

KillDoctorLuckyConsoleController			
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.	
In the second se	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 Item 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.	
playGame - Player Pet Move	Create a test environment and simulate	Controller correctly processes the player's move	
Choice	player choice.	pet choice.	
playGame - Player Make an	Create a test environment and simulate	Controller correctly processes the player's	
Attempt Choice	player choice.	"make an attempt" choice.	
		Returns a string containing the joined names	
joinNames - Valid List of Objects	A list of objects with names.	separated by commas.	
<u>,                                      </u>			
joinNames - Empty List of Objects	An empty list of objects.	Returns an empty string.	
, , , , , , , , , , , , , , , , , , , ,	controller.appendCommand("Test	The message is appended to the Appendable	
appendCommand - Valid Input	Message");	without errors.	
	controller.appendCommand("Test		
	Message"); with Appendable that throws	IllegalStateException is thrown with a wrapped	
appendCommand - IOException	an IOException.	IOException.	
CreatePlayer			
Test Case	Input	Expected Outcome	
	CreatePlayer createPlayer = new		
	CreatePlayer(new	No Exception is thrown, and the object is	
Constructor - Valid Input	RandomGeneratorImpl());	created.	
Constructor - Name and Space	CreatePlayer createPlayer = new	No Exception is thrown, and the object is	
Index	CreatePlayer("Player1", 1, 5);	created with the specified parameters.	
	Create a KillDoctorLucky model, execute	Auto-Player is created with random values for	
execute - Auto-Player Creation	with name = null.	name, currentSpaceIndex, and maxItems.	
- 12222 1220 1 Wy or or or or or or	Create a KillDoctorLucky model, execute	Player is created with the specified name,	
		, s	
execute - Player Creation	with valid name and parameters.	currentSpaceIndex, and maxItems.	

Test Case	Input	Expected Outcome
	MaxTurn maxTurnCommand = new	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.
	Parse	
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	cicated.
Constructor - File Not Found	Parse("nonexistent.txt");	FileNotFoundException is thrown.
Constructor - The Not Found	Create a KillDoctorLucky model and	The Noti outlide Acception 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.	
execute - Null Model	Create a KillDoctorLucky model and	IllegalArgumentException is thrown.
	•	
average Naukawasad Fila	execute the command with a specification	Illa and Charles Even which in the way we
execute - Malformed File	file that has incorrect formatting.	IllegalStateException is thrown.  Model populated with mansion details, target
Valid Charification File	Valid file with correct formatting	
Valid Specification File	Valid file with correct formatting	character, spaces, and items.
Invalid Specification File -		IllegalStateException is thrown due to missing mansion information.
Missing Mansion Information	File missing mansion information	
Invalid Specification File -	File with income at forms atting	IllegalStateException is thrown due to incorrect
Incorrect Format	File with incorrect formatting	format.
Null Madal	Execute command with a null model	IllegalArgumentException is thrown since the model cannot be null.
Null Model	Execute command with a null model	
NA aviana uma liba ma a Cantina a Nainaina a	File recipies the continue for thomas	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
	MovePlayer	T
Test Case	Input	Expected Outcome
		Create a MovePlayer instance with
		playerName set to "Player1" and targetSpace
Constructor Test 1	MovePlayer("Player1", 4)	set to 4.
		Create a MovePlayer instance with
		playerName set to "Player2" and targetSpace
Constructor Test 2	MovePlayer("Player2", 7)	set to 7.
	Create a KillDoctorLucky instance, create a	
	MovePlayer instance with playerName set	
	to "Player1" and targetSpace set to 4, then	The player with the name "Player1" is moved
Execute Test 1	execute the MovePlayer command.	to space 4 in the game.

