

	KillDoctorLuckyConsoleControlle	er
Test Case	Input	Expected Outcome
	KillDoctorLuckyConsoleController controller	
	= new	
	KillDoctorLuckyConsoleController(new	
	StringReader("input"), new StringBuilder(),	No Exception is thrown, and the controller object
Constructor - Valid Input	new RandomGeneratorImpl());	is created.
	KillDoctorLuckyConsoleController controller	
	= new	
	KillDoctorLuckyConsoleController(null, new	
Constructor - Null Readable or	StringBuilder(), new	
Appendable	RandomGeneratorImpl());	IllegalArgumentException is thrown.
	Create a test environment with necessary	Executes the game flow without errors,
playGame - Valid Game Flow	input.	displaying prompts and game actions.
playGame - Invalid Specification	Create a test environment with a non-	Controller displays an error message and
File	existent specification file.	prompts for a valid file.
	Create a test environment and simulate	Controller correctly processes the player's move
playGame - Player Move Choice	player choice.	choice.
playGame - Player I tem Pickup	Create a test environment and simulate	Controller correctly processes the player's item
Choice	player choice.	pickup choice.
playGame - Player Look Around	Create a test environment and simulate	Controller correctly processes the player's "look
Choice	player choice.	around" choice.
		Returns a string containing the joined names
joinNames - Valid List of Objects	A list of objects with names.	separated by commas.
joinNames - Empty List of Objects	An empty list of objects.	Returns an empty string.
10 1 1 1 1 1 1	controller.appendCommand("Test	The message is appended to the Appendable
appendCommand - Valid Input	Message");	without errors.
	controller.appendCommand("Test	III IC4-4- Francisco i- Alexandro della companya
a none ad Company and I O Even which	Message"); with Appendable that throws	IllegalStateException is thrown with a wrapped
appendCommand - IOException	an IO Exception.	IOException.
Took Coco	CreatePlayer	Evanstad Outcome
Test Case	Input CreatePlayer createPlayer = new	Expected Outcome
	CreatePlayer(new	No Everation is thrown and the chiest is
Constructor Valid Input	RandomGeneratorImpl());	No Exception is thrown, and the object is
Constructor - Valid Input	CreatePlayer createPlayer = new	created. No Exception is thrown, and the object is
Constructor - Name and Space		created with the specified parameters.
Index	CreatePlayer("Player1", 1, 5); Create a KillDoctorLucky model, execute	Auto-Player is created with random values for
execute - Auto-Player Creation	with name = null.	name, currentSpaceIndex, and maxItems.
execute - Auto-riayer Creation	Create a KillDoctorLucky model, execute	Player is created with the specified name,
execute - Player Creation	with valid name and parameters.	currentSpaceIndex, and maxItems.
execute - Flayer Creation	MaxTurn	currents pacernues, and maxitems.
Tort Caro		Evnocted Outcome
Test Case	Input MaxTurn maxTurnCommand = new	Expected Outcome
Constructor Valid Incut		No Exception is thrown, and the object is
Constructor - Valid Input	MaxTurn(10);	created.

	Create a KillDoctorLucky model, execute	
execute - Set Max Turn	with maxTurn = 10.	The maximum turn is set to 10 in the model.
execute - Null Model	Execute the command with a null model.	IllegalArgumentException is thrown.
execute Namivious	Parse	megan agament.
Test Case	Input	Expected Outcome
rest case	Parse parseCommand = new	No Exception is thrown, and the object is
Constructor - Valid Input	Parse("specification.txt");	created.
Constructor Valid Impat	Parse parseCommand = new	orcated.
Constructor - File Not Found	Parse("nonexistent.txt");	FileNotFoundException is thrown.
Constructor The Notification	Create a KillDoctorLucky model and	Thereta dandexception is thrown.
	execute the command with a valid	The model is populated with mansion details,
execute - Parse Specification File	specification file.	target character, spaces, and items.
execute - Null Model	Execute the command with a null model.	IllegalArgumentException is thrown.
execute - Null Widdel	Create a KillDoctorLucky model and	meganargumentexception is thrown.
	execute the command with a specification	
execute - Malformed File	file that has incorrect formatting.	IllegalStateException is thrown.
CACCACC WAITONITICAT IIC	The that has meaned formatting.	Model populated with mansion details, target
Valid Specification File	Valid file with correct formatting	character, spaces, and items.
Invalid Specification File -	valid the with confect formatting	IllegalStateException is thrown due to missing
Missing Mansion Information	File missing mansion information	mansion information.
Invalid Specification File -		IllegalStateException is thrown due to incorrect
Incorrect Format	File with incorrect formatting	format.
		IllegalArgumentException is thrown since the
Null Model	Execute command with a null model	model cannot be null.
		Model populated with mansion details, target
Maximum Items Section Missing	File missing the section for items	character, and spaces, but no items added.
Spaces Without Neighbors	File with spaces but without neighbors	Spaces created, but no neighbors defined.
	·	Target character created with the name, index
Target Character Name Only	File with target character name only	set to 0 by default.
		Target character created with the index, name
Target Character Index Only	File with target character index only	set to null by default.
-	-	Model remains unchanged as there is no data
Empty Specification File	An empty specification file	in the file
	RandomGenerator	
Test Case	Input	Expected Outcome
Constructor - Real Random	RandomGenerator randomGen = new	No Exception is thrown, and the object is
Generator	RandomGenerator();	created with a real random generator.
Constructor - Mocked Random	RandomGenerator randomGen = new	No Exception is thrown, and the object is
Generator	RandomGenerator(1, 2, 3);	created with mocked values.
	Generate random integers using a real	Random integers within the specified bound are
nextInt - Real Random Generator	random generator.	generated.
nextInt - Mocked Random	Generate integers using a mocked	Integers from the mocked values are
Generator	generator.	generated in the same order.
nextInt - No More Mocked	Attempt to generate values when there	
Values	are no more mocked values.	IllegalStateException is thrown.
DoctorLuckyModel		
Test Case	Input	Expected Outcome

	DoctorLuckyModel doctorLucky = new	No Exception is thrown, and the object is
Constructor Valid Insut		
Constructor - Valid Input	DoctorLuckyModel("Lucky", 100);	created with the specified name and health.
Construction loss slid Hookk	DoctorLuckyModel doctorLucky = new	IllegalArgumentException is thrown due to an
Constructor - Invalid Health	DoctorLuckyModel("Lucky", -10);	invalid health value.
	Create a DoctorLuckyModel with a valid	
getHealth - Valid Health	health value and call getHealth.	Returns the valid health value.
	Create a DoctorLuckyModel with a valid	Returns a string representation of the
toString - Valid Health	health value and call toString.	DoctorLucky object with the name and health.
	Create two DoctorLuckyModel objects with	
	the same name and compare their hash	
hashCode - Same Name	codes.	Hash codes of both objects should be equal.
	Create two DoctorLuckyModel objects with	
	the same name and compare them using	The equals method should return true as the
equals - Equal Objects	the equals method.	objects have the same name.
	Create two DoctorLuckyModel objects with	
	different names and compare them using	The equals method should return false as the
equals - Not Equal Objects	the equals method.	objects have different names.
	Compare a DoctorLuckyModel object with	
equals - Comparison with	an object of a different class using the	The equals method should return false as the
Different Class	equals method.	classes are different.
	PlayerModel	
Test Case	Input	Expected Outcome
		No Exception is thrown, and the object is
	PlayerModel player = new	created with the specified name,
Constructor - Valid Input	PlayerModel("Alice", 1, 5);	currentSpaceIndex, and maxItems.
setItem - Add Item Below Max	Create a PlayerModel with maxItems = 5,	Items are added successfully, and the items list
Limit	add 3 items using setItem.	contains 3 items.
		The 5th item addition should throw an
	Create a PlayerModel with maxItems = 5,	IllegalStateException since the maxItems limit
setItem - Add Item at Max Limit	add 5 items using setItem.	is reached.
	Create a PlayerModel with items added,	
getItems - Get Items	call getItems.	Returns the list of items added to the player.
<u> </u>		Returns a string representation of the Player
	Create a PlayerModel with valid data and	object with the name, maxItems, and
toString - Valid Data	call toString.	currentSpaceIndex.
	Create two PlayerModel objects with the	
hashCode - Same Name	same name and compare their hash codes.	Hash codes of both objects should be equal.
	Create two PlayerModel objects with the	The state of the s
	same name and compare them using the	The equals method should return true as the
equals - Equal Objects	equals method.	objects have the same name.
equality Equal Objects	Create two PlayerModel objects with	osjecta nave the same name.
	different names and compare them using	The equals method should return false as the
equals - Not Equal Objects	the equals method.	objects have different names.
equals - NOL Equal Objects	Compare a PlayerModel object with an	objects have unferent halfles.
oquals Companionth		The equals method should return false as the
equals - Comparison with	object of a different class using the equals	The equals method should return false as the
Different Class	method.	classes are different.
	ItemModel	

