatesList = new ArrayList<Integer>(); worldcoordinatesList .add (36) worldcoordinatesList .add(30) new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).getWorldCoordinates()	36 ,30
Invalid World Coordinates	List<Integer> worldcoordinatesList = new ArrayList<Integer>(); worldcoordinatesList .add (-36) worldcoordinatesList	IllegalArgumentException

	.add(30) new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).getWorldCoordinates()	
Fetching all the spaces visible to a space:List<Space>() Passing the coordinates as a List List<int> temp = Arrays.asList(16, 21, 21, 28)	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).getAllVisibleSpaces(“Dining Hall”);	Armory | 2 | 22 19 23 26 Trophy Room | 18 | 10 21 15 26 Dining Hall | 3 | 12 11 21 20
Fetching all players of the game	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).getPlayersList()	Returns a list of players.
Adding a player to an object	Player player = new Player(“David”, -50, space, 3, 2, false) new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).addPlayer(playerobject)	Player gets added to the players lit
If exists player in the world	Player player = new	IllegalArgumentException

	Player(“David”, -50, space, 3, 2, false) new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacter(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).addPlayer(playerobject)	(Player already exists)
Displaying information about a Room	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList,playersList, new TargetPetImpl("Tommy", "Drawing Room")).getRoomInfo()	Space: Name: Billiard Room, Coordinates: 16, 21, 21 28, Items: Billiard Cue, Neighbours: Trophy Room, Dining Hall, Armory Players: David
Get player next movement room	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList,playersList, new TargetPetImpl("Tommy", "Drawing Room")).movePlayerNext(playerobject);	Space output should be one of the neighbouring space of the current space which the player is in. Space: Library
Get Target character movement’s next room	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList,playersList, new TargetPetImpl("Tommy", "Drawing Room")).nextTargetCharacterRoom(spaceobject)	Returns next room in the ordered list. Room: Armory

Get current space of the player	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList,playersList, new TargetPetImpl("Tommy", "Drawing Room")).getPlayerCurrentRoom(player: string);	If player does not exist to find out his location. Player doesn’t exist
Get current space of the player	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList,playersList, new TargetPetImpl("Tommy", "Drawing Room")).getPlayerCurrentRoom(player: string);	Space: Dining Room
Get Player Information	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList,playersList, new TargetPetImpl("Tommy", "Drawing Room")).getPlayerInfo(player object);	Player: David Space: Dining Room Item: Crepe Pan
Task to be performed by the player for his current turn	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList,playersList, new TargetPetImpl("Tommy", "Drawing Room")).playerTurn(1,playerobject);	If task 1 says the player has to move to the next space. Player David’s object must be updated to store current space name.
Display graphical representation of the world	new BoardGameImpl(“Doctor	We get the layout of the

map	Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).createGraphicalRepresentation()	world as output
Testing equals	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room").equals(new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room"))	True
Testing equals	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).equals(new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room"))	False

Testing toString()	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).toString()	Dr. Lucky’s Mansion Size: 36 30
Testing hashcode	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).hashcode()	Hashcode of the object
Testing Attack Target	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).attackTarget(“player 1”, “Crepe Pan”)	Target character health decreased by 2.
Testing Attack Target if item doesn’t exist on the player	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).attackTarget(“player 1”, “Crepe Pan”)	IllegalArgumentException
Testing Attack target if invalid item name present	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist),	IllegalArgumentException

	worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).attackTarget(“player 1”, “abcd”)	
Testing Attack target if hand used	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).attackTarget(“player 1”, “hand”) Passing item name - hand	Target character health decreased by 2.
Testing attack target if target character and player different room	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).attackTarget(“player 1”, “hand”) Passing item name - hand	IllegalStateException
Testing if item removed from the game when used to attack target	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).attackTarget(“player 1”, “Crepe Pan”) Passing item name - Crepe Pan	IllegalStateException

Testing move pet when player is in the same space as that of pet.	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).movepet(“player1”,” Laundry room”);	Laundry room
Testing move pet when space name is invalid	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).movepet(“player1”,” Laundry room”);	IllegalArgumentException
Testing move pet when player is not in the same space as that of pet.	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).movepet(“player1”,” Laundry room”);	Attic returns the next room in the depth first traversal
Testing move pet when player executes some other turn.	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).movepet(“player1”,” Attic”);	Garden returns the next element in the depth first traversal
Testing if game ends	new BoardGameImpl(“Doctor Lucky’s Mansion”, new	False

	TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).movepet(“player1”,” Laundry room”);	checks if target health is not zero
Testing if game begins	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).ifGameEnds();	True checks if target health is zero
Testing if pet exists in the neighbouring room and that space becomes invisible.	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).lookAround(“player1”) Current room of player1 - Living Room Current Room of Pet - Drawing Room	Player Current Room: Living Room, Spaces visible - Music Room, Players in current Space - players2, player3. Outputs all the neighbours except the one pet is in
Testing Look Around	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).lookAround(“player1”)	Player Current Room: Living Room, Spaces visible - Music Room, Drawing Room Items in the current space - Sofa Players in current Space - players2, player3. Neighbouring Spaces: Neighbour 1 - Music Room Items - Crepe Pan Players - player4

	Current room of player1 - Living Room Current Room of Pet - Attic (which is not a neighbour)	Neighbour 2 - Drawing Room Items - Divider Players - player5
Testing Look Around if target present in the same room	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).lookAround(“player1”) Current room of player1 - Living Room Current Room of Target - Living Room Current Room of Pet - Attic (which is not a neighbour)	Player Current Room: Living Room, Spaces visible - Music Room, Drawing Room Items in the current space - Sofa Players in current Space - players2, player3. Neighbouring Spaces: Neighbour 1 - Music Room Items - Crepe Pan Players - player4 Neighbour 2 - Drawing Room Items - Divider Players - player5
Testing Look Around if items present in the same room	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList,new TargetPetImpl("Tommy", "Drawing Room")).lookAround(“player1”) Current room of player1 - Living Room Current Room of Target - Living Room Current Room of Pet - Attic	Player Current Room: Living Room, Spaces visible - Music Room, Drawing Room Items in the current space - Sofa Players in current Space - players2, player3. Neighbouring Spaces: Neighbour 1 - Music Room Items - Crepe Pan Players - player4 Neighbour 2 - Drawing Room Items - Divider Players - player5

	(which is not a neighbour)	
Test playComputerPlayerTurn compulsorily hit the target when no player is looking around	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).lookAround(“comp1”) Current room of comp1 - Living Room Current Room of Target - Living Room Current Room of Pet - Attic (which is not a neighbour)	Target Health decreases by amount equivalent to the item used. If no item is used, then poke in the eye to reduce health by 1.
Test playComputerPlayerTurn executing another turn when player looking around	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).lookAround(“player1”) Current room of player1 - Living Room Current Room of Target - Living Room Current Room of Pet - Attic (which is not a neighbour)	Executes another random turn apart from hitting the target character
Test playComputerPlayerTurn hitting target character if no items on the player.	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList,	Target Health decreases by 1

	playersList, new TargetPetImpl("Tommy", "Drawing Room")).lookAround(“player1”) Current room of player1 - Living Room Current Room of Target - Living Room Current Room of Pet - Attic (which is not a neighbour)	
Testing if two players can see each other if they are in neighbours space.	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).ifPlayerSeen(“player1”,”player2”) Current space of player1 - Living Room Current space of player2- Drawing Room	True
Testing if two players can see each other if one player is in non neighbour space	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).ifPlayerSeen(“player1”,”player2”) Current space of player1 - Living Room Current space of player2- Playzone	False As they are not neighbours

Testing if two players can see each if they are in the same room	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).ifPlayerSeen(“player 1”,”player2”) Current space of player1 - Living Room Current space of player2- Living Room	True
Testing move player if space is neighbours	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).movePlayer(“player1”, “Drawing Room”) Current space of player1 - Living Room	Player successfully moved
Testing if target character moved to first room after the last room	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).movePlayer(“player1”, “Drawing Room”) Current Room of Target Character - Nursery	Drawing Room
Testing move players if space non neighbours	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr.	IllegalStateException

	Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).movePlayer(“player1”, “Laundry Room”) Current space of player1 - Living Room	
Testing pickitem if item exists in the space	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).pickItem (“player1”, “Divider”) Current space of player1 - Drawing Room	Item successfully picked
Testing pickitem if item not exists in the space	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).pickItem (“player1”, “Crepe Pan”) Current space of player1 - Drawing Room	IllegalStateException
Testing for picktItem if it is more than the capacity of the player	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersList, new TargetPetImpl("Tommy", "Drawing Room")).pickItem (“player1”, “Crepe Pan”)	IllegalStateException