	Create a KillDoctorLucky instance, create a	
	MovePlayer instance with playerName set	
	to "Player2" and targetSpace set to 7, then	The player with the pame "Dlayer?" is mayed
Evacuta Tast 2		The player with the name "Player2" is moved
Execute Test 2	execute the MovePlayer command.	to space 7 in the game.
Task Casa	PickUpItem	For sected Outserns
Test Case	Input	Expected Outcome
		Create a Right haltons instance with player Mares
Constructor Took 1	Dield belte ve ("Dlesse va" "Heeve 1")	Create a PickUpItem instance with playerName
Constructor Test 1	PickUpItem("Player1", "Item1")	set to "Player1" and itemName set to "Item1".
		Create a PickUpItem instance with playerName
Constructor Tost 2	Dick! Inltom("Playor?" "Itom?")	1
Constructor Test 2	PickUpItem("Player2", "Item2")	set to "Player2" and itemName set to "Item2".
	Create a KillDoctorLucky instance, create a	
	PickUpItem instance with playerName set	
		Dlever "Dlever" wisks on items "Ithoras III in the
5 . T . A	to "Player1" and itemName set to "Item1",	Player "Player1" picks up item "Item1" in the
Execute Test 1	then execute the PickUpItem command.	game.
	Create a KillDoctorLucky instance, create a	
	PickUpItem instance with playerName set	
	to "Player2" and itemName set to "Item2",	Player "Player2" picks up item "Item2" in the
Execute Test 2	then execute the PickUpItem command.	game.
	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.
	PetModel	
Test Case	Input	Expected Outcome
	PetModel petModel = new	No Exception is thrown, and the object is
Constructor - Valid Input	PetModel("Cat");	created with the specified name.
	Create a PetModel with a valid name value	Returns a string representation of the PetModel
toString - Valid name	and call toString.	object with the name.
	Ŭ	
	Create two PetModel objects with the	
hashCode - Same Name	same name and compare their hash codes.	Hash codes of both objects should be equal.
The street of th	Create two PetModel objects with the	nash codes of both objects should be equal.
	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.
	TECHNOLOUS THE HOLD	DUIEUS HAVE THE SATHE HATTE.

	Create two PetModel objects with different	
		The equals method should return false as the
equals - Not Equal Objects	method.	objects have different names.
equals - Not Equal Objects	Compare a PetModel object with an object	objects have different flames.
equals - Comparison with	of a different class using the equals	The equals method should return false as the
Different Class	method.	classes are different.
Different class	DoctorLucky Model	classes are uniferent.
Took Coop		Funcated Outcomes
Test Case	Input	Expected Outcome
	DoctorLuckyModel doctorLucky = new	No Exception is thrown, and the object is
Constructor - Valid Input	DoctorLuckyModel("Lucky", 100);	created with the specified name and health.
	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>	0-0-0-0-0	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.
tooting valid bata	can tooting.	ситенорисстисх.
	Create two PlayerModel objects with the	
hashCode - Same Name		Hash codes of both objects should be equal.
nashoue - Salle Naffle	same name and compare their hash codes.	masin codes of both objects should be equal.

	<u> </u>	
	Create two PlayerModel 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 PlayerModel 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 PlayerModel 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.
Directit class	Create a PlayerModel instance and pick up	classes are directific
	two Item instances using pickUpItem with	T
	different damages, then call	The name of the item with the maximum
getItemNameWithMaxDamage	getItemNameWithMaxDamage.	damage should be returned.
	Create a PlayerModel instance and call	
	removeItemByName with the name of an	
	item that does not exist in the player's	No items should be removed, and no exceptions
removeItemByName Test 1	items.	should be thrown.
	Create a PlayerModel instance, pick up two	
	Item instances using pickUpItem, and call	
	removeItemByName with the name of one	The specified item should be removed from the
removeItemByName Test 2	of the picked up items.	player's items.
-	Create a PlayerModel instance, pick up two	
	Item instances using pickUpItem, and call	
	removeItemByName with the name of	
	another item that does not exist in the	No items should be removed, and no exceptions
removeItemByName Test 3	player's items.	should be thrown.
,	ItemModel	
Test Case	Input	Expected Outcome
Test case	Imput	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.
equais - Not Equal Objects	the equals method.	objects have unferent harnes.

	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.
and Annais alafa Cat Manais	Create a KillDoctorLuckyModel object with	Data and a state of a second s
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.
get layers line deer layers line	Create a KillDoctorLuckyModel object and	the players.
getMaxTurn - Get Max Turn	call getMaxTurn.	Returns the maximum turn value that was set.
	Create a KillDoctorLuckyModel object with	
	predefined players and call	
getPlayerByTurn - Get Player By	getPlayerByTurn with various turn	Returns the player corresponding to the given
Turn	numbers.	turn number.
	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.
	Create a KillDoctorLuckyModel instance	A string regress retation of the recognism should
gotN/ansignInfo Tost	with a pre-defined mansion and call getMansionInfo.	A string representation of the mansion should be returned.
getMansionInfo Test	Create a KillDoctorLuckyModel instance	be returned.
	with a pre-defined doctorLucky and call	A string representation of the doctorLucky
getDoctorLuckyInfo Test	getDoctorLuckyInfo.	should be returned.
Set Doctor Edoky i i i o i e o i	Create a KillDoctorLuckyModel instance	should be retained.
	with pre-defined players and call	
	getPlayerByName with an existing player	The player with the specified name should be
getPlayerByName Test 1	name.	returned.
	Create a KillDoctorLuckyModel instance	
	with pre-defined players and call	
	getPlayerByName with a non-existing	null should be returned as there is no player
getPlayerByName Test 2	player name.	with the specified name.

	Create a KillDoctorLuckyModel instance	
	with pre-defined players and call	
getCurrentSpaceIndexByPlayerN	getCurrentSpaceIndexByPlayerName with	The current space index of the specified player
ame Test		should be returned.
alle lest	a player name.	should be returned.
	Create a KillDoctorLuckyModel instance	
	with a pre-defined mansion and call	A link of a simble color of a second control of
	getNeighborsBySpaceIndex with a space	A list of neighboring spaces for the specified
getNeighborsBySpaceIndex Test	index.	space index should be returned.
	Constant Will Destant walnut Adada Linetana	
	Create a KillDoctorLuckyModel instance	
	with a pre-defined mansion and call	A list of items in the specified space should be
getItemsBySpaceIndex Test	getItemsBySpaceIndex with a space index.	returned.
	Create a KillDoctorLuckyModel instance	
	with pre-defined players and call	The player type of the specified player should
gePlayerTypeByName Test	gePlayerTypeByName with a player name.	be returned.
	Create a KillDoctorLuckyModel instance	
	with a pre-defined doctorLucky and call	The name of the doctorLucky should be
getDoctorLuckyName Test	getDoctorLuckyName.	returned.
	Create a KillDoctorLuckyModel instance	
getDoctorLuckyCurrentSpaceInd	with a pre-defined doctorLucky and call	The current space index of the doctorLucky
ex Test	getDoctorLuckyCurrentSpaceIndex.	should be returned.
	Create a KillDoctorLuckyModel instance	
	with a pre-defined doctorLucky and call	The health of the doctorLucky should be
getDoctorLuckyHealth Test	getDoctorLuckyHealth.	returned.
	Create a KillDoctorLuckyModel instance	
	with a pre-defined mansion and call	The name of the space with the specified index
getSpaceNameByIndex Test	getSpaceNameByIndex with a space index.	should be returned.
	Create a KillDoctorLuckyModel instance	
	with a pre-defined mansion and call	A string representation of the space with the
getSpaceInfo Test	getSpaceInfo with a space index.	specified index should be returned.
	Create a KillDoctorLuckyModel instance	A new pet should be created with the specified
setPet Test	and call setPet with a pet name.	name.
	Create a KillDoctorLuckyModel instance	
	with a pre-defined pet and call	
getPetName Test	getPetName.	The name of the pet should be returned.
	Create a KillDoctorLuckyModel instance	
	with a pre-defined pet and call	The current space index of the pet should be
getPetCurrentSpaceIndex Test	getPetCurrentSpaceIndex.	returned.
	Create a KillDoctorLuckyModel instance	
	with a pre-defined mansion and call	The number of spaces in the mansion should be
getSpaceNumFromMansion Test	getSpaceNumFromMansion.	returned.
-	Create a KillDoctorLuckyModel instance	
	with a pre-defined pet and call movePet	
movePet Test	with an index.	The pet should be moved to the specified index.
	MansionModel	, and the same and the opening index
Test Case	Input	Expected Outcome
i est case	Imput	Елрескей Оиксопте

	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.
Trainet.	Create a MansionModel object and call	and secretaristani medilodi.
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.
Berehanes enterpases int	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.
	Create a MansionModel instance, add	
	spaces using addSpace, and call	The space at the specified index should be
getSpaceByIndex Test 1	getSpaceByIndex with a valid index.	returned.
800000000000000000000000000000000000000	Create a MansionModel instance, add	
	spaces using addSpace, and call	
	getSpaceByIndex with an index that is out	null should be returned as there is no space at
getSpaceByIndex Test 2	of bounds.	the specified index.
	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.
800.00.8.000	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		Returns a string representation of the space.
	space.toString()	
hashCode Test	space.hashCode()	Hash code based on name and points.
a sura la Ta at	space.equals(sameNameSamePointsSpac	house if the property of the state of the st
equals Test	e)	true if name and points match, false otherwise.

gotIndoy Tost	space gotIndov/)	Returns the index of the space.
getIndex Test	space.getIndex()	Returns the index of the space.