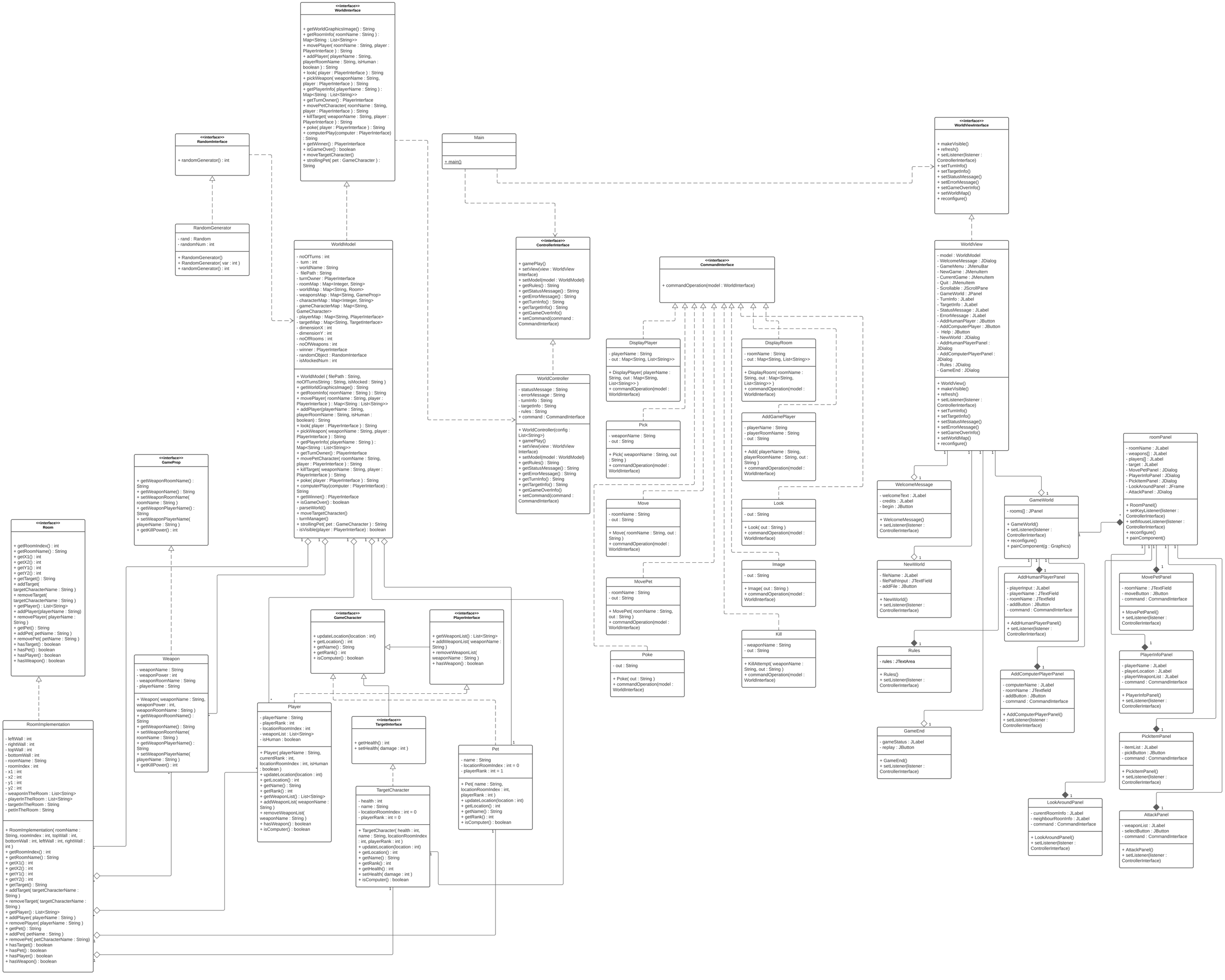
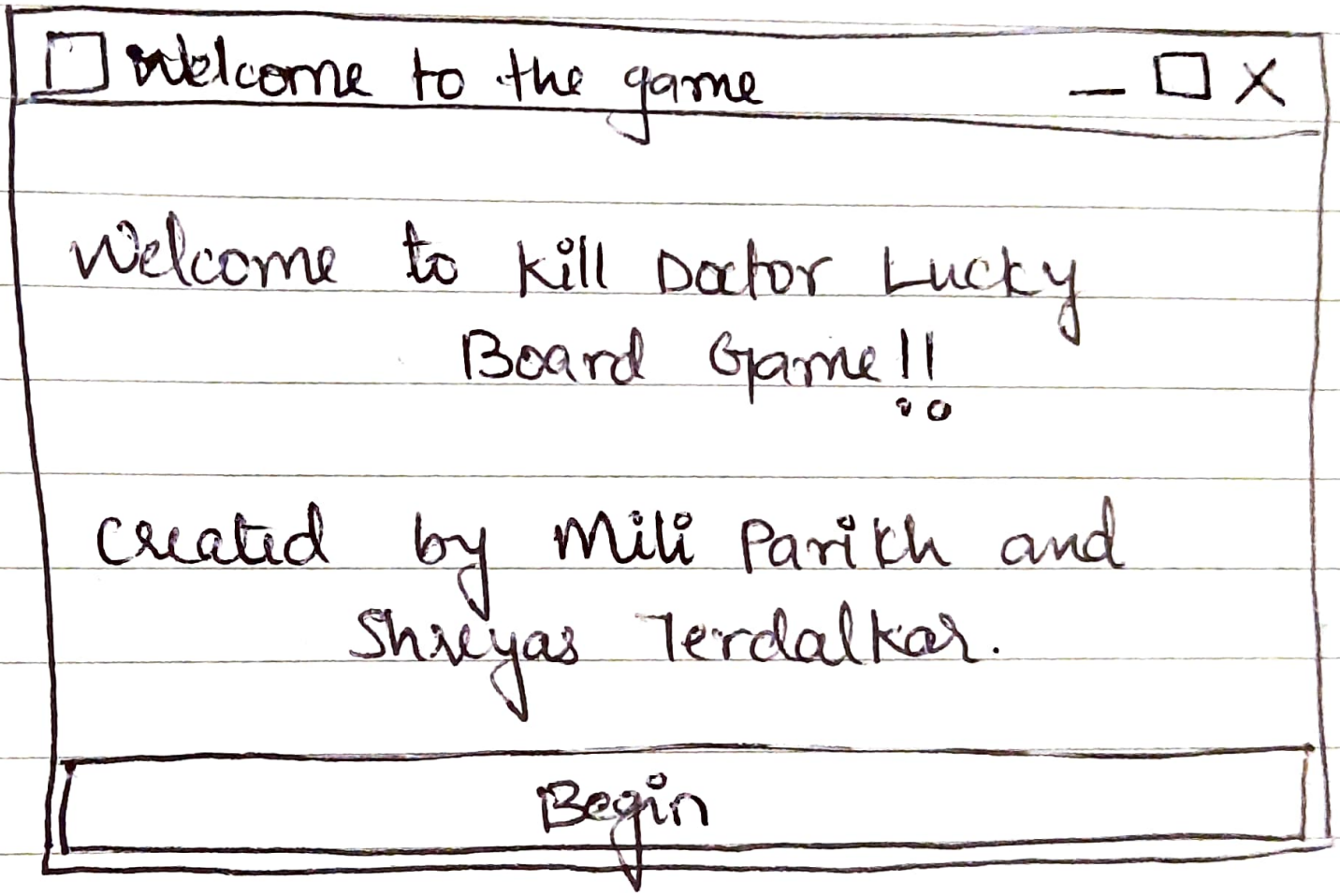
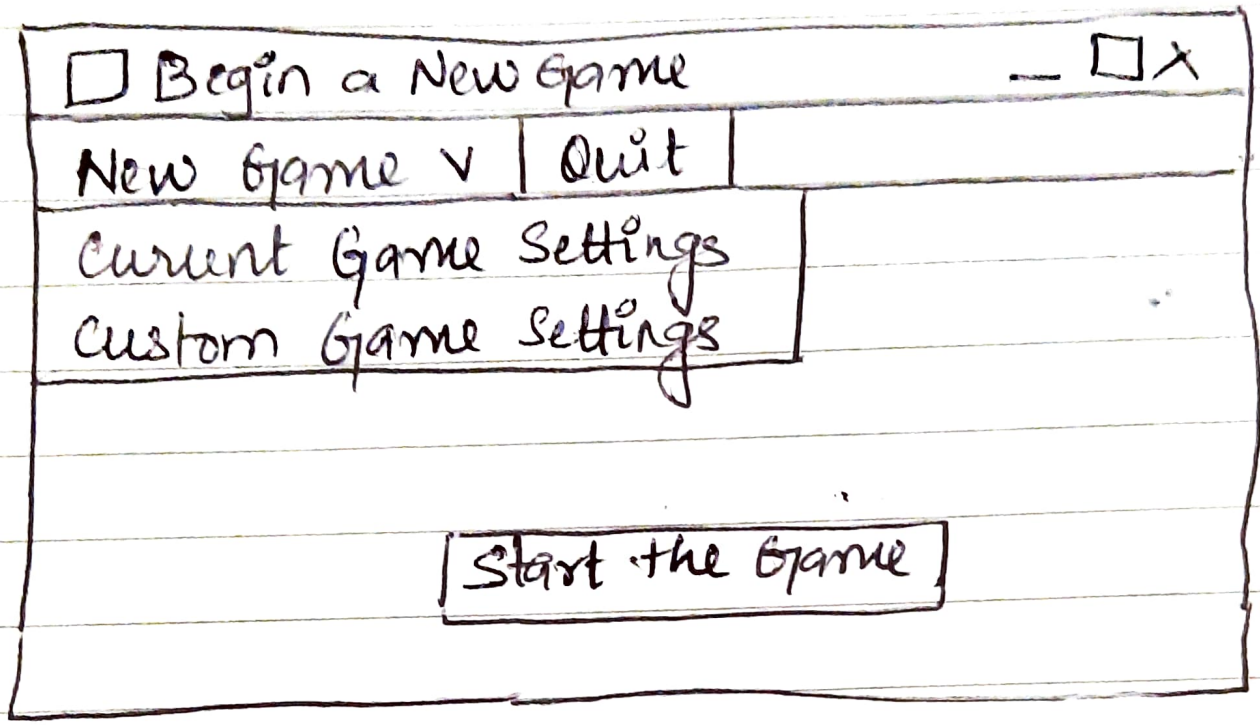

<<interface>>
RandomInterface
+ randomGenerator() : int
RandomGenerator
- rand : Random
- randomNum : int
+ RandomGenerator()
+ RandomGenerator(var : int)
+ randomGenerator() : int
<<interface>>
WorldInterface
+ getWorldGraphicsImage() : String
+ getRoomInfo(roomName : String) : Map<String : List<String>>
+ movePlayer(roomName : String, player : PlayerInterface) : String
+ addPlayer(playerName : String, playerRoomName : String, isHuman : boolean) : String
+ look(player : PlayerInterface) : String
+ pickWeapon(weaponName : String, player : PlayerInterface) : String
+ getPlayerInfo(playerName : String) : Map<String : List<String>>
+ getTurnOwner() : PlayerInterface
+ movePetCharacter(roomName : String, player : PlayerInterface) : String
+ killTarget(weaponName : String, player : PlayerInterface) : String
+ poke(player : PlayerInterface) : String
+ computerPlay(computer : PlayerInterface) : String
+ getWinner() : PlayerInterface
+ isGameOver() : boolean
+ moveTargetCharacter()
+ strollingPet(pet : GameCharacter) : String
Main
+ main()
<<interface>>
WorldViewInterface
+ makeVisible()
+ refresh()
+ setListener(listener : ControllerInterface)
+ setTurnInfo()
+ setTargetInfo()
+ setStatusMessage()
+ setErrorMessage()
+ setGameOverInfo()
+ setWorldMap()
+ reconfigure()
WorldModel
- noOfTurns : int
- turn : int
- worldName : String
- filePath : String
- turnOwner : PlayerInterface
- roomMap : Map<Integer, String>
- worldMap : Map<String, Room>
- weaponsMap : Map<String, GameProp>
- characterMap : Map<Integer, String>
- gameCharacterMap : Map<String, GameCharacter>
- playerMap : Map<String, PlayerInterface>
- targetMap : Map<String, TargetInterface>
- dimensionX : int
- dimensionY : int
- noOfRooms : int
- noOfWeapons : int
- winner : PlayerInterface
- randomObject : RandomInterface
- isMockedNum : int
+ WorldModel (filePath : String, noOfTurnsString : String, isMocked : String)
+ getWorldGraphicsImage() : String
+ getRoomInfo(roomName : String) : String
+ movePlayer(roomName : String, player : PlayerInterface) : Map<String : List<String>>
+ addPlayer(playerName : String, playerRoomName : String, isHuman : boolean) : String
+ look(player : PlayerInterface) : String
+ pickWeapon(weaponName : String, player : PlayerInterface) : String
+ getPlayerInfo(playerName : String) : Map<String : List<String>>
+ getTurnOwner() : PlayerInterface
+ movePetCharacter(roomName : String, player : PlayerInterface) : String
+ killTarget(weaponName : String, player : PlayerInterface) : String
+ poke(player : PlayerInterface) : String
+ computerPlay(computer : PlayerInterface) : String
+ getWinner() : PlayerInterface
+ isGameOver() : boolean
- parseWorld()
+ moveTargetCharacter()
- turnManager()
+ strollingPet(pet : GameCharacter) : String
- isVisible(player : PlayerInterface) : boolean
<<interface>>
ControllerInterface
+ gamePlay()
+ setView(view : WorldView Interface)
+ setModel(model : WorldModel)
+ getRules() : String
+ getStatusMessage() : String
+ getErrorMessage() : String
+ getTurnInfo() : String
+ getTargetInfo() : String
+ getGameOverInfo()
+ setCommand(command : CommandInterface)
WorldController
- statusMessage : String
- errorMessage : String
- turnInfo : String
- targetInfo : String
- rules : String
+ command : CommandInterface
+ WorldController(config : List<String>)
+ gamePlay()
+ setView(view : WorldView Interface)
+ setModel(model : WorldModel)
+ getRules() : String
+ getStatusMessage() : String
+ getErrorMessage() : String
+ getTurnInfo() : String
+ getTargetInfo() : String
+ getGameOverInfo()
+ setCommand(command : CommandInterface)
<<interface>>
CommandInterface
+ commandOperation(model : WorldInterface)
DisplayPlayer
- playerName : String
- out : Map<String, List<String>>
+ DisplayPlayer(playerName : String, out : Map<String, List<String>>)
+ commandOperation(model : WorldInterface)
DisplayRoom
- roomName : String
- out : Map<String, List<String>>
+ DisplayRoom(roomName : String, out : Map<String, List<String>>)
+ commandOperation(model : WorldInterface)
Pick
- weaponName : String
- out : String
+ Pick(weaponName : String, out : String)
+ commandOperation(model : WorldInterface)
AddGamePlayer
- playerName : String
- playerRoomName : String
- out : String
+ Add(playerName : String, playerRoomName : String, out : String)
+ commandOperation(model : WorldInterface)
Move
- roomName : String
- out : String
+ Move(roomName : String, out : String)
+ commandOperation(model : WorldInterface)
Look
- out : String
+ Look(out : String)
+ commandOperation(model : WorldInterface)
Image
- out : String
+ Image(out : String)
+ commandOperation(model : WorldInterface)
MovePet
- roomName : String
- out : String
+ MovePet(roomName : String, out : String)
+ commandOperation(model : WorldInterface)
Kill
- weaponName : String
- out : String
+ KillAttempt(weaponName : String, out : String)
+ commandOperation(model : WorldInterface)
Poke
- out : String
+ Poke(out : String)
+ commandOperation(model : WorldInterface)
WorldView
- model : WorldModel
- WelcomeMessage : JDialog
- GameMenu : JMenuBar
- NewGame : JMenuItem
- CurrentGame : JMenuItem
- Quit : JMenuItem
- Scrollable : JScrollPane
- GameWorld : JPanel
- TurnInfo : JLabel
- TargetInfo : JLabel
- StatusMessage : JLabel
- ErrorMessage : JLabel
- AddHumanPlayer : JButton
- AddComputerPlayer : JButton
- Help : JButton
- NewWorld : JDialog
- AddHumanPlayerPanel : JDialog
- AddComputerPlayerPanel : JDialog
- Rules : JDialog
- GameEnd : JDialog
+ WorldView()
+ makeVisible()
+ refresh()
+ setListener(listener : ControllerInterface)
+ setTurnInfo()
+ setTargetInfo()
+ setStatusMessage()
+ setErrorMessage()
+ setGameOverInfo()
+ setWorldMap()
+ reconfigure()
WelcomeMessage
- welcomeText : JLabel
- credits : JLabel
- begin : JButton
+ WelcomeMessage()
+ setListener(listener : ControllerInterface)
NewWorld
- fileName : JLabel
- filePathInput : JTextField
- addFile : JButton
+ NewWorld()
+ setListener(listener : ControllerInterface)
Rules
- rules : JTextArea
+ Rules()
+ setListener(listener : ControllerInterface)
GameEnd
- gameStatus : JLabel
- replay : JButton
+ GameEnd()
+ setListener(listener : ControllerInterface)
GameWorld
- rooms[] : JPanel
+ GameWorld()
+ setListener(listener : ControllerInterface)
+ reconfigure()
+ painComponent(g : Graphics)
AddHumanPlayerPanel
- playerInput : JLabel
- playerName : JTextField
- roomName : JTextfield
- addButton : JButton
- command : CommandInterface
+ AddHumanPlayerPanel()
+ setListener(listener : ControllerInterface)
AddComputerPlayerPanel
- computerName : JLabel
- roomName : JTextfield
- addButton : JButton
- command : CommandInterface
+ AddComputerPlayerPanel()
+ setListener(listener : ControllerInterface)
LookAroundPanel
- curentRoomInfo : JLabel
- neighbourRoomInfo : JLabel
- command : CommandInterface
+ LookAroundPanel()
+ setListener(listener : ControllerInterface)
roomPanel
- roomName : JLabel
- weapons[] : JLabel
- players[] : JLabel
- target : JLabel
- MovePetPanel : JDialog
- PlayerInfoPanel : JDialog
- PickItemPanel : JDialog
- LookAroundPanel : JFrame
- AttackPanel : JDialog
+ RoomPanel()
+ setKeyListener(listener : ControllerInterface)
+ setMouseListener(listener : ControllerInterface)
+ reconfigure()
+ painComponent()
MovePetPanel
- roomName : JTextField
- moveButton : JButton
- command : CommandInterface
+ MovePetPanel()
+ setListener(listener : ControllerInterface)
PlayerInfoPanel
- playerName : JLabel
- playerLocation : JLabel
- playerWeaponList : JLabel
- command : CommandInterface
+ PlayerInfoPanel()
+ setListener(listener : ControllerInterface)
PickItemPanel
- itemList : JLabel
- pickButton : JButton
- command : CommandInterface
+ PickItemPanel()
+ setListener(listener : ControllerInterface)
AttackPanel
- weaponList : JLabel
- selectButton : JButton
- command : CommandInterface
+ AttackPanel()
+ setListener(listener : ControllerInterface)
<<interface>>
GameProp
+ getWeaponRoomName() : String
+ getWeaponName() : String
+ setWeaponRoomName(roomName : String)
+ getWeaponPlayerName() : String
+ setWeaponPlayerName(playerName : String)
+ getKillPower() : int
<<interface>>
Room
+ getRoomIndex() : int
+ getRoomName() : String
+ getX1() : int
+ getX2() : int
+ getY1() : int
+ getY2() : int
+ getTarget() : String
+ addTarget(targetCharacterName : String)
+ removeTarget(targetCharacterName : String)
+ getPlayer() : List<String>
+ addPlayer(playerName : String)
+ removePlayer(playerName : String)
+ getPet() : String
+ addPet(petName : String)
+ removePet(petName : String)
+ hasTarget() : boolean
+ hasPet() : boolean
+ hasPlayer() : boolean
+ hasWeapon() : boolean
<<interface>>
GameCharacter
+ updateLocation(location : int)
+ getLocation() : int
+ getName() : String
+ getRank() : int
+ isComputer() : boolean
<<interface>>
PlayerInterface
+ getWeaponList() : List<String>
+ addWeaponList(weaponName : String)
+ removeWeaponList(weaponName : String)
+ hasWeapon() : boolean
Weapon
- weaponName : String
- weaponPower : int
- weaponRoomName : String
- playerName : String
+ Weapon(weaponName : String, weaponPower : int, weaponRoomName : String)
+ getWeaponRoomName() : String
+ getWeaponName() : String
+ setWeaponRoomName(roomName : String)
+ getWeaponPlayerName() : String
+ setWeaponPlayerName(playerName : String)
+ getKillPower() : int
Player
- playerName : String
- playerRank : int
- locationRoomIndex : int
- weaponList : List<String>
- isHuman : boolean
+ Player(playerName : String, currentRank : int, locationRoomIndex : int, isHuman : boolean)
+ updateLocation(location : int)
+ getLocation() : int
+ getName() : String
+ getRank() : int
+ getWeaponList() : List<String>
+ addWeaponList(weaponName : String)
+ removeWeaponList(weaponName : String)
+ hasWeapon() : boolean
+ isComputer() : boolean
<<interface>>
TargetInterface
+ getHealth() : int
+ setHealth(damage : int)
TargetCharacter
- health : int
- name : String
- locationRoomIndex : int = 0
- playerRank : int = 0
+ TargetCharacter(health : int, name : String, locationRoomIndex : int, playerRank : int)
+ updateLocation(location : int)
+ getLocation() : int
+ getName() : String
+ getRank() : int
+ getHealth() : int
+ setHealth(damage : int)
+ isComputer() : boolean
Pet
- name : String
- locationRoomIndex : int = 0
- playerRank : int = 1
+ Pet(name : String, locationRoomIndex : int, playerRank : int)
+ updateLocation(location : int)
+ getLocation() : int
+ getName() : String
+ getRank() : int
+ isComputer() : boolean
RoomImplementation
- leftWall : int
- rightWall : int
- topWall : int
- bottomWall : int
- roomName : String
- roomIndex : int
- x1 : int
- x2 : int
- y1 : int
- y2 : int
- weaponInTheRoom : List<String>
- playerInTheRoom : List<String>
- targetInTheRoom : String
- petInTheRoom : String
+ RoomImplementation(roomName : String, roomIndex : int, topWall : int, bottomWall : int, leftWall : int, rightWall : int)
+ getRoomIndex() : int
+ getRoomName() : String
+ getX1() : int
+ getX2() : int
+ getY1() : int
+ getY2() : int
+ getTarget() : String
+ addTarget(targetCharacterName : String)
+ removeTarget(targetCharacterName : String)
+ getPlayer() : List<String>
+ addPlayer(playerName : String)
+ removePlayer(playerName : String)
+ getPet() : String
+ addPet(petName : String)
+ removePet(petCharacterName : String)
+ hasTarget() : boolean
+ hasPet() : boolean
+ hasPlayer() : boolean
+ hasWeapon() : boolean

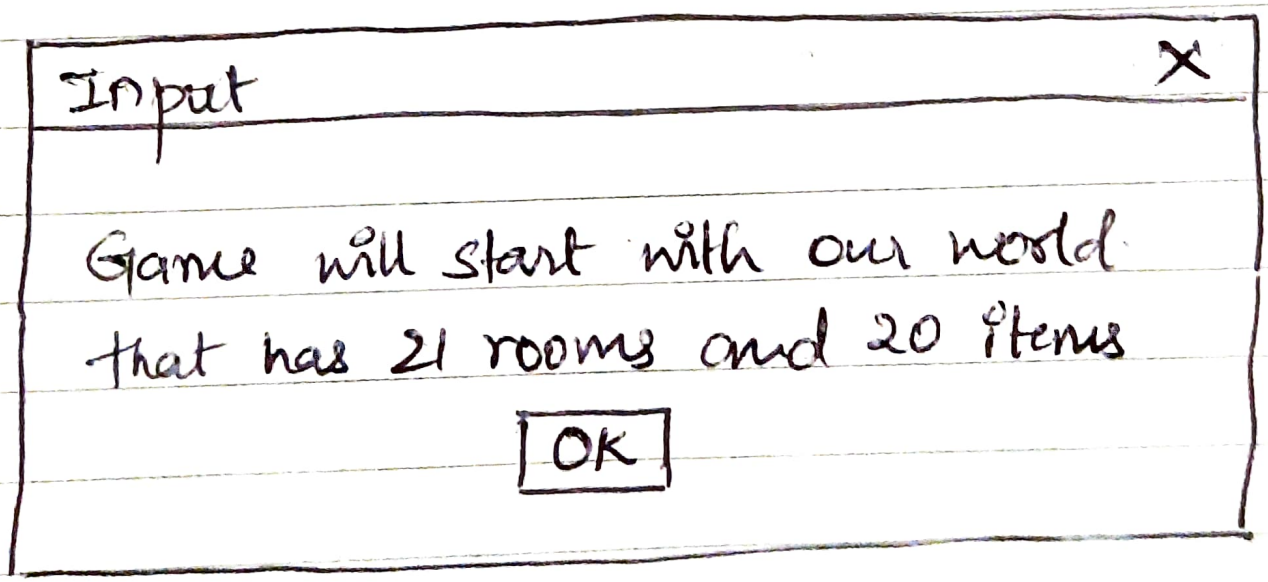
WELCOME TO THE GAME PAGE



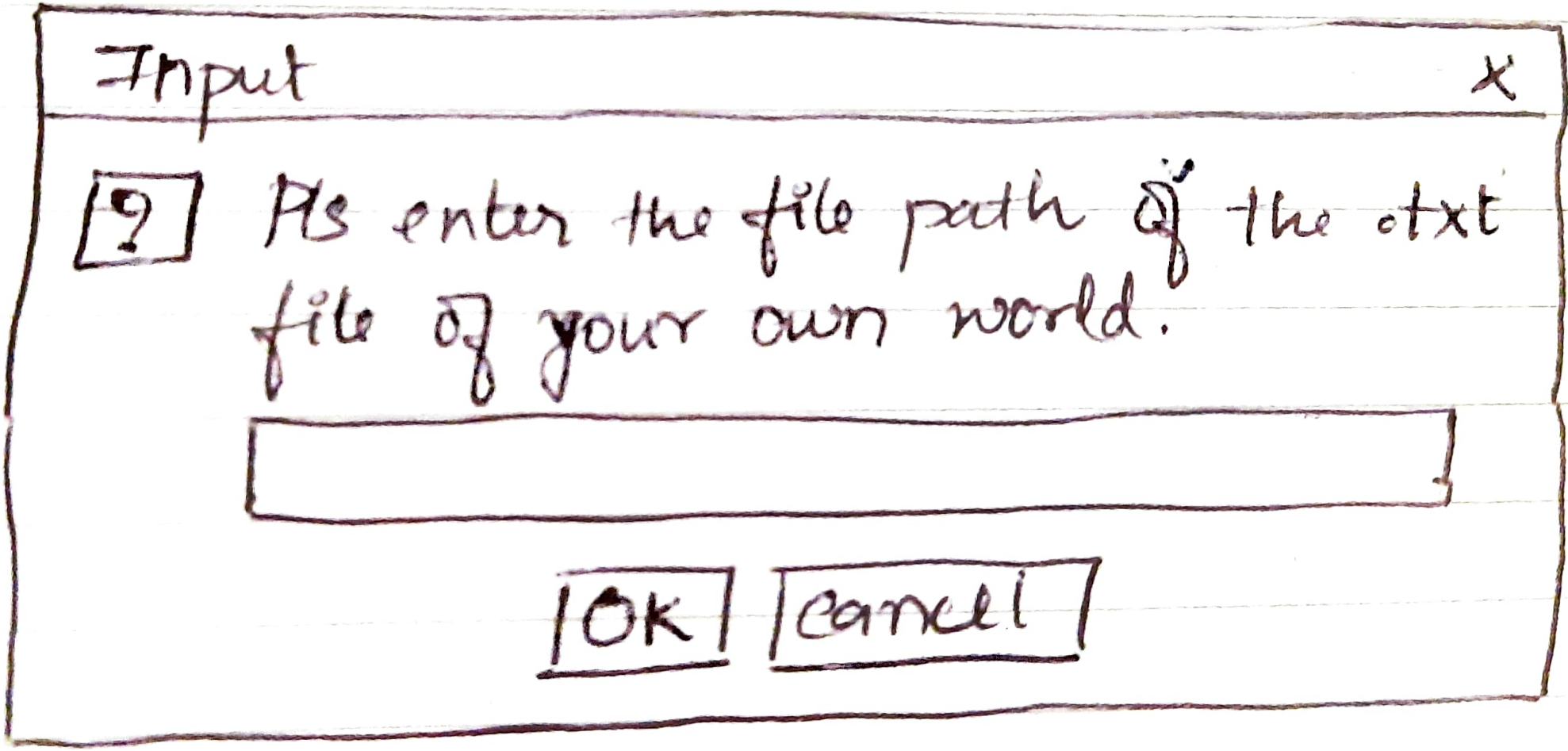
START A NEW GAME PAGE



Option → Current Game settings (JDialog)



Option → Custom Game Settings (JDialog)



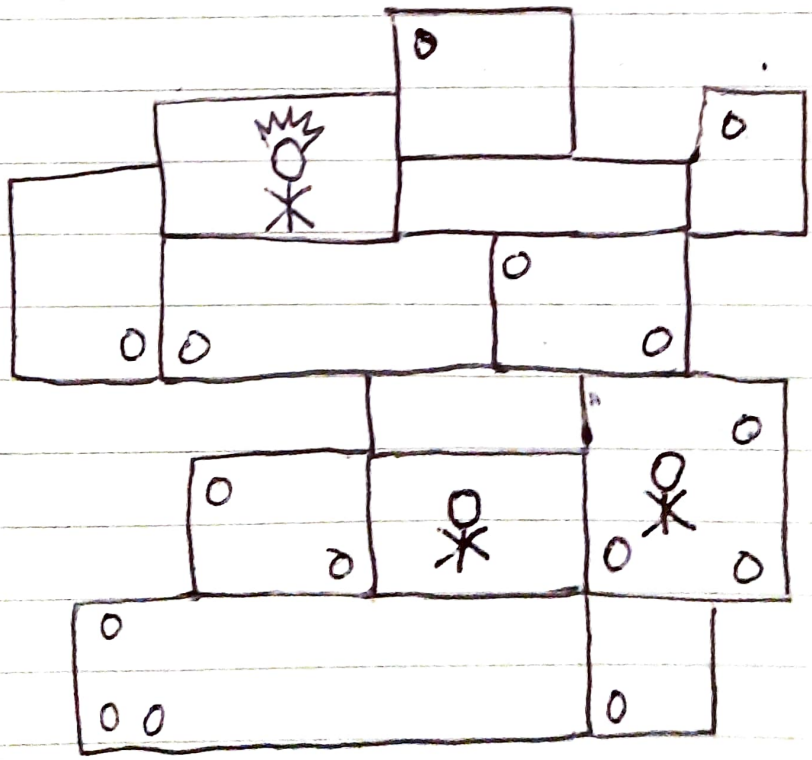
GAME PAGE

Doctor Lucky's Mansion _ □ X

Add Player | Help | Quit

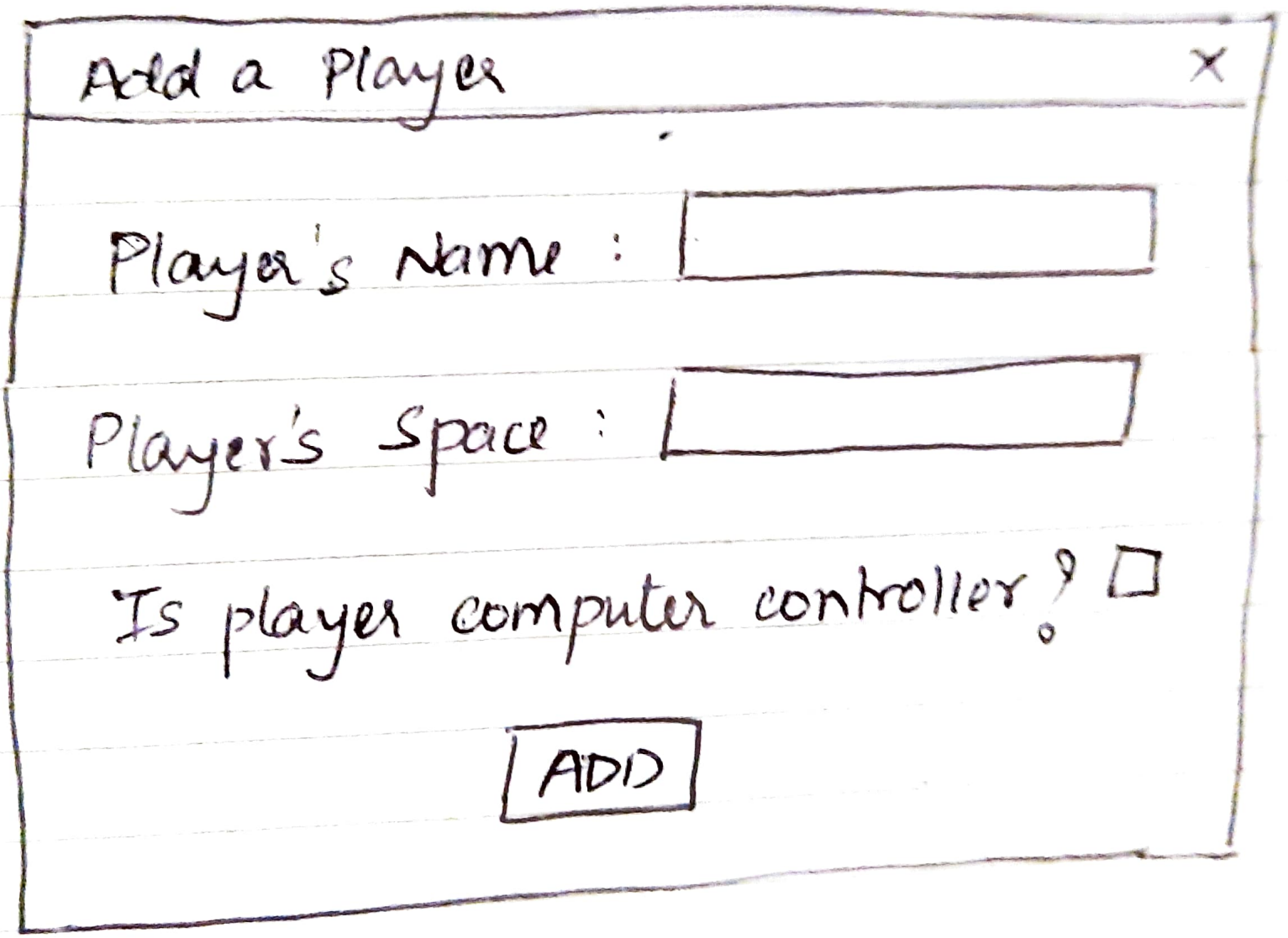
Add Human Player
Add computer Player

Turn of Player : Player's Name
Player is present in X Space.
Turn Numer : 1/100
Target is present in X Space.
Target Health : 9/10

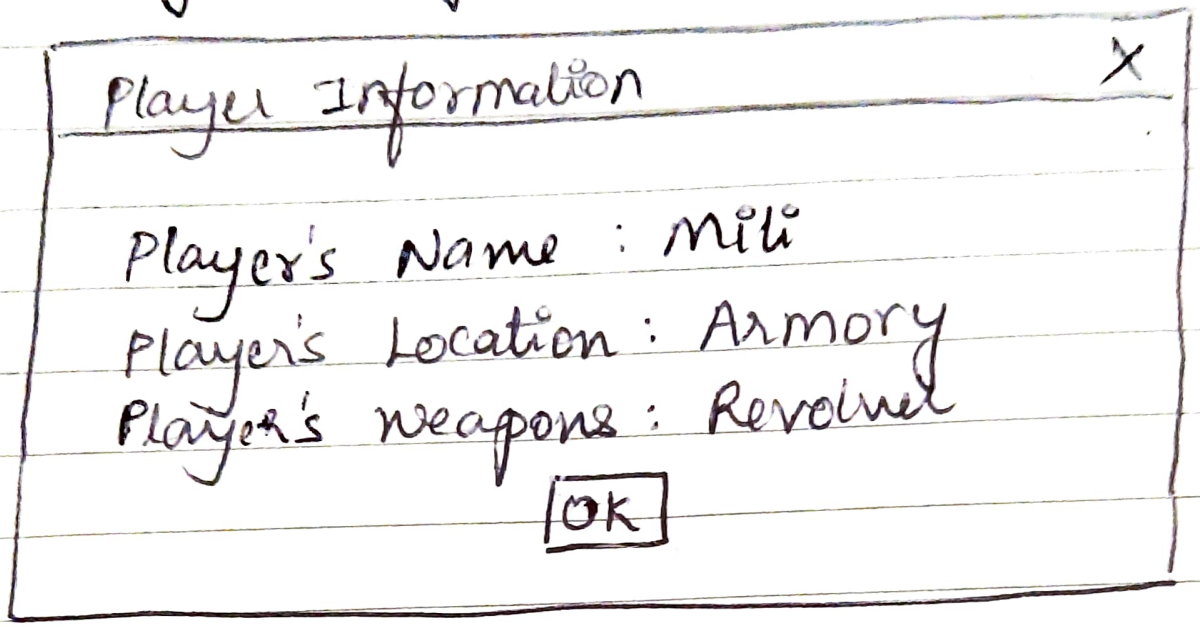


O → items → players → target

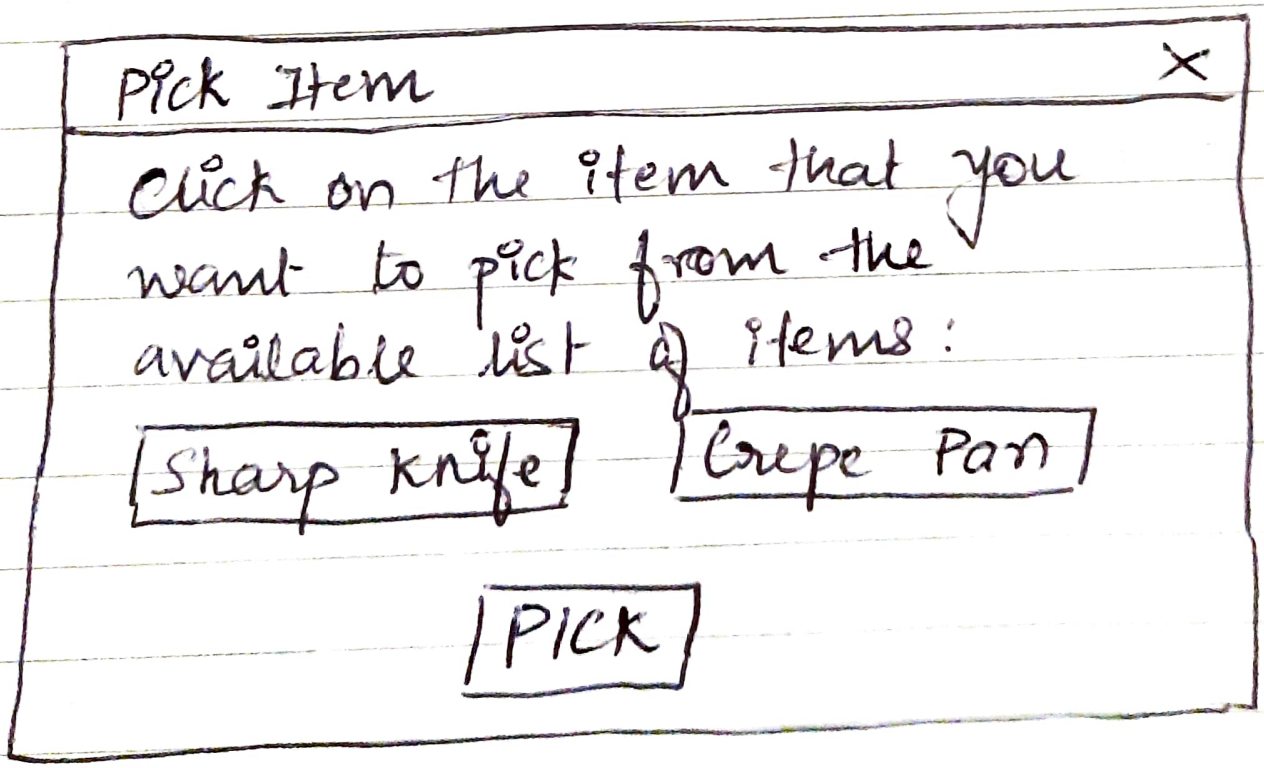
ADD A NEW PLAYER (JDialog)



PLAYER INFORMATION
(By clicking on the player icon)



PICK ITEMS



LOOK AROUND

Looking Around ×

Looking Around (Space Name):
Space's Name: Kitchen
Player's in space: Mili
Item's in space: Crepe Pan
Neighbours of space:
1: Dining Hall
Player's in dining hall:
Item's in dining hall:
Neighbours of dining hall:

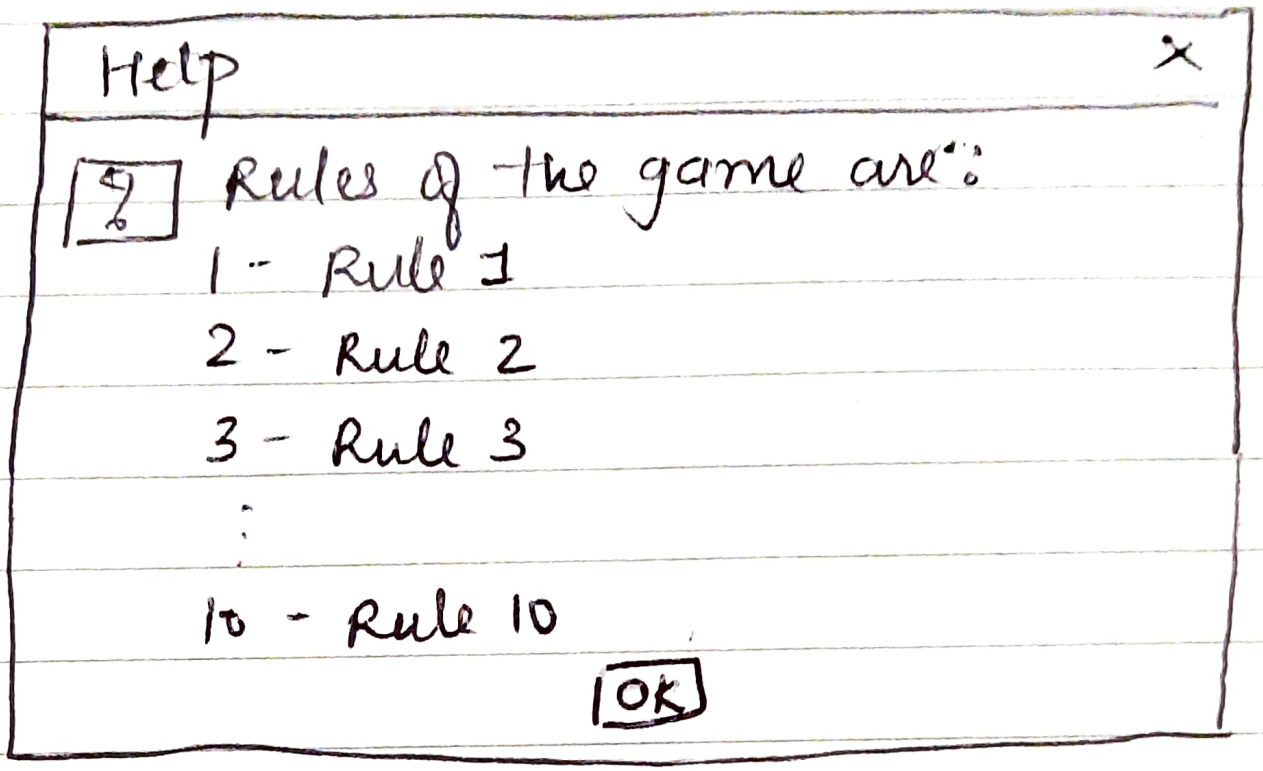
2: Parlor
Player's in Parlor:
Item's in Parlor:
Neighbours of Parlor:

3: Wine Cellar
Player's in Wine Cellar:
Item's in Wine Cellar:
Neighbours of Wine Cellar:

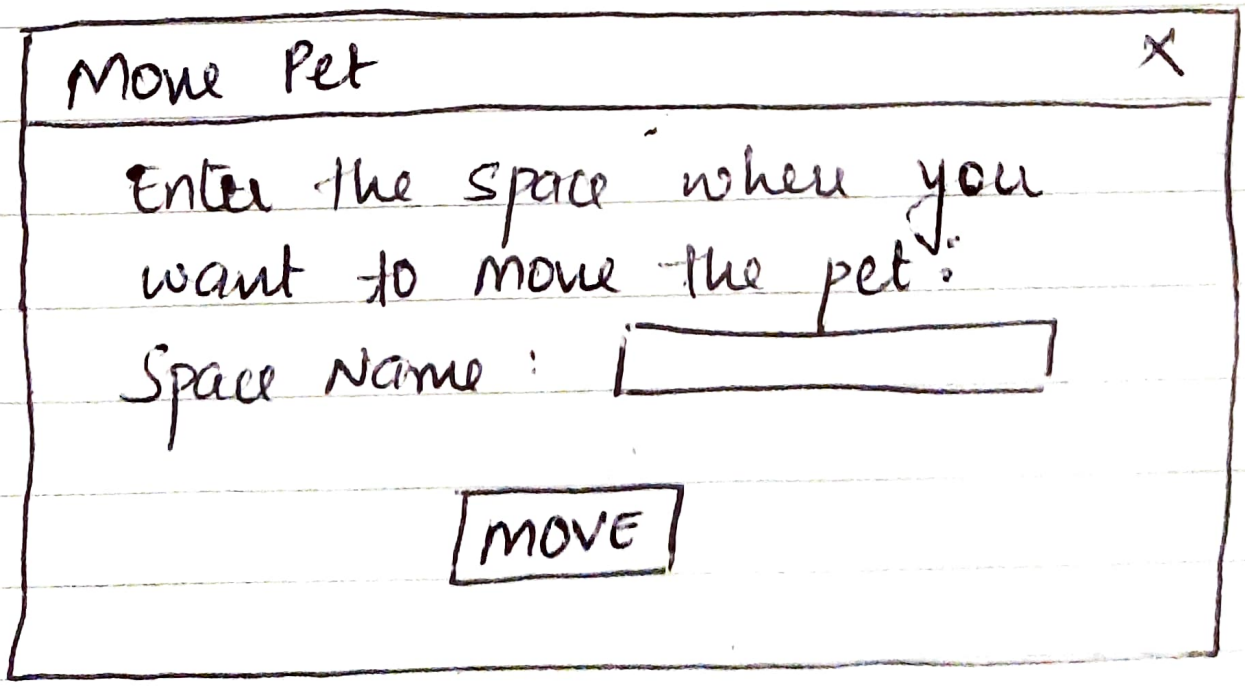
(Target / Pet info if present)

OK

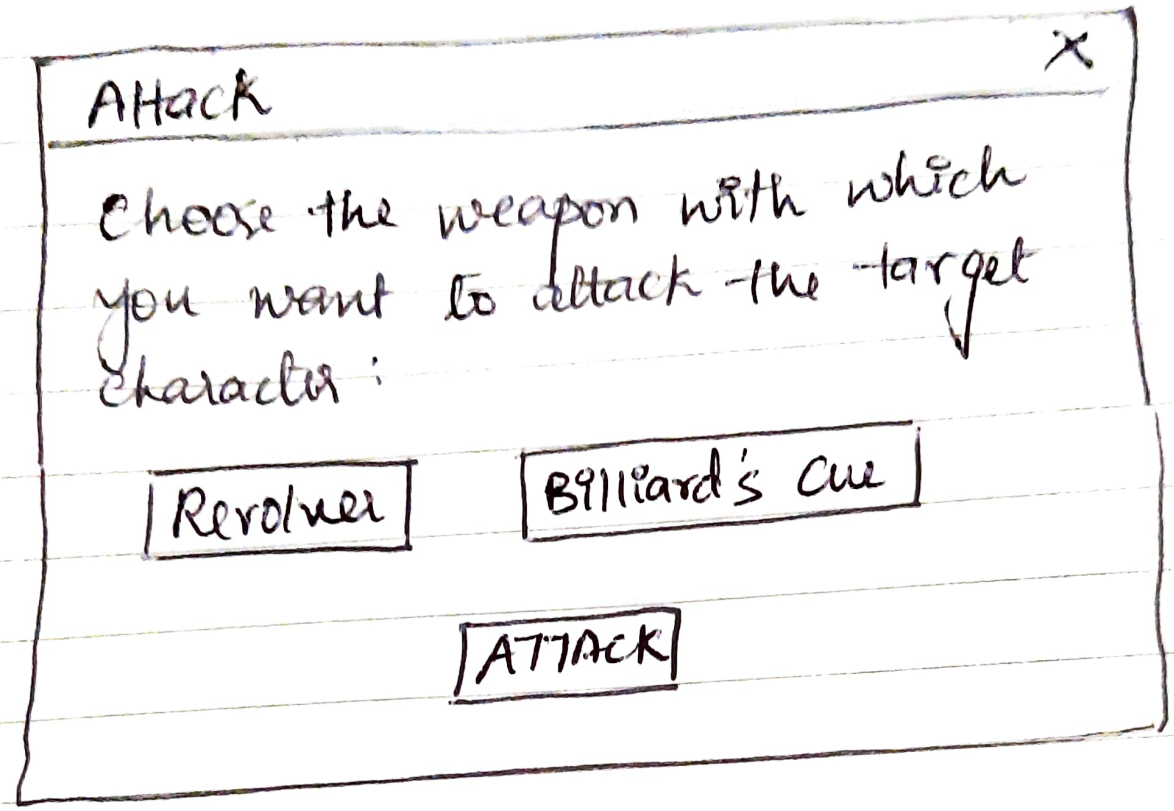
HELP (JDialog)



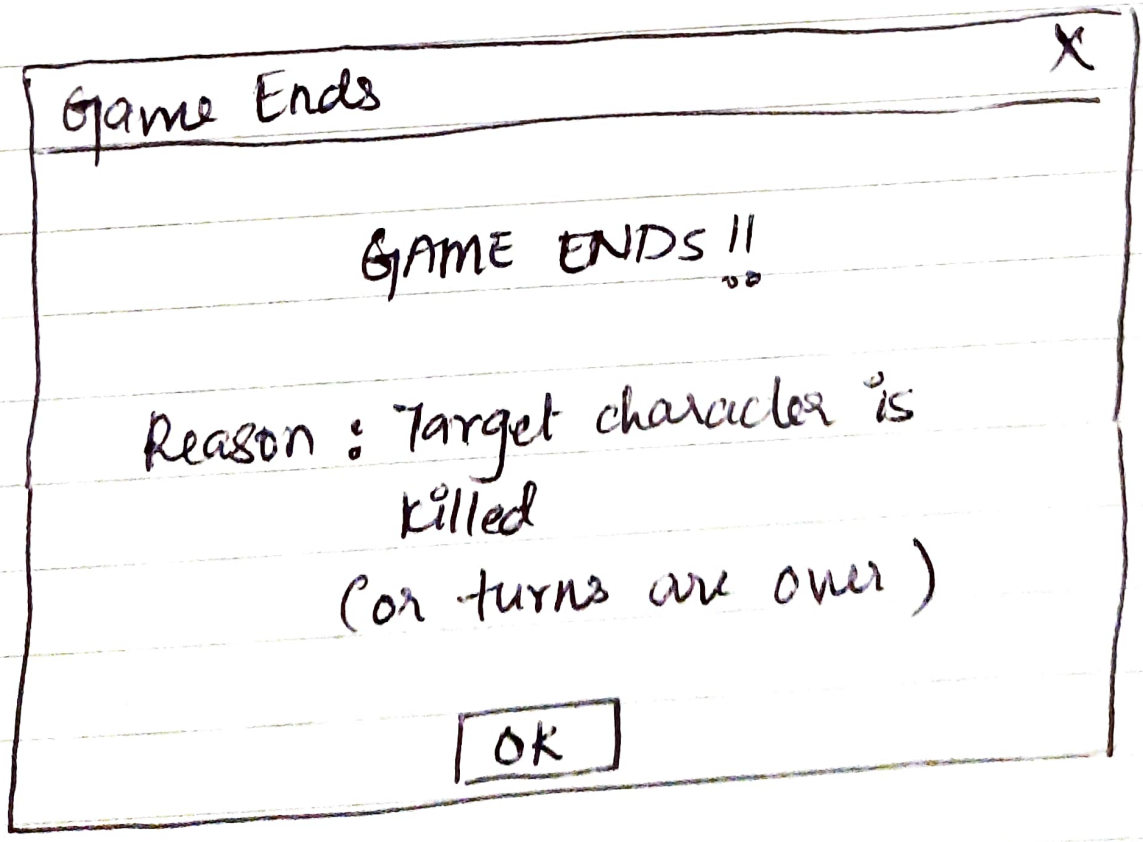
MOVEPET



ATTACK TARGET



GAME ENDS



MILESTONE 4 MODEL CHANGES:

1. The return type of getRoomInfo and getPlayerInfo were initially String because the output of the console based game was in the form of String. However, Now the getRoomInfo and getPlayerInfo methods of the model return a copy of the information in the form of HashMap<String, List<String>> The key strings represent parameters eg. Room Name, Room Index, Room Weapons, Room Characters, etc. and Player Name, Player Location, Player Weapons, etc. and the value contains corresponding information in the for of List of Strings.
2. The moveTargetCharacter and strollingPet methods were private as they were internally called after each player's actions were executed. However in the GUI based game, these methods will have to be called from the controller and to be displayed in the form of Target Information on the GUI by the view. Therefore, these methods were made Public. All the rest of the model remains unchanged as the requirements of the GUI based game are still entirely fulfilled by our well designed model.