	Current space of player1 - Drawing Room Items on the player - Divider Item Capacity - 1	
Testing if Computer Player winning the game.	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersLst, new TargetPetImpl("Tommy", "Drawing Room")).getGameWinner()	Get the computer player or human player name based on who kills the target character i.e; making the health zero.
Testing wandering pet movement	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersLst, new TargetPetImpl("Tommy", "Drawing Room")).petMovementDfs(currentpetroom)	Gets the next rooms using the DFS algorithm
Testing wandering pet backtracking movement	new BoardGameImpl(“Doctor Lucky’s Mansion”, new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist), worldcoordinatesList, playersLst, new TargetPetImpl("Tommy", "Drawing Room")).petMovementDfs(currentpetroom)	Gets the previous visited node in the path

TargetCharacterImpl Test

Testing	Input	Expected Value
Fetching the name of the target character	new TargetCharacterImpl("Dr. Lucky", 50, spaceobject, itemlist).getName()	Dr.Lucky
Fetching health of the target character	new TargetCharacterImpl("Dr. Lucky", 50, spaceobject, itemlist).getHealth()	50
Health cannot be negative	new TargetCharacterImpl("Dr. Lucky", -50, spaceobject, itemlist).getHealth()	IllegalArgumentException
Get Items of the targetcharacter	new TargetCharacterImpl("Dr. Lucky", 50, spaceobject, itemlist).getItemList()	Fetches the list of items the target character can hold
Get the target character's current room	new TargetCharacterImpl("Dr. Lucky", 50, spaceobject, itemlist).getCurrentRoom()	Room: Armory
Increase the health of the target character	new TargetCharacterImpl("Dr. Lucky", 50, spaceobject, itemlist).increasehealth().getHealth()	51
Decrease the health of the target character	new TargetCharacterImpl("Dr. Lucky", 50, spaceobject, itemlist).decreasehealth().getHealth()	49
Decrease the health of the target character below 0	new TargetCharacterImpl("Dr. Lucky", 0, spaceobject, itemlist).decreasehealth().getHealth()	IllegalArgumentException
Testing equals	new TargetCharacterImpl("Dr. Lucky", 50, spaceobject, itemlist).equals(new TargetCharacter("Dr. Lucky", 50, spaceobject, itemlist))	True
Testing equals	new TargetCharacterImpl("Dr. Lucky", 50, spaceobject, itemlist).equals(new	False

	TargetCharacter(“Dr. Lucky”, 51, spaceobject, itemlist))	
Testing toString	new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist).toString()	Dr. Lucky Health: 50
Testing Hashcode	new TargetCharacterImpl(“Dr. Lucky”, 50, spaceobject, itemlist).hashcode()	Hashcode of the object

SpaceImpl Test:

Testing	**Input**	**Expected Value**
Fetching the name of the space	new SpaceImpl(“Billiard Room”, Arrays.asList(2, 3, 9, 10), itemlist).getName()	Billiard Room
Fetching Room Location details	new SpaceImpl(“Billiard Room”, Arrays.asList(2, 3, 9, 10), itemlist).getRoomLocation()	2 3 9 10
Get Items in a room	new SpaceImpl(“Billiard Room”, Arrays.asList(2, 3, 9, 10), itemlist).getItemList()	Crepe Pan 8 Revolver 9
Testing equals	new SpaceImpl(“Billiard Room”, Arrays.asList(2, 3, 9, 10), itemlist).equals(new SpaceImpl(“Billiard Room”, Arrays.asList(2, 3, 9, 10), itemlist))	True
Testing equals	new SpaceImpl(“Billiard Room”, Arrays.asList(2, 3, 9, 10), itemlist).equals(new SpaceImpl(“Billiard Room”, Arrays.asList(2, 3, 9, 10), itemlist))	False
Invalid Room Location	new SpaceImpl(“Billiard Room”, Arrays.asList(2, -3, 9, 10), itemlist).getRoomLocation()	IllegalArgumentException

Testing toString()	new SpaceImpl(“Billiard Room”, Arrays.asList(2, -3, 9, 10), itemlist).toString()	Carriage House | 2 | 28 0 35 5 Items: Item: Name: Crepe Pan, Damage: 3, Room: Kitchen
Testing Hashcode	new SpaceImpl(“Billiard Room”, Arrays.asList(2, -3, 9, 10), itemlist).hashcode()	Hashcode of the object

ItemImpl Test:

