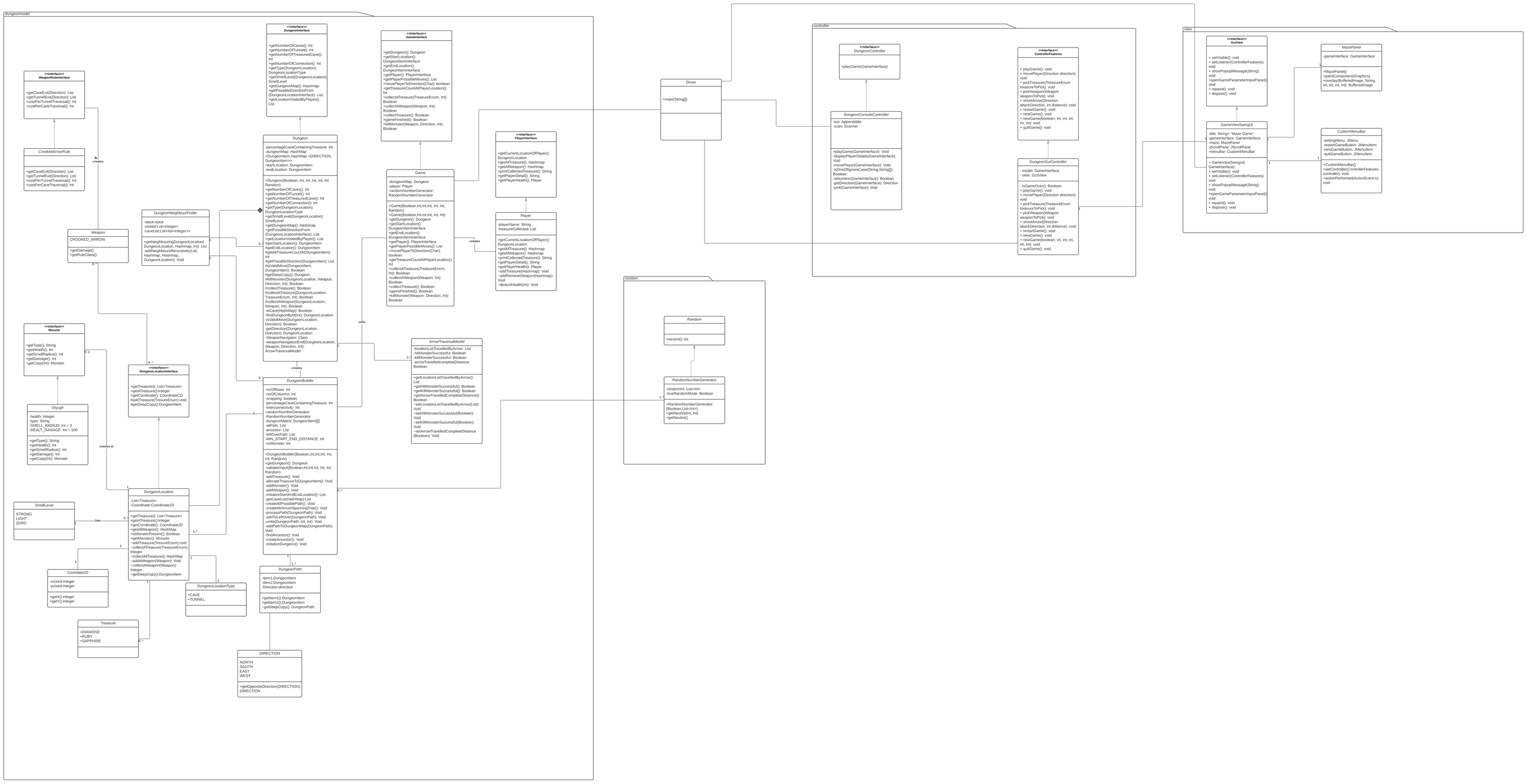
Project: Dungeon Game



Testing Plan

Sr No	Class Name	Method Name	Test Description
1	Game Builder	setPlayerName()	1. Validates for NULL value.
2		setStartLocation()	1. Validate for NULL values 2. Checks if the location is within the dungeon range
3		setEndLocation()	1. Validate for NULL values 2. Checks if the location is within the dungeon range 3. Check if the end_location is 5 distance away from the start location.
4		getDungeonBuilder()	1. Check every item in the object. It should be the same as what we set above.
5	Game	getPlayerName()	1. The name of player should be what we have set.
6		getPlayer()	1. Will be used to test the entire Player object.
7		getDungeon()	1. Will be used to test the entire Dungeon.
8		movePlayer()	1. Will be check by getting the player location, before and after the call. 2. Check if player can directly travel to the given location from his present location.
9	DungeonBuilder	setNoOfRows()	1. Test for non-negative values. 2. The value will be validated by a getter. 3. Test for minimum possible value(value cannot be 0).
10		setNoOfColumns()	1. Test for non-negative values. 2. The value will be validated by a getter. 3. Test for minimum possible value(value cannot be 0).
11		setWrapping()	1. Will be validated by a getter.
12		setTreasurePercentage()	1. Will be validated by a getter. 2. Will be validated by counting the number of dungen cells containing treasure out of total dungeon cells.
13		setInterconnectivityDegree()	1. Will be validated using toString() method.
14		setRandom()	1. Will be validated for NULL value.
15	Dungeon	getNoOfRows()	Used to validate point 9
16		getNoColumns()	Used to validate point 10
17		isWrappingDungeon()	Used to validate point 11
18		getTreasurePercentage()	Used to validate point 12
19		getDegreeOfInterconnectivity()	Used to validate point 13
20		validateMove(CoordinateLocation, CoordinateLocation)	1. Validate for NULL input 2. Validate that both location exists in the dungeon 3. Validate that the from_location allows direct travel to to_location 4. If all three are successful, return true. Else return false.
21		canAddTreasure(DungeonItem)	1. Check for NULL values. 2. Check if location is cave or tunnel.
22		addTreasure(DungeonItem, Treasure)	1. Validate for NULL values. 2. Check if treasure can be added to the given location. 3. Add tresaure to the location. If successful, return true. Else return false or Exception.
23		buildDungeonMap()	1. Will be validated by the toStirng method.
24		getDungeonType()	1. Check if type is 1 of the ENUM type. 2. Validate using toString() method.
25		createDungeonUsingKruskal()	1. Check if every node is connected to every other node after calling this function.
26	Player	connectLeftOverNode(Int)	1. Validate when the interconnectivity is more than 0. 2. Validate that the cycles ar created depending on the interconnectivity.
27		getPlayerName()	1. Check will toString() method on player object.