Test Case	Input	Expected Outcome
		No Exception is thrown, and the object is
	ItemModel item = new	created with the specified name, position, and
Constructor - Valid Input	ItemModel("Sword", 1, 10);	damage.
getName - Get Name	Create an ItemModel and call getName.	Returns the name of the item.
getPosition - Get Position	Create an ItemModel and call getPosition.	Returns the position of the item.
getDamage - Get Damage	Create an ItemModel and call getDamage.	Returns the damage value of the item.
	Create an ItemModel with valid data and	Returns a string representation of the Item
toString - Valid Data	call toString.	object with the name, position, and damage.
	Create two ItemModel objects with the	
hashCode - Same Name	same name and compare their hash codes.	Hash codes of both objects should be equal.
	Create two ItemModel objects with the	
	same name and compare them using the	The equals method should return true as the
equals - Equal Objects	equals method.	objects have the same name.
	Create two ItemModel objects with	
	different names and compare them using	The equals method should return false as the
equals - Not Equal Objects	the equals method.	objects have different names.
	Compare an ItemModel object with an	
equals - Comparison with	object of a different class using the equals	The equals method should return false as the
Different Class	method.	classes are different.
	KillDoctorLuckyModel	
Test Case	Input	Expected Outcome
Constructor - Valid Mansion	Create a KillDoctorLuckyModel object with	The mansion and doctorLucky are correctly
Specification	valid mansion specification.	initialized, and no exceptions are thrown.
	Create a KillDoctorLuckyModel object and	Returns the mansion object that was set using
getMansion - Get Mansion	call getMansion.	the constructor.
	Create a KillDoctorLuckyModel object and	
	call setDoctorLucky to set the doctor's	The doctorLucky object is created with the
setDoctorLucky - Set DoctorLucky	name and health.	specified name and health.
		The player object is added to the players list
	Create a KillDoctorLuckyModel object and	with the specified name, space index, and max
setPlayer - Set Player	call setPlayer to add a player.	items.
	Create a KillDoctorLuckyModel object with	
getMansionInfo - Get Mansion	a predefined mansion and call	Returns a string containing information about
Info	getMansionInfo.	the mansion.
	Create a KillDoctorLuckyModel object with	Returns a string containing information about
getPlayersInfo - Get Players Info	predefined players and call getPlayersInfo.	the players.
, , , , , , , , , , , , , , , , , , , ,	Create a KillDoctorLuckyModel object and	. ,
getMaxTurn - Get Max Turn	call getMaxTurn.	Returns the maximum turn value that was set.
0	Create a KillDoctorLuckyModel object with	The state of the s
	predefined players and call	
getPlayerByTurn - Get Player By	getPlayerByTurn with various turn	Returns the player corresponding to the given
Turn	numbers.	turn number.
TUITI	numbers.	tarri namber.

	Create a KillDoctorLuckyModel object with	
getDoctorLucky - Get	a predefined doctorLucky and call	
DoctorLucky	getDoctorLucky.	Returns the doctorLucky object that was set.
	Create a KillDoctorLuckyModel object and	
	call setMansion to set the mansion name,	The mansion object is created with the specified
setMansion - Set Mansion	height, and width.	name, height, and width.
	MansionModel	
Test Case	Input	Expected Outcome
	Create a MansionModel object with valid	The mansion object is correctly initialized with
	mansion specifications (name, height, and	the specified name, height, and width. No
Constructor - Valid Mansion	width).	exceptions are thrown.
getSpacesNum - Get Spaces	Create a MansionModel object and call	Returns the number of spaces that was set
Number	getSpacesNum.	using the setSpacesNum method.
getItemsNum - Get Items	Create a MansionModel object and call	Returns the number of items that was set using
Number	getItemsNum.	the setItemsNum method.
	Create a MansionModel object and call	
setSpacesNum - Set Spaces	setSpacesNum to set the number of	The spacesNum is correctly set to the specified
Number	spaces.	value.
setItemsNum - Set Items	Create a MansionModel object and call	The itemsNum is correctly set to the specified
Number	setItemsNum to set the number of items.	value.
	Create a MansionModel object and call	A space is added to the list of spaces with the
addSpace - Add Space to	addSpace to add a space to the mansion	specified index, name, and points. No
Mansion	with valid index, name, and points.	exceptions are thrown.
	Create a MansionModel object with	Returns a list of spaces that were added using
getSpaces - Get Spaces List	predefined spaces and call getSpaces.	the addSpace method.
	Create a MansionModel object with	
	predefined name, height, width,	Returns a string representation of the mansion
	spacesNum, and itemsNum and call	including its name, height, width, spacesNum,
toString - Convert to String	toString.	and itemsNum.
	SpaceModel	
Test Case	Input	Expected Outcome
Constructor Test	SpaceModel(index, name, [], points, [], [])	Space is created with provided parameters.
		SpaceModel object created with builder
build Test	Set properties using builder and build	properties.
		true if points represent a neighbor, false
isNeighbor Test	space.isNeighbor(validNeighborPoints)	otherwise.
getName Test	space.getName()	Returns the name of the space.
getItems Test	space.getItems()	Returns an unmodifiable list of items.
getPoints Test	space.getPoints()	Returns a clone of the points array.
getNeighbors Test	space.getNeighbors()	Returns an unmodifiable list of neighbors.
	space.addItem(validItemName,	
addItem Test	validPosition, validDamage)	Item is added to the list of items.
addNeighbor Test	space.addNeighbor(newSpace)	New space added to list of neighbors if valid.
addPlayer Test	space.addPlayer(validPlayer)	Player is added to the list of players.
toString Test	space.toString()	Returns a string representation of the space.
hashCode Test	space.hashCode()	Hash code based on name and points.
.	*	

	space.equals(sameNameSamePointsSpac	
equals Test	e)	true if name and points match, false otherwise.
getIndex Test	space.getIndex()	Returns the index of the space.