Testing	**Input**	**Expected Value**
Fetching the name of the space	new ItemImpl(“Crepe Pan”, 1).getName()	Crepe Pan
Fetching Item Damage	new ItemImpl(“Crepe Pan”, 2).getItemDamage()	2
Invalid Item Damage	new ItemImpl(“Crepe Pan”, -3).getItemDamage()	IllegalArgumentException
Testing equals	new ItemImpl(“Crepe Pan”, 3).equals(new ItemImpl(“Crepe Pan”, 3))	True
Testing equals	new ItemImpl(“Crepe Pan”, 3).equals(new ItemImpl(“Crepe Pan”, 4))	False
Testing toString()	new ItemImpl(“Crepe Pan”, 3).toString()	Item: Crepe Pan, Damage: 3
Testing Hashcode	new ItemImpl(“Crepe Pan”, 3).hashcode()	Hashcode of the object

PlayerImpl Test:

Testing	**Input**	**Expected Value**

Fetching the name of the space	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).getName()	James
Fetching the current room	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).getcurrentRoom().getName()	Armory
Fetching number of turns	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).getNoOfTurns()	5
Invalid number of turns	new PlayerImpl(“James”, 70, spaceobject, -5, itemlist, 3, true).getNoOfTurns()	IllegalArgumentException
Get Item Capacity	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).getItemCapacity()	3
Invalid Item Capacity	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, -3, true).getItemCapacity()	IllegalArgumentException
Testing Item list	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).getItemList()	Returns the list of items the player has
Testing human player	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).getIsComputerPlayer()	False
Testing Computer player	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).getIsComputerPlayer()	True
Testing next movement of player	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).pickNextmove(playerobject);	Returns one of the neighbouring spaces name.
Testing next movement of player	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).pickNextmove(playero	If no neighbours, remains in the same space.

	bject);	
Testing add item to player	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).addItem();	Item exists in the room then item gets added to the player
Testing add item to player	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).addItem();	IllegalStateException (because if the itemcapacity of the player is equal to no of items he has.
Testing equals	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).equals(new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true))	True
Testing equals	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).equals(new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true))	False
Testing toString()	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).toString()	Player: James, Room: Billiard Room, Player Type: Human
Testing Hashcode	new PlayerImpl(“James”, 70, spaceobject, 5, itemlist, 3, true).hashcode()	Hashcode of the object

TargetPetImpl Test:

Testing	**Input**	**Expected Value**
Fetching the name of the pet	new TargetPetImpl (“Tommy”, “Drawing Room”).getName()	Tommy
Fetching current room of the pet	new TargetPetImpl (“Tommy”, “Drawing Room”).getCurrentRoom()	2
Invalid Current room	new TargetPetImpl (“Tommy”, null).getCurrentRoom()	IllegalArgumentException
Testing equals	new TargetPetImpl (“Tommy”, “Drawing Room”).equals(new	True

	TargetPetImpl (“Tommy”, “Drawing Room”))	
Testing equals	new TargetPetImpl (“Tommy”, “Drawing Room”).equals(new TargetPetImpl (“Tommy”, “Music Room”))	False
Testing toString()	new TargetPetImpl (“Tommy”, “Drawing Room”).toString()	Pet - Name: Tommy, Current Room: Drawing Room
Testing Hashcode	new TargetPetImpl (“Tommy”, “Drawing Room”).hashcode()	Hashcode of the object

Testing Controller:

Testing Readable and Appendable:

Testing	**Input**	**Expected Output**
Passing null for Readable	Readable in = null; BoardGameContructor(Readable in, Appendable out)	IllegalArgumentException
Passing null for Appendable	Appendable out = null; new BoardGameContructor(Readable in, Appendable out)	IllegalArgumentException
Passing valid values	new BoardGameContructor(Readable in, Appendable out)	Object for the controller needs to be created

DisplayRoomInfo:

Testing	**Input**	**Expected Output**
Passing null	null	IllegalArgumentException
Passing an non existent Space name.	DisplayRoomInfo(“ABC”)	Space doesn’t exist
Passing a valid space name	DisplayRoomInfo(“Armory”) execute(boardgame: BoardGame)	GetRoomInfo method gets called and Room information including its name, neighours and items present would be popping up.

MoveSpace:

Testing	**Input**	**Expected Output**
Passing null	null	IllegalArgumentException
Passing invalid space	MoveSpace(“ABC”, “James”)	Space not found
Passing non existent player	MoveSpace(“Armory”, “DEF”)	Player not found
Passing a valid space and player name	MoveSpace(“Armory”, “James”) execute(boardgame:BoardGame)	Output the data of any of the neighbouring spaces.

LookAround:

Testing	**Input**	**Expected Output**
Passing null	null	IllegalArgumentException
Passing invalid player	LookAround(“ABC”)	Player not found
Passing a valid player name	LookAround(“James”) execute(boardgame:BoardGame)	Displays a list of neighbours of a space and the players in the current space.

PickUpItem:

Testing	**Input**	**Expected Output**
Passing null	null	IllegalArgumentException

Passing invalid item	PickUpItem(“test”, “James”)	Test is not an item in the space
Passing invalid player name	PickUpItem(“Revolver”, “ABC”)	Displays a list of neighbours of a space.
Passing a item but non existent in the space	PickUpItem(“Crepe pan”, “James”)	Crepe pan doesn’t belong to Space the player is in.
Item holding capacity of the player lesser than	PickUpItem(“Crepe pan”, “James”)	Could not have the player hold another item as it has reached its capacity.
Passing a valid player and item name	PickUpItem(“Revolver”, “James”) execute(boardgame:BoardGame)	Item picked up by the player. And gets added to

AddPlayer:

Testing	**Input**	**Expected Output**
Passing null	null	IllegalArgumentException
Passing invalid item	Space space = new Space(“Armory”,Arrays.asList(36,30)); AddHumanPlayer(“David”, 50, space, 3, 2, false)	Item not present
Passing item name not pertaining to the given space	AddHumanPlayer(“David”, 50, space, 3, 2, false)	Item not present in the given space.
Health cannot be negative	AddHumanPlayer(“David”, -50, space, 3, 2, false)	IllegalArgumentException
Item capacity of the player cannot be negative	AddHumanPlayer(“David”, -50, space, 3, -2, false)	IllegalArgumentException
Turns cannot be negative	AddHumanPlayer(“David”, -50, space, -3, 2, false)	IllegalArgumentException
Passing a valid player, valid Item, valid capacity and valid health	AddHumanPlayer(“David”, -50, space, 3, 2, false) execute(boardgame:BoardGame)	Player added to the Game.

ComputerPlayerTurn:

Testing	**Input**	**Expected Output**
Passing null	null	IllegalArgumentException
Passing random	Random r = new Random() ComputerPlayerTurn(“John”, null)	IllegalArgumentException
Passing null value	Random r = new Random() ComputerPlayerTurn(null, r)	IllegalArgumentException
Passing a valid player, valid Item, valid capacity and valid health	Random r = new Random() ComputerPlayerTurn() execute(boardgame:BoardGame)	Player added to the Game.

DisplayPlayerInfo:

Testing	**Input**	**Expected Output**
Passing null	null	IllegalArgumentException
Passing an non existent Player name.	DisplayPlayerInfo(“XYZ”)	Could not find player
Passing a valid Player name	DisplayPlayerInfo(“James”) execute(boardgame: BoardGame)	GetPlayerInfo method gets called and Player information including room in which he is present and items present on him is displayed as output.

GraphicalRepresentation:

Testing	**Input**	**Expected Output**
Valid values	–	createGraphicalRepresentation( ) method gets called.

MovePet:

Testing	Input	Expected Output
Passing null	null	IllegalArgumentException
Passing an invalid Space name	MovePet(“player1”, “abcd”)	Space doesn’t exist
Passing an invalid player name	MovePet(null, “Music Room”)	Player doesn’t exist
Passing a valid Space and Player name	MovePet(“player1”, “Music Room”)	Player moves the pet to a location of his choice which we pass as an argument.
Passing a valid Space and player name	MovePet(“player1”, “Music Room”)	

AttackTarget:

Testing	Input	Expected Output
Passing null	null	IllegalArgumentException
Passing an invalid player name	AttackTarget(player: String, itemname: String)	Not a valid player
Passing an invalid item name	AttackTarget(“player1”, “abc”)	Item doesn’t exist on the player
Passing a valid player and item name	AttackTarget(“player1”, “Crepe pan”)	Player attacks the target character and decreases its health by an amount equivalent to the damage.
Passing a valid player and item name and testing if player kills the target character	AttackTarget(“player1”, “Crepe pan”)	Target Character killed successfully
Passing a valid player and item name and testing if turns over	AttackTarget(“player1”, “Crepe pan”)	Target character escaped as the number of turns has been exhausted

GetPlayerTurn:

Testing	Input	Expected Output
Passing null	null	IllegalArgumentException

Passing an invalid player name	GetPlayerTurn(“abc”);	Not a valid player
Passing a valid player	GetPlayerTurn(“player1”);	Gets the next player in the order