28		getTreasureCollected()	1. Check before and after collecting treasure with toString() method.
29		getPlayer()	1. Get the player object and validate other details like name and treasure collected. 2. Check if a new copy is passed by trying to mutate the object.
30	RandomNumberGenerator	RandomNumberGenerator (Boolean,List<Int>)	1. Check for NULL values.
31		getNextInt(Int, Int)	1. Check if the random number is between the specified range. 2. Check if the number is the one passed in constructor in pseudorandom mode.
32		getNextInt()	1. Check if the number is the one passed in constructor in pseudorandom mode.
33	DungeonItem	getTreasure()	1. Validate if the list of treasure returned is valid for that location.

New Tests added

Sr No	Class Name	Method Name	Test Description
34	Dungeon	getDungeon()	1. Used to test if correct number of arrows are added in the dungeon. 2. Used to test if correct number of monster(Otyugh) are added in the dungeon.
35		getType()	1. Check if the method returns CAVE when a cave is passed as input. 2. Check if the method returns TUNNEL when a tunnel is passed as input. 3. Test NULL handling.
37	Game	isGameOver()	1. Check if method returns true when player reached the end cave and slayed the monster there. 2. Check if method returns true when the player is killed by the monster. 3. Check if method returns false in all other condition.
38		getStartLocation()	1. Used to test if the start location is a cave. 2. Used to test that there is never a monster at the start cave.
39		getEndLocation()	1. Used to test if the end location is a cave. 2. Used to test that there is always a monster at the end cave.
40	Monster	getDamage()	1. Will be used to test the damage done by the monster. 2. This method will be tested by getting the player's health after monster's attack.
41		getSmellRadius()	1. Test if the monster has correct smell radius.
42		getHealth()	1. Test if the monster is at full health when the game starts. 2. Test that the player loses health when it is hit by an arrow.
43			
44	Player	collectATreasure()	1. Validate that the treasure selected is a valid treasure. 2. Test that the amout being picked is actually present at the player's current location. 3. Test if the player was collected correctly.
45		getAllWeapon()	1. Test if all weapons are picked correctly. 2. Test that the player has 3 arrows at the start of the game. 3. Used to test if the player's weapon picking ability works properly.
46		addRemoveWeapon()	1. Test if the weapon is picked by the player. 2. Test if the weapon is used by the player.
47			
48	DungeonILocation	getTreasureCountOf()	1. Validate for NULL input 2. Validate if correct count of the given treasure is returned.
49		addWeapon()	1. Check for NULL values.
50		addMonster()	1. Validate for NULL values. 2. getMonster() will be used to validate if the addMonster() method worked properly.
51		getCoordinate()	1. Used to check if the current coordinate for a dungeon location is returned.