(Model testing does not change due to the new changes in the model because in milestone 3 model testing was done by calling the .toString() method and hence the strings were compared. So even after the above changes, the same strings can be compared and hence no changes are needed in the test cases)

MILESTONE 4 TESTING PLAN:

We will first create a Mock Model that implements the WorldInterface. All the methods in this mock model overrides the methods of the interface and returns statements on the console using out.append() where out is an object of appendable. By doing so, we can conclude which methods are being called when. Further, a unicode is also returned by the methods for identifying if the correct method is called or not.

We will also create a Mock View that implements the WorldViewInterface. All the methods in this mock view overrides the methods of the interface and returns statements on the console using out.append() where out is an object of appendable. By doing so, we can conclude which methods are being called when. Further, a unicode is also returned by the methods for identifying if the correct method is called or not.

Further, we will pass instances of the mock model and the mock view to the controller to test the controller in isolation. Test cases to test the controller methods in isolation are listed below.

TEST	INPUT	EXPECTED VALUE
Start the game with new world	controller.gamePlay() controller.setModel("path")	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called.

		OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated.
Start the game with new world, incorrect path	controller.gamePlay() controller.setModel(“path”)	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. Incorrect input. Pls try again.
Start the game with new world, file does not exist	controller.gamePlay() controller.setModel(“path”)	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. File does not exist. Pls try again.
Start the game and quit	controller.gamePlay() controller.setModel(“path”)	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld

		called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated. IsGameOver method in mock model called. Game is over.
Start the game with current world	controller.gamePlay() controller.setModel("path") controller.getGameOverInfo())	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated.
Adding human player	controller.gamePlay() controller.setModel("path") CommandInterface addGamePlayerObj = new AddGamePlayer("Mili", "Armory") controller.setCommand(model) addGamePlayerObj.execute() controller.reconfigure()	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times.

		setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view called.
Adding computer player	controller.gamePlay() controller.setModel("path") CommandInterface addGamePlayerObj = new AddGamePlayer("Mili", "Armory") controller.setCommand(model) addGamePlayerObj.execute() controller.reconfigure()	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times. setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view

		called.
Asking for help	controller.gamePlay() controller.setModel("path") CommandInterface addGamePlayerObj = new AddGamePlayer("Mili", "Armory") controller.setCommand(model) addGamePlayerObj.execute() controller.reconfigure() controller.getRules()	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times. setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view called. Rules called.
Moving to another room	controller.gamePlay() controller.setModel("path") CommandInterface addGamePlayerObj = new AddGamePlayer("Mili", "Armory") controller.setCommand(model) addGamePlayerObj.execute()	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated.

	controller.reconfigure() CommandInterface MoveObj = new Move(“Billiard Room”) controller.setCommand(model)	setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times. setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view called. Move constructor initiated. setListener method in Move called. OK button was clicked. Move command constructor initiated. MovePlayer method in the mock model called. Refresh method in the mock view called.
Looking into another room	controller.gamePlay() controller.setModel(“path”) CommandInterface addGamePlayerObj = new AddGamePlayer(“Mili”, “Armory”) controller.setCommand(model) addGamePlayerObj.execute() controller.reconfigure() CommandInterface LookObj = new Look() controller.setCommand(model)	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view

		called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times. setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view called. Look constructor initiated. setListener method in Look called. OK button was clicked. Look command constructor initiated. Look method in the mock model called. Refresh method in the mock view called.
Picking a weapon	controller.gamePlay() controller.setModel(“path”) CommandInterface addGamePlayerObj = new AddGamePlayer(“Mili”, “Armory”) controller.setCommand(model) addGamePlayerObj.execute() controller.reconfigure() CommandInterface PickObj = new Pick(“Revolver”) controller.setCommand(model)	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times. setListener method of the GameWorld

		called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view called. Pick constructor initiated. setListener method in Pick called. OK button was clicked. Pick command constructor initiated. pickWeapon method in the mock model called. Refresh method in the mock view called.
Moving the pet	controller.gamePlay() controller.setModel(“path”) CommandInterface addGamePlayerObj = new AddGamePlayer(“Mili”, “Armory”) controller.setCommand(model) addGamePlayerObj.execute() controller.reconfigure() CommandInterface movePetObj = new MovePet(“Nursery”) controller.setCommand(model)	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times. setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in

		AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view called. MovePet constructor initiated. setListener method in MovePet called. OK button was clicked. MovePet command constructor initiated. movePetCharacter method in the mock model called. Refresh method in the mock view called.
Attacking target with weapon	controller.gamePlay() controller.setModel(“path”) CommandInterface addGamePlayerObj = new AddGamePlayer(“Mili”, “Armory”) controller.setCommand(model) addGamePlayerObj.execute() controller.reconfigure() CommandInterface PickObj = new Pick(“Revolver”) controller.setCommand(model) controller.reconfigure() CommandInterface MoveObj = new Move(“Billiard Room”) controller.setCommand(model) controller.reconfigure() CommandInterface MoveObj = new Move(“Dining Hall”) controller.setCommand(model) controller.reconfigure() CommandInterface killObj =	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times. setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated.

	new Kill(“Revolver”) controller.setCommand(model)	AddPlayer method in the mock model called. Refresh method in the mock view called. Pick constructor initiated. setListener method in Pick called. OK button was clicked. Pick command constructor initiated. pickWeapon method in the mock model called. Refresh method in the mock view called. Move constructor initiated. setListener method in Move called. OK button was clicked. Move command constructor initiated. MovePlayer method in the mock model called. Refresh method in the mock view called. Move constructor initiated. setListener method in Move called. OK button was clicked. Move command constructor initiated. MovePlayer method in the mock model called. Refresh method in the mock view called. Kill constructor initiated. setListener method in Kill called. OK button was clicked. Kill command constructor initiated. killTarget method in the mock model called. Refresh method in the mock view called.
Attacking target by poking	controller.gamePlay() controller.setModel(“path”) CommandInterface addGamePlayerObj = new AddGamePlayer(“Mili”, “Armory”) controller.setCommand(model) addGamePlayerObj.execute()	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated.

	controller.reconfigure() CommandInterface pokeObj = new Poke(“Revolver”) controller.setCommand(model)	setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times. setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view called. Poke constructor initiated. setListener method in Poke called. OK button was clicked. Poke command constructor initiated. poke method in the mock model called. Refresh method in the mock view called.
No of turns getting over	(Assuming turns = 1 target health = 5) controller.gamePlay() controller.setModel(“path”) CommandInterface addGamePlayerObj = new AddGamePlayer(“Mili”, “Armory”) controller.setCommand(model) addGamePlayerObj.execute() controller.reconfigure() CommandInterface pokeObj = new Poke(“Revolver”)	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view

	controller.setCommand(model)	called. GameWorld constructor initiated. RoomPanel constructor initiated x21 times. setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view called. Poke constructor initiated. setListener method in Poke called. OK button was clicked. Poke command constructor initiated. poke method in the mock model called. Refresh method in the mock view called. isGameOver method in the mock model called. Game is over as turns are over.
Player winning the game	(Assuming turns = 5 target health = 1) controller.gamePlay() controller.setModel(“path”) CommandInterface addGamePlayerObj = new AddGamePlayer(“Mili”, “Armory”) controller.setCommand(model) addGamePlayerObj.execute() controller.reconfigure() CommandInterface pokeObj = new Poke(“Revolver”) controller.setCommand(model)	MockView constructor initiated. WelcomeMessage constructor initiated. setListener method in WelcomeMessage called. OK button was clicked. makeVisible method in the mock view is called. setListener method was called. NewWorld constructor initiated. setListener method in NewWorld called. OK button selected. MockModel constructor initiated. Refresh method in the mock view called. GameWorld constructor initiated.

		RoomPanel constructor initiated x21 times. setListener method of the GameWorld called. setListener method of the RoomPanel called. AddHumanPlayer constructor initiated. setListener method in AddHumanPlayer called. OK button was clicked. AddGamePlayer command constructor initiated. AddPlayer method in the mock model called. Refresh method in the mock view called. Poke constructor initiated. setListener method in Poke called. OK button was clicked. Poke command constructor initiated. poke method in the mock model called. Refresh method in the mock view called. isGameOver method in the mock model called. Game is over as the target character is killed.

MODEL TESTING

Test	Input	Expected Value
Looking around		
Test for no weapons in the current room	model.Look()	Shreyas's Turn : Player Shreyas is in Dining Hall Neighbour rooms : [Kitchen, Armory, Tennessee Room, Drawing Room]

Test for one weapon in the current room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Neighbour rooms : [Dining Hall, Tennessee Room, ServantS' Quarters]
Test for multiple weapons in the current room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife, Crepe Pan Neighbour rooms : [Dining Hall, Tennessee Room, ServantS' Quarters]
Test for another player in the current room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Other character/s in the room : Professor Neighbour rooms : [Dining Hall, Tennessee Room, ServantS' Quarters]
Test for multiple players in the current room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Other character/s in the room : Professor, Alex Neighbour rooms : [Dining Hall, Tennessee Room, ServantS' Quarters]

Test for computer player in the current room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Other character/s in the room : Professor, Computer Neighbor rooms : [Dining Hall, Tennessee Room, ServantS' Quarters]

Test for another player in the neighbor room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Other character/s in the room : Professor, Computer Neighbour rooms : [Dining Hall, Tennessee Room, Servants' Quarters] Alex is in Dining Hall
Test for multiple players in the neighbour room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Other character/s in the room : Computer Neighbour rooms : [Dining Hall, Tennessee Room, Servants' Quarters] Alex is in Dining Hall Professor is in Servants' Quarters
Test for computer player in the neighbour room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Neighbour rooms : [Dining Hall, Tennessee Room, Servants' Quarters] Alex is in Dining Hall Computer is in Tennessee Room Professor is in Servants' Quarters
Test for no neighbour rooms	model.Look()	Shreyas's Turn : Player Shreyas is in Plaza Neighbour rooms : None
Test for one neighbour room	model.Look()	Shreyas's Turn : Player Shreyas is in Piazza Neighbour rooms : [Carriage House] Alex is in Dining Hall Computer is in Tennessee Room Professor is in Servants' Quarters

Test for neighbour room with multiple weapons	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Neighbour rooms : [Dining Hall, Tennessee Room, Servants' Quarters] Alex is in Dining Hall Computer is in Tennessee Room Professor is in Servants' Quarters

Test for Target in the current room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Character/s in the room : Target Neighbour rooms : [Dining Hall, Tennessee Room, Servants' Quarters] Alex is in Dining Hall Computer is in Tennessee Room Professor is in Servants' Quarters
Test for Target in the neighbour room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Neighbour rooms : [Dining Hall, Tennessee Room, Servants' Quarters] Alex is in Dining Hall Target is in Tennessee Room Professor is in Servants' Quarters
Test for pet in the current room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Character/s in the room : Fortune Neighbour rooms : [Dining Hall, Tennessee Room, Servants' Quarters] Alex is in Dining Hall Computer is in Tennessee Room

		Professor is in Servants' Quarters
Test for pet in the neighbour room	model.Look()	Shreyas's Turn : Player Shreyas is in Kitchen Weapon/s in the room : Sharp Knife Neighbour rooms : [Dining Hall, Tennessee Room] Alex is in Dining Hall Target is in Tennessee Room
Moving pet		
Test for moving pet to the neighbour room	model.MovePet("Dining Hall") model.DisplayRoom("Dining Hall")	Pet moved to Dining Hall. Room Name : Dining Hall Neighbors : Trophy Room, Kitchen, Armory Character/s in the room : Professor, Fortune
Test for moving pet to non neighbour room	model.MovePet("Piazza")	Invalid input, please re-enter
Killing Target		

Test for attempting successful damage to the target using a weapon that the player had by incrementing turn to the next player and the target's health being reduced and the weapon getting removed from the game	model.Kill("Billiard Cue") model.DisplayRoom("Billiard Cue") model.DisplayPlayer("Shreyas")	Shreyas's turn: KILL Billiard Cue Damage successful! Doctor Lucky's health: 47 Professor's Turn: DISPLAYROOM Billiard Room Name of the room : Billiard Room Neighbours of the room : [Dining Hall, Drawing Room, Armory] Weapon/s in the room : None Player/s in the room : [Shreyas] Professor's Turn: DISPLAYPLAYER Shreyas Name of the player : Shreyas Location of the player : Billiard Room Player has weapon/s : None

Unsuccessful damage because of another player seeing from the neighbour room, turn being incremented, target's health not getting reduced, but weapon being removed from the world	model.Kill("Billiard Cue") model.DisplayRoom("Billiard Cue") model.DisplayPlayer("Shreyas")	Shreyas's turn: KILL Billiard Cue Damage unsuccessful! Doctor Lucky's health: 50 Professor's Turn: DISPLAYROOM Billiard Room Name of the room : Billiard Room Neighbours of the room : [Dining Hall, Drawing Room, Armory] Professor is in Armory Weapon/s in the room : None Player/s in the room : [Shreyas] Professor's Turn: DISPLAYPLAYER Shreyas Name of the player : Shreyas Location of the player : Billiard Room Player has weapon/s : Billiard Cue

Unsuccessful damage because the player doesnt have the specified weapon, turn not being incremented, target's health not getting reduced, weapon not being removed from the world, user asked for new input	model.Kill("Sharp Knife") model.DisplayRoom("Billiard Cue") model.DisplayPlayer("Shreyas")	Shreyas's turn: KILL Billiard Cue Incorrect input, please re-enter Shreyas's Turn: DISPLAYROOM Billiard Room Name of the room : Billiard Room Neighbours of the room : [Dining Hall, Drawing Room, Armory] Weapon/s in the room : None Player/s in the room : [Shreyas] Shreyas's Turn: DISPLAYPLAYER Shreyas Name of the player : Shreyas Location of the player : Billiard Room Player has weapon/s : Billiard Cue

Unsuccessful damage because of the another player seeing from the same room, turn being incremented, target's health not getting reduced, but weapon being removed from the world	model.Kill("Billiard Cue") model.DisplayRoom("Billiard Cue") model.DisplayPlayer("Shreyas")	Shreyas's turn: KILL Billiard Cue Damage unsuccessful! Doctor Lucky's health: 50 Professor's Turn: DISPLAYROOM Billiard Room Name of the room : Billiard Room Neighbours of the room : [Dining Hall, Drawing Room, Armory] Professor is in Billiard Room Weapon/s in the room : None Player/s in the room : [Shreyas, Professor] Professor's Turn: DISPLAYPLAYER Shreyas Name of the player : Shreyas Location of the player : Billiard Room Player has weapon/s : Billiard Cue

Unsuccessful damage because the player is not in the same space as the target, turn not being incremented, target's health not getting reduced, weapon not being removed from the world, user asked for new input	model.Kill("Billiard Cue") model.DisplayRoom("Billiard Cue") model.DisplayPlayer("Shreyas")	Shreyas's turn: KILL Billiard Cue Target is not in Billiard Room Incorrect input, please re-enter Shreyas's Turn: DISPLAYROOM Billiard Room Name of the room : Billiard Room Neighbours of the room : [Dining Hall, Drawing Room, Armory] Weapon/s in the room : None Player/s in the room : [Shreyas] Shreyas's Turn: DISPLAYPLAYER Shreyas Name of the player : Shreyas Location of the player : Billiard Room Player has weapon/s : Billiard Cue

Successful damage being done by the computer player by incrementing turn to the next player and the target's health being reduced and the weapon getting removed from the game	NA	Computer's turn: Damage successful! Doctor Lucky's health: 47 Professor's Turn: DISPLAYROOM Billiard Room Name of the room : Billiard Room Neighbours of the room : [Dining Hall, Drawing Room, Armory] Weapon/s in the room : None Player/s in the room : [Computer] Professor's Turn: DISPLAYPLAYER Computer Name of the player : Computer Location of the player : Billiard Room Player has weapon/s : None

Successful damage by poking in the eye by incrementing turn to the next player and the target's health being reduced	model.Poke()	Shreyas's turn: POKE Damage successful! Doctor Lucky's health: 49 Professor's Turn: DISPLAYROOM Billiard Room Name of the room : Billiard Room Neighbours of the room : [Dining Hall, Drawing Room, Armory] Weapon/s in the room : None Player/s in the room : [Shreyas] Professor's Turn: DISPLAYPLAYER Shreyas Name of the player : Shreyas Location of the player : Billiard Room Player has weapon/s : None
Game over		

After enough damages, target's health becomes zero, game ends	model.Poke()	Target is dead. Game Over!
Maximum no of turns limit reached	model.Look()	Maximum no of turns reached. Game Over!
Adding Target's pet during parsing of the world specification and creation of the world at room 0 along with the target	model.DisplayRoom(“Armory)	Name of the room : Armory Neighbours of the room : [Dining Hall, Drawing Room, Billiard Room] Weapon/s in the room : [Revolver] Character/s in the room : Target, Fortune
Adding player displays information of their current room details and target information	model.AddPlayer(“Shreyas”, “Armory”)	Name of the room : Armory Neighbours of the room : [Dining Hall, Drawing Room, Billiard Room] Weapon/s in the room : [Revolver] Target is in Armory Character/s in the room : Shreyas , Doctor lucky, Fortune
Adding computer player displays information of their current room details and target information	model.AddPlayer(“Shreyas” , “Armory”)	Name of the room : Armory Neighbours of the room : [Dining Hall, Drawing Room, Billiard Room] Weapon/s in the room : [Revolver] Target is in Armory Character/s in the room : Computer , Doctor lucky, Fortune
Information about the Target moving in the world is being displayed at the end of every turn to the user	model.Pick(“Revolver”)	PICK Revolver Shreyas has picked Revolver Target moved to Billiard Room Computer's turn :

Information of pet moving in the world using DFS algorithm after every turn	model.Pick("Revolver")	PICK Revolver Shreyas has picked Revolver Target moved to Billiard Room Fortune moved to Carriage House Computer's turn :
DISPLAY ROOM COMMAND: **Display information about a specified space in the world.**		
Valid Test with all fields available	model.DisplayRoom("Billiard Cue")	ROOM NAME : BILLIARD ROOM WEAPON IN THE ROOM: BILLIARD CUE NEIGHBOURS: TROPHY ROOM, DINING HALL, ARMORY PEOPLE IN THE ROOM: DOCTOR LUCKY, PLAYERONE
Invalid Test for room not in the World	model.DisplayRoom("Plazza")	IllegalArgumentException
Test for room having no weapon	model.DisplayRoom("Servant's Qarter")	ROOM NAME : SERVANT'S QUARTERS WEAPON IN THE ROOM: NONE NEIGHBOURS: TROPHY ROOM, DINING HALL, ARMORY PEOPLE IN THE ROOM: DOCTOR LUCKY, PLAYERONE
Test for room having no neighbours	model.DisplayRoom("Plazza")	IllegalArgumentException
Test for room having no characters	model.DisplayRoom("Foyer")	ROOM NAME : FOYER WEAPON IN THE ROOM: REVOLVER NEIGHBOURS: TENNESSEE ROOM, DINING HALL, ARMORY PEOPLE IN THE ROOM: NONE

Test for room having only target character	model.DisplayRoom("Lilac Room")	ROOM NAME : LILAC ROOM WEAPON IN THE ROOM: SHARP KNIFE NEIGHBOURS: TROPHY ROOM, DINING HALL, FOYER PEOPLE IN THE ROOM: DOCTOR LUCKY
Test for room having only player	model.DisplayRoom("Library")	ROOM NAME : LIBRARY WEAPON IN THE ROOM: CHAIN SAW NEIGHBOURS: TROPHY ROOM, DINING HALL PEOPLE IN THE ROOM: PLAYERONE
CREATE IMAGE COMMAND: **Create a graphical representation of the world map and provide the ability to save the graphical representation to a file as a PNG file.**		
Valid Test for graphics image	mansion.txt	image.png
Invalid Test for image file not found	mansion.txt	FileNotFoundException

ADD PLAYER COMMAND: **Add a human-controlled player to the game.**		
Valid Test for creating player	model.AddPlayer("One")	PLAYER ONE ADDED
ADD COMPUTERPLAYER COMMAND: **Add a computer-controlled player to the game.**		
Valid Test for creating computer player	model.AddComputerPlayer()	COMPUTER PLAYER ADDED
MOVE COMMAND: **Move a player.**		
Valid Test for moving player	model.Move("Lilac Room")	LOCATION : LILAC ROOM

Valid Test for computer player moving automatically on its turn	NA	LOCATION : LIBRARY
Invalid Test for moving to non neighbour room	model.Move(“Foyer”)	IllegalArgumentException
PICK COMMAND: **Allow a player to pick up an item.**		
Valid Test for picking item from current room	model.Pick(“Chain Saw”)	WEAPON TAKEN : CHAINSAW
Invalid Test for room with no item	model.Pick(“Billiard Cue”)	IllegalArgumentException
Invalid Test for item already picked	model.Pick(“Sharp Knife”)	IllegalArgumentException
DISPLAY PLAYER COMMAND: **Display a description of a specific player.**		
Valid Test for all fields present	model.DisplayPlayer(“One”)	PLAYER : ONE LOCATION : BILLIARD ROOM WEAPON TAKEN : BILLIARD CUE
Valid Test for player with no weapon	model.DisplayPlayer(“One”)	PLAYER : ONE LOCATION : BILLIARD ROOM WEAPON TAKEN : NONE
Invalid Test for player not in the World	model.DisplayPlayer(“Hero”)	IllegalArgumentException
Limit the maximum number of turns allowed		

Valid Test for no of turns exceeding limit	model.Look()	GAME IS OVER! TURNS LIMIT REACHED
Others		
Valid Test for input being readable and output being appendable	mansion.txt	WELCOME TO DOCTOR LUCKY'S WORLD

Invalid Test for input being readable and output being appendable	mansion.txt	IllegalStateException
Tests for Model		
General		
Model object is null	model == null	IllegalArgumentException
Model object is readable	mansion.t xt	WELCOME TO DOCTOR LUCKY'S WORLD
Parsing World specification correctly	mansion.t xt	WELCOME TO DOCTOR LUCKY'S WORLD
Overlapping spaces	mansion.t xt	IllegalArgumentException
Rooms with same names	mansion.t xt	IllegalArgumentException
Weapons with same name	mansion.t xt	IllegalArgumentException
Test getWorldGraphicsImage()		
Valid Test for graphics image	mansion.t xt	image.png
Invalid Test for image file not found	mansion.t xt	FileNotFoundException
Test getNeighbour()		
Top Left Room	LANCASTER ROOM	["LILAC ROOM","SERVANTS' QUARTERS"]
Top Right Room	NURSERY	["MASTER SUITE","LIBRARY"]
Bottom Left Room	CARRIAGE HOUSE	["WINTER GARDEN"]
Botton Right Room	GREEN HOUSE	["HEDGE MAZE"]
Middle Room	DINING HALL	["TENNESSEE ROOM","PARLOR","KITCHEN","WINE

		CELLAR","DRA WING ROOM","ARM ORY","BILLIAR D ROOM","TROP HY ROOM"]
Middle Room	PARLOR	["SERVANTS' QUARTERS","DINING ROOM","KITCHEN","TENNES SEE ROOM"]
Incorrect Input Room	ATTIC	IllegalArgumentException
Test getRoomInfo()		

Room 1	BILLIARD ROOM	"ROOM NAME : BILLIARD ROOM, WEAPON IN THE ROOM : BILLIARD CUE, NEIGHBOURS : TROPHY ROOM, DINING HALL, ARMORY PEOPLE IN THE ROOM : DOCTOR LUCKY"
Room 2	FOYER	"ROOM NAME : FOYER, WEAPON IN THE ROOM : NO WEAPON FOUND, NEIGHBOURS : DRAWING ROOM, PIAZZA PEOPLE IN THE ROOM : PLAYERONE"
Incorrect Input Room	CLINIC ROOM	IllegalArgumentException
Valid Test with all fields available	model.DisplayRoom(“Billiard Room”)	ROOM NAME : BILLIARD ROOM WEAPON IN THE ROOM: BILLIARD CUE NEIGHBOURS: TROPHY ROOM, DINING HALL, ARMORY PEOPLE IN THE ROOM:

		DOCTOR LUCKY, PLAYERONE
Invalid Test for room not in the World	model.DisplayRoom(“My room”)	IllegalArgumentException
Test for room having no weapon	model.DisplayRoom(“Servant’s Quarter”)	ROOM NAME : SERVANT'S QUARTERS WEAPON IN THE ROOM: NONE NEIGHBOURS: TROPHY ROOM, DINING HALL, ARMORY PEOPLE IN THE ROOM: DOCTOR LUCKY, PLAYERONE
Test for room having no neighbours	model.DisplayRoom(“Plaza”)	IllegalArgumentException

Test moveTargetCharacter()

Increment once	move()	BILLIARD ROOM
Increment six times	move() six times	FOYER
Test movePlayer()		
Valid Test for moving player	model.Move(“One”, “Lilac Room”)	LOCATION : LILAC ROOM
Valid Test for computer player moving	NA	LOCATION : LIBRARY

automatically on its turn		
Invalid Test for moving to non neighbour room	model.Move(“One”, “Library”)	IllegalArgumentException
Test getTurn()		
Valid Test for no of turns exceeding limit	NA	GAME IS OVER! TURNS LIMIT REACHED

Playerone's turn	NA	PLAYERONE'S TURN

Individual class testing

Test	Input	Expected Value
Testing room implementation		
Valid test for creation of room using valid arguments	"0, Armory, 22, 19, 23, 26"	"Room Index : 0, Room Name : Armory, TopWall : 22, LeftWall : 19, BottomWall : 23, RightWall : 26"
Invalid test while creating a room for invalid/null arguments	" , Armory, 22, 19, 23, 26" "0, , 22, 19, 23, 26" "0, Armory, , 19, 23, 26" "0, Armory, 22, , 23, 26" "0, Armory, 22, 19, , 26" "0, Armory, 22, 19, 23, " " 57, Armory, 22, 19, 23, 26" "0, @#$%, 22, 19, 23, 26" "0, Armory, 66, 19, 23, 26" "0, Armory, 22, 98, 23, 26" "0, Armory, 22, 19, -9, 26" "0, Armory, 22, 19, 23, 0.8”	IllegalArgumentException

Valid test for testing of all getter methods	"0, Armory, 22, 19, 23, 26"	"Room Index : 0, Room Name : Armory, TopWall : 22, LeftWall : 19, BottomWall : 23, RightWall : 26 Target : Doctor Lucky Player : Computer Pet : Fortune Weapon : Revolver”

Valid test for testing of all setter methods	1. Creation “0, Armory, 22, 19, 23, 26" 2. Add Target “Doctor Lucky” 3. Remove Target “Doctor Lucky” 4. Add Player “Shreyas” 5. Remove Player “Shreyas” 6. Add Pet “Fortune” 7. Remove Pet “Fortune”	1. "Room Index : 0, Room Name : Armory, TopWall : 22, LeftWall : 19, BottomWall : 23, RightWall : 26" 2. “Doctor Lucky” 3. Null 4. “Shreyas” 5. Null 6. “Fortune” 7. Null
Invalid test for testing invalid/null arguments to setter methods	1. Creation “0, Armory, , 19, 23, 26" 2. Add Target “” 3. Remove Target “Doctor Loki” 4. Add Player 123 5. Remove Player Null 6. Add Pet Null 7. Remove Pet “#$%”	IllegalArgumentException
Testing player implementation		
Valid test for creation of player using valid arguments	"Shreyas, 2, 0"	“Player Name : Shreyas Player Rank : 2 Player room index : 0”
Invalid test while creating a player for invalid/null arguments	"$%^, 2, "	IllegalArgumentException
Valid test for testing of all getter methods	"Shreyas, 2, 0"	1. “Shreyas, 2, 0” 2. “Shreyas” 3. 2 4. 0

		5. [Revolver]
Valid test for testing of all setter methods	1. Creation - “Shreyas, 2, 0” 2. Update Location - 4 3. Add weapon - [Revolver] 4. Remove weapon - [Revolver]	1. “Player Name : Shreyas Player Rank : 2 Player room index : 0” 2. 4 3. [Revolver] 4. Null
Invalid test for testing invalid/null arguments to setter methods	1. Creation - “ , 2, 0” 2. Update Location - 46 3. Add weapon - [^&*] 4. Remove weapon - [Revolver]	IllegalArgumentException
Testing weapon implementation		
Valid test for creation of weapon using valid arguments	"Sharp Knife, 3, 8"	"Weapon name - Sharp Knife Kill Power - 3 Location - Kitchen"
Invalid test while creating a weapon for invalid/null arguments	" , 3, 8"	IllegalArgumentException
Valid test for testing of all getter methods	"Sharp Knife, 3, 8"	"Weapon name - Sharp Knife Kill Power - 3 Location - Kitchen"
Valid test for testing of all setter methods	1. creation - "Sharp Knife, 3, 8" 2. set weapon location - Null 3. set player name - Shreyas	1. "Weapon name - Sharp Knife Kill Power - 3" 2. "Location - Null" 3. "Player - Shreyas"
Invalid test for testing invalid/null arguments to setter methods	1. creation - "Sharp Knife, 3, 8" 2. set weapon location - Null 3. set player name - Null	IllegalArgumentException
Testing target implementation		
Valid test for creation of target using valid arguments	"Doctor Lucky, 50, 0, 0"	"Name - Doctor Lucky Health - 50 Location - Armory Player rank - 0"

Invalid test while creating a target for invalid/null arguments	" , 50, 0, 0"	IllegalArgumentException
Valid test for testing of all getter methods	"Doctor Lucky, 50, 0, 0"	1. "Name - Doctor Lucky" 2. "Health - 50" 3. "Location - Armory" 4. "Player rank - 0"
Valid test for testing of all setter methods	1. update location - 2 2. set health - 3	1. "Location - Dining Hall" 2. "Health - 47"
Invalid test for testing invalid/null arguments to setter methods	1. update location - 90 2. set health - 51	IllegalArgumentException
Testing pet implementation		
Valid test for creation of pet using valid arguments	"Fortune, 1, 0"	“Name : Fortune Location : Armory Player rank : 1”
Invalid test while creating a pet for invalid/null arguments	" , 1, 0"	IllegalArgumentException
Valid test for testing of all getter methods	"Fortune, 1, 0"	1. “Name : Fortune” 2. “Location : Armory” 3. “Player rank : 1”
Valid test for testing of all setter methods	1. Update Location - 1	1. "Location - Billiard Room"
Invalid test for testing invalid/null arguments to setter methods	1. Null	IllegalArgumentException
Command classes implementation		
Valid test for creation of command using valid arguments	valid arguments for respective commands	"Command executed successfully"
Invalid test while creating a command for invalid/null arguments	Null	IllegalArgumentException