Sherly Hartono and Annanya Sah

Milestone 4

FOR UML please check in canvas for the pdf called UML.pdf

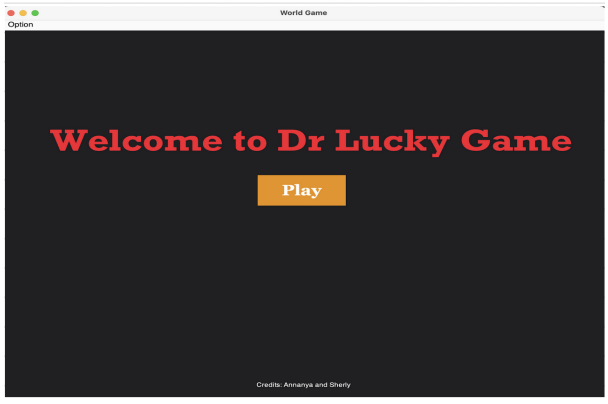
Summary of the changes that we needed to make the model to separate it from the text-based controller:

In order to make the Model decoupled from the text based controller, we introduced a read only model version of our facade(DefaultWorld) containing just the getters for information used in the view. ImageGenerator class used in the DefaultWorld is now extensively used by methods making state changes (players moving) and a new image is created with them. This is read by view through the readOnly model and refreshed by controller. To make the calculations easier to handle, we have added an unmodifiable list of spaces in the ImageGenerator class.

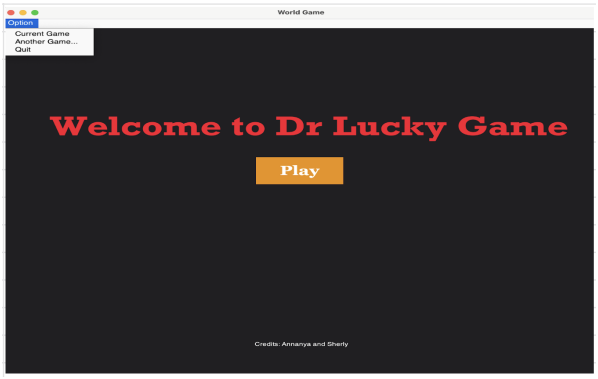
We changed the signature of action commands such as movePet(), move() and getPlayerInfo() to now take Points instead of String, these points are the pixel positions being clicked by the user on the view which then gets handled by the model.

UI

1. Welcome Panel

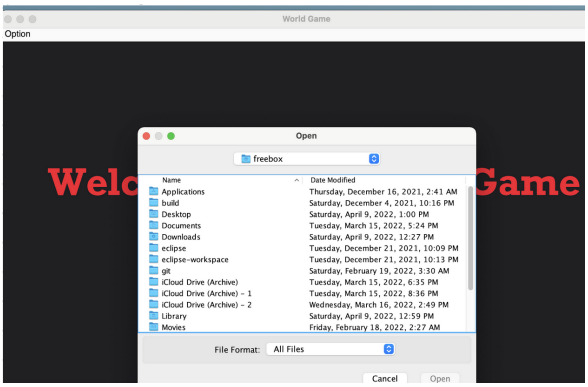


Click option →



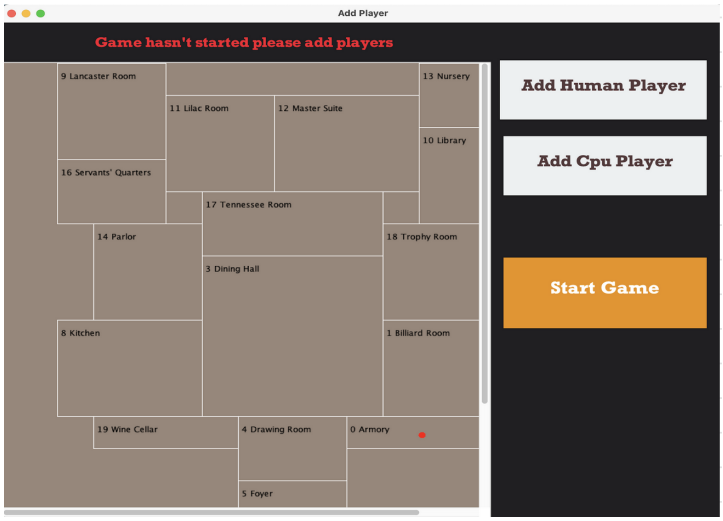
Click Another Game.. →

Will ask user to select files



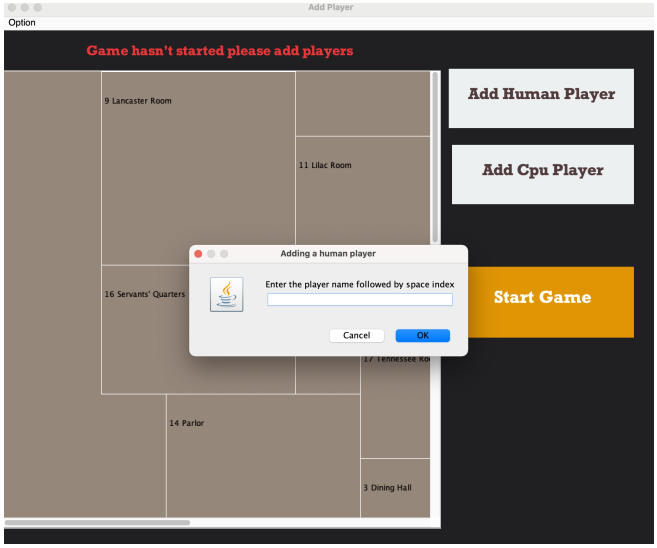
2. Game Panel

2.1 Adding player



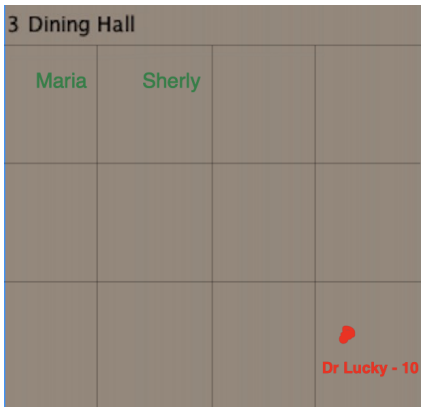
Click Add Human Player →

Will ask user to enter name and space index



Click Add Cpu Player →

Will directly add cpu name on the map



(the lines on the grid will not be actually drawn. It' just to indicate where each players will be)

Click Start Game → Will take you to play the game

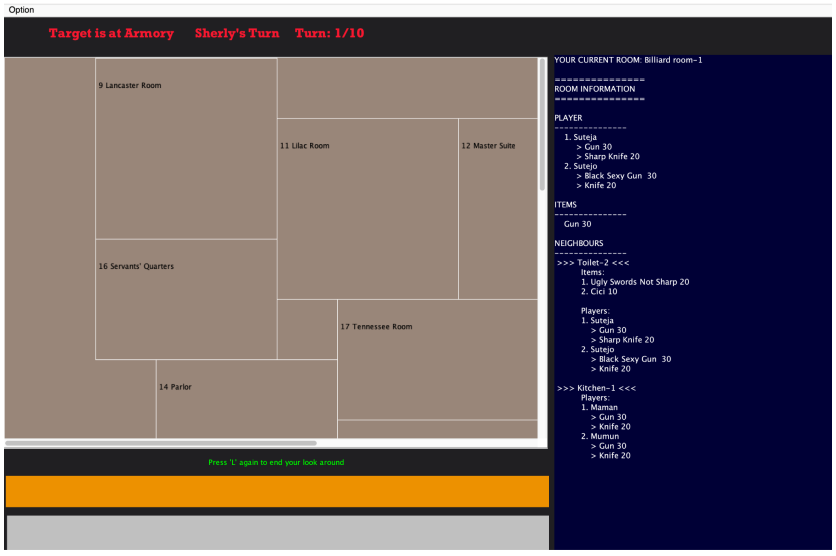
2.2 Playing game

- At the top highlighted in red are the current game status
- To the right are panels that will display information for:
  - Info at the start of every turn
  - Player's description
  - Look around info
- To the bottom in greens are prompts for user
- Results of user actions are shown in orange box
- Results of cpu actions are shown in gray box

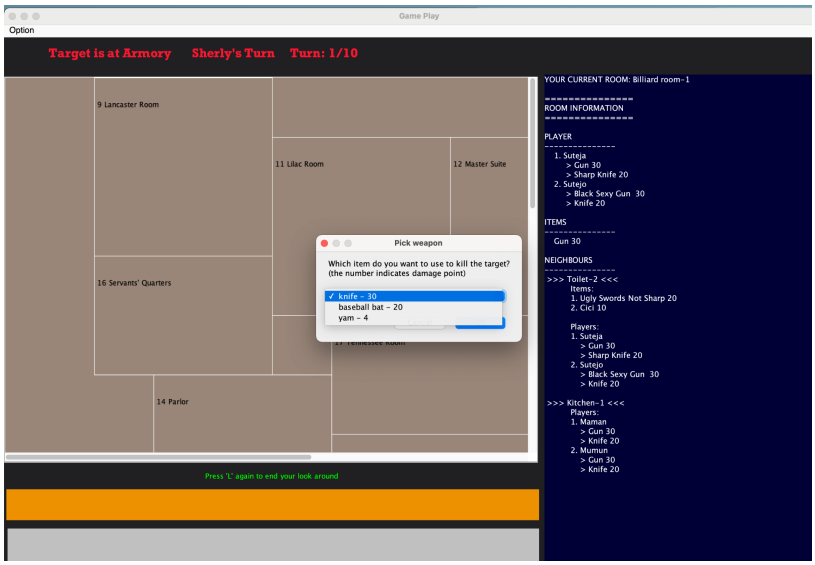
Features:

1. press 'C' will move the CPU player if it's Cpu's turn →

2. press 'L' will allow user to look around →



3. press 'I will pick an item or 'K' will prompt user for weapon →



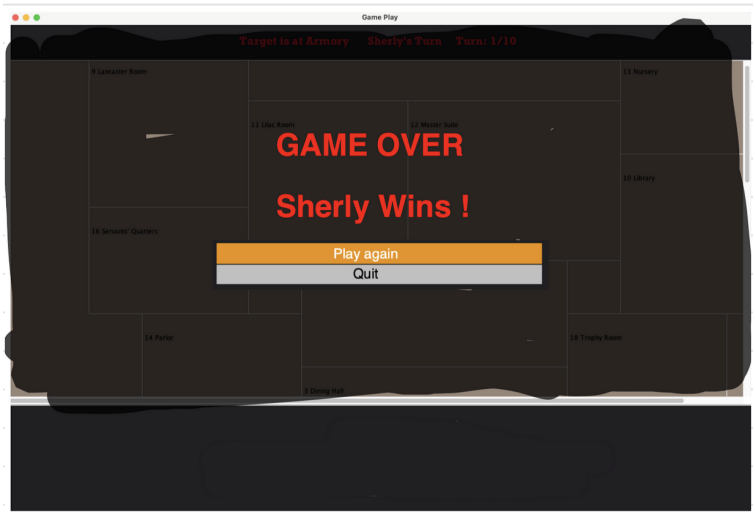
4. press 'P'' allows the current player to move a pet to a space. Then a dialogue will pop up

5. press 'Q' or selecting quit from menu will quit and close the window

6. Click on player's own icon in his turn will show the items he is carrying on the right panel

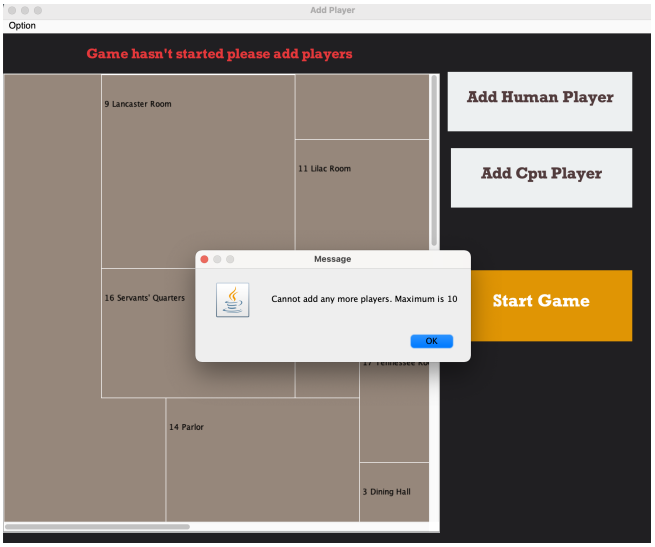
2.3 Game ending

User can also press 'Q' or select quit from menu to quit and close the window

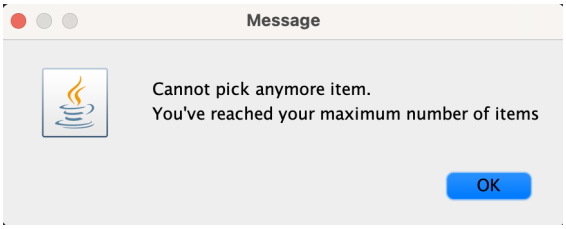


2.4 Other Dialogues

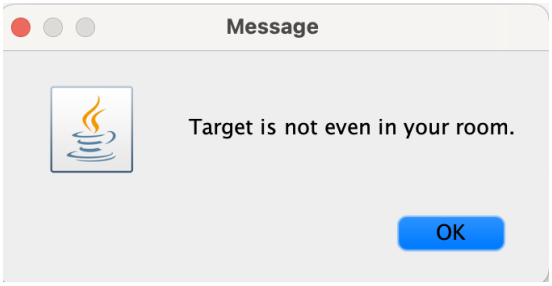
1. During adding player:



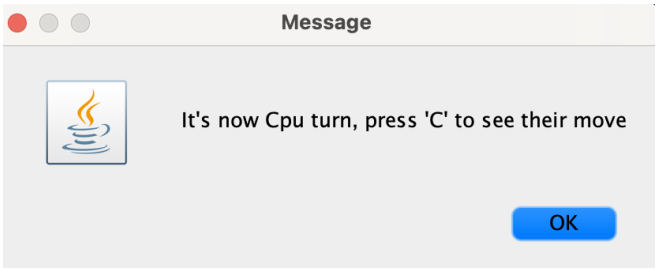
2. During game play
   - User maximum's item reached when he picks up an item 'I



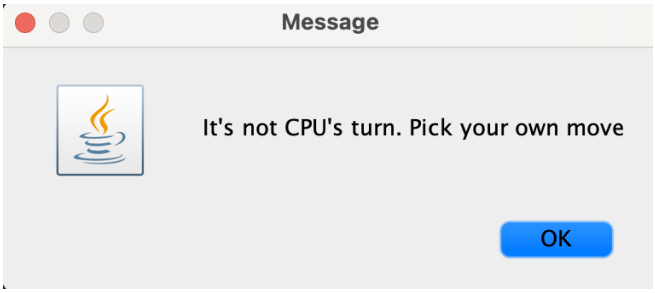
   - User try to kill 'K' but target is not there



   - User try to make any move, but its not their turn



- CPU making a move but its not their turn



Test Cases Milestone 4

1. Default World Class

Test the following new added methods

+ getPlayerInfo(Point): String
+ makeHumanMovePet(Point): String
+ makeHumanMove(Point) : String // need to add draw player
+ set(Readable, RandomIntGenerator, int, boolean)

1. Test getPlayerInfo(Point): String

Test valid arguments	**Input**	**Expected**
Player exist in this point	Point coord = new Point(2,4) Stringresult = world.getPlayerInfo(coord):	“Sherly is in Armory, max item: 5, items held: Gun - 20, Knife - 22”

Player does not exist	Point coord = new Point(2,4) String result = world.getPlayerInfo(coord):	“Sherly is in Armory, max item: 5, items held: Gun - 20, Knife - 22”
Point is null	Point coord =null String result = world.getPlayerInfo(coord):	Throws IAE(“No player in this coordinate”)

2. **makeHumanMove(Point): String**

Test valid arguments	**Input**	**Expected**
Player exist in this point	Point coord = new Point(2,4) List<String> result = world.makeHumanMove(coord):	“Sherly is in Armory, max item: 5, items held: Gun - 20, Knife - 22”
Player does not exist	Point coord = new Point(100,20) List<String> result = world.makeHumanMove(coord):	Throws IAE(“No player in this coordinate”)
Point is null	Point coord = null List<String> result = world.makeHumanMove(coord):	Throws IAE(“Cannot get info at null coordinate”)
Not a valid coordinate	Point coord = new Point(2,4) List<String> result = world.makeHumanMove(coord):	Throws IAE(“No player in this coordinate”)

3. **makeHumanMovePet(Point) : String**

Test valid arguments	**Input**	**Expected**
Valid move	Point coord = new Point(2,4) List<String> result = world.makeHumanMovePet(coord):	“Sherly have moved Fortune the Cat to Billiard Room”
Space does not exists	Point coord = new Point(100,20) List<String> result = world.makeHumanMovePet(coord):	Throw IAE(“Space does not exist in this point”)
Point is null	Point coord = null List<String> result = world.makeHumanMovePet(coord):	Throw IAE(“Point cannot be null”)

4. **set(Readable, RandomIntGenerator, int, boolean)**

Test valid arguments	**Input**	**Expected**
Valid world	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - in = new StringReader("15 16 Northeastern X Disney Land\n" + "50 Prof Jump"; + "Fortune the cat"; " 3 1 5 2 Archery Range\n" + " 0 2 2 4 West Village H\n" + " 0 0 2 2 Zombie VR"; "0 3 Suction Cup Arrows\n" + "1 10 Deadline Extension\n" + "2 2 Motion sick VR headset\n"); set(in, ranGen maxTurn , true);	Name: Northeastern X Disney Land Size: 15 X 16 Target: Doctor Lucky's health is 50 Target's Position: 0 Number of Rooms: 3 Number of Items: 3 Number of Players: 0 Pet: Fortune the cat Pet’s position: 0 maxTurn: 5

World name with numbers	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - Readable worldInfo = new StringReader(“Northeastern X 1 2 3 Disney Land\n" + "50 Prof Jump\n" + "50 Fortune the cat\n" + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n"); set(in, ranGen maxTurn , true);	Name: Northeastern X 1 2 3 Disney Land Size: 15 X 16 Target: Doctor Lucky's health is 50 Target's Position: 0 Number of Rooms: 3 Number of Items: 3 Number of Players: 0 Pet: Fortune the cat Pet’s position: 0 maxTurn: 5

Test invalid arguments	**Input**	**Expected**
No cat inserted	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - In = new StringReader("15 16 Northeastern X Disney Land\n" + "50 Prof Jump\n" + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n") set(in, ranGen maxTurn , true);	IllegalArgumentException

Test zero space	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - In = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the Cat\n" + " 0\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n") set(in, ranGen maxTurn , true);	IllegalArgumentException
Room index greater than total Space	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat” + " 2\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 4\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 4 2 Motion sick VR headset\n"); set(in, ranGen maxTurn , true);	IllegalArgumentException
Actual And Given Space Number Mismatch	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat” + " 2\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n" set(in, ranGen maxTurn , true);	IllegalArgumentException

Zero maximum turn	- RandomIntGenerator ranGen = new RandomIntGenerator(0, 2, 3, 10, 24, 20); - maxTurn = 0; - in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat” + " 2\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n" set(in, ranGen maxTurn , true);	IllegalArgumentException
Test item room index greater than total space	- RandomIntGenerator ranGen = new RandomIntGenerator(0, 2, 3, 10, 24, 20); - maxTurn = 0; - in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat” + " 2\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 4\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 4 2 Motion sick VR headset\n")" set(in, ranGen maxTurn , true);	IllegalArgumentExceptio

2. View Concrete Classes

Class Name	Test case	Actual	Expected
WorldMouseAdapter	Null event passed to mouseclick()	mouseClicked(null)	Throw IllegalStateException(“Unsupported event”)
	Valid mouse clickevent on mouseClick()	mouseClicked(Event s)	No errors, mouse click works
WorldMouseKeyListener	Trying to set null type key listerner	.setKeyListener(null)	Throw IllegalStateException(“Unsupported event”)

	Trying to call keyTyped() with null	.keyTyped(null)	Throw IllegalStateException(“Unsupported event”)
WorldButtonListener	Trying to call action performed with null	.actionPerformed(null)	Throw IllegalStateException(“Unsupported event”)
DefaultView	Trying to call setFeatures() with null	.setFeatures(nulll)	Throw IllegalStateException(“Unsupported feature”)
	Trying to call confugureKeyListener() with null	.configureKeyListener(null)	Throw IllegalStateException(“Unsupported listener”)
	Trying to call configureButtonListener() with null	configureButtonListener(null)	Throw IllegalStateException(“Unsupported listener”)
	Trying to call configureButtonListener() with null	configureButtonListener(null)	Throw IllegalStateException(“Unsupported listener”)
GameStatusPanel	Trying to setText as null	.setText(null)	Throw IllegalArgumentException(“Cannot set invalid string”)
	Trying to setText as “”	.setText(“”)	Throw IllegalArgumentException(“Cannot set invalid string”)
HowToPanel	Trying to setText as null	.setText(null)	Throw IllegalArgumentException(“Cannot set invalid string”)
	Trying to setText as “”	.setText(“”)	Throw IllegalArgumentException(“Cannot set invalid string”)
ActionResultPanel	Trying to setText as null	.setText(null)	Throw IllegalArgumentException(“Cannot set invalid string”)
	Trying to setText as “”	.setText(“”)	Throw IllegalArgumentException(“Cannot set invalid string”)

3. Controller class

We will test using two MockModel classes.
One that just logs the string to indicate that the controller’s call reached the model. Another one that always throws an exception.

We will test the following methods in the controller’s class

1. playGame(View) : void
2. addCpuPlayer(): void
3. addHumanPlayer(): void

4. displayPlayerInfo(): void
5. humanLook(): void
6. humanMove(: void
7. humanMovePet(): void
8. humanPickItem() : void
9. humanKill() : void
10. takeCpuTurn() : void
11. loadNewGame() : void
12. quitGame() : void
13. generateMap() : void

The before set up:

```
@Before
 public void setUp() {
      // 1. Create log for input and random outpu
      StringBuilder inlog = new StringBuilder();
      String outString = "1234567\n"; // check if it will output this

      // 2. Create two models
      MockModel mockModel = new MockWorldModel(inlog, outString);
      MockException mockExc = new MockWorldException(inlog, outString);

      // 3. Create mock view
      mockView = new MockView(inlog);

      // 4. Create controller using mocks
      cont = new DefaultController(mockModel, mockView);
      contException = new DefaultController(mockExc, mockView);

 }
```

1. **playGame() : void**

Test valid arguments	**Input**	**Expected**
Logs	cont.playGame(mockModel) String actual = inlog.toString()	“Playing game”
Exception	contExc.playGame(mockExc)	Throws IAE

2. **addCpuPlayer() : void**

Test valid arguments	**Input**	**Expected**

Logs	cont.addCpuPlayer() String actual = inlog.toString()	“Adding cpu player. Updating panels”
Exception	contExc.addCpuPlayer()	Throws IAE

3. **addHumanPlayer() : void**

Test valid arguments	**Input**	**Expected**
Logs	cont.addHumanPlayer() String actual = inlog.toString()	“Adding human player. Updating panels”
Exception	contExc.addHumanPlayer()	Throws IAE

4. **displayPlayerInfo() : void**

Test valid arguments	**Input**	**Expected**
Logs	cont.displayPlayerInfo() String actual = inlog.toString()	“Displaying player info. Updating panels”
Exception	contExc.displayPlayerInfo()	Throws IAE

5. **humanLook() : void**

Test valid arguments	**Input**	**Expected**
Logs	cont.humanLook() String actual = inlog.toString()	“Human player looking around. Updating panels”
Exception	contExc.humanLook()	Throws IAE

6. **humanMove() : void**

Test valid arguments	**Input**	**Expected**
Logs	cont.humanMove() String actual = inlog.toString()	“Human player moving. Updating panels”

Exception	contExc.humanMove()	Throws IAE

7. **humanMovePet(): void**

Test valid arguments	**Input**	**Expected**
Logs	cont.humanMovePet() String actual = inlog.toString()	“Human player moving pet. Updating panels”
Exception	contExc.humanMovePet()	Throws IAE

8. **humanPickItem() : void**

Test valid arguments	**Input**	**Expected**
Logs	cont.humanPickItem() String actual = inlog.toString()	“Human player picking itemUpdating panels”
Exception	contExc.humanPickItem()	Throws IAE

9. **humanKill() : void**

Test valid arguments	**Input**	**Expected**
Logs	cont.humanKill() String actual = inlog.toString()	“Human player attempt a kill. Updating panels”
Exception	contExc.humanKill()	Throws IAE

10. **takeCpuTurn() : void**

Test valid arguments	**Input**	**Expected**
Logs	cont.takeCpuTurn()	“Taking Cpu turn.. Updating panels”

	String actual = inlog.toString()	
Exception	contExc.takeCpuTurn()	Throws IAE

11. loadNewGame() : void

Test valid arguments	**Input**	**Expected**
Logs	cont.loadNewGame() String actual = inlog.toString()	“Load new game. Updating panels”
Exception	contExc.loadNewGame()	Throws IAE

12. quitGame() : void

Test valid arguments	**Input**	**Expected**
Logs	cont.quitGame() String actual = inlog.toString()	“quit game”
Exception	contExc.quitGame()	Throws IAE

13. generateMap() : void

Test valid arguments	**Input**	**Expected**
Logs	cont.generateMap() String actual = inlog.toString()	“GenerateMap”
Exception	contExc.generateMap()	Throws IAE

Milestone 1-3: Testing Suite

1. DefaultWorld

Requirement changes:

- The pet enters the game in the same space as the target character.
- Have user to be able to attempt to kill target character
- Game end rules:
    - A player successfully kills the target character in which case they win the game.
    - The maximum number of turns is reached in which case the target character escapes and runs away to live another day and nobody wins.

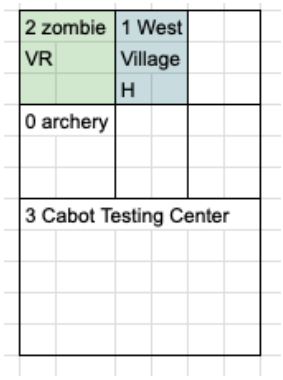
First we will instantiate worldImplObject using a helper method
World worldDisnu = worldHelper(Readable in, RandomIntGenerator ranGen, int maxTurn);

We will pass in this argument in the setUp() method:

```
- in = new StringReader(
   "15 16 Northeastern X Disney Land\n"
   + "50 Prof Jump\n"
   + " 4\n"
   + " 3  1  5  2 Archery Range\n"
   + " 0  3  2  4 West Village H\n"
   + " 0  0  2  2 Zombie VR\n"
   + " 6  1 10  8 Cabot Testing Center\n"
   + " 3\n"
   + " 0 3 Suction Cup Arrows\n"
   + " 1 10 Deadline Extension\n"
   + " 2 2 Motion sick VR headset\n");

- ranGen = randomIntGenHelper(5, 2, 3);
- worldDisnu = worldHelper(in, ranGen, 5);

   0-2-1-2-0-3-0
```



0 - Archery Range
item: Suction cup arrow ,**damage point=3**

1 - West Village H
item: deadline extension, **damage point=10**

2 - Zombie VR
item: motion sick VR headset, **damage point=2**

3 - Cabot Testing Center

**1.1 and 1.2 Test constructor and `toString()` with valid arguments

Test valid arguments	**Input**	**Expected**
Valid world	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - in = new StringReader("15 16 Northeastern X Disney Land\n" + "50 Prof Jump"; + "Fortune the cat"; " 3 1 5 2 Archery Range\n" + " 0 2 2 4 West Village H\n" + " 0 0 2 2 Zombie VR"; "0 3 Suction Cup Arrows\n" + "1 10 Deadline Extension\n" + "2 2 Motion sick VR headset\n");	Name: Northeastern X Disney Land Size: 15 X 16 Target: Doctor Lucky's health is 50 Target's Position: 0 Number of Rooms: 3 Number of Items: 3 Number of Players: 0 Pet: Fortune the cat Pet's position: 0 maxTurn: 5
Large world	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - Readable worldInfo = new StringReader("200 X 200Northeastern X Disney Land\n" + "50 Prof Jump\n" + "50 Fortune the cat\n" + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n");	Name: Northeastern X Disney Land Size: 200 X 200 Target: Doctor Lucky's health is 50 Target's Position: 0 Number of Rooms: 3 Number of Items: 3 Number of Players: 0 Pet: Fortune the cat Pet's position: 0 maxTurn: 5
Long world name	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - Readable worldInfo = new StringReader("Northeastern Northeastern Northeastern X Disney Land\n" + "50 Prof Jump\n" + "50 Fortune the cat\n" + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n"	Name: Northeastern Northeastern Northeastern X Disney Land Size: 200 X 200 Target: Doctor Lucky's health is 50 Target's Position: 0 Number of Rooms: 3 Number of Items: 3 Number of Players: 0 Pet: Fortune the cat Pet's position: 0 maxTurn: 5

	+ " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n");	
World name with numbers	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - Readable worldInfo = new StringReader(“Northeastern X 1 2 3 Disney Land\n" + "50 Prof Jump\n" + "50 Fortune the cat\n" + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n");	Name: Northeastern X 1 2 3 Disney Land Size: 15 X 16 Target: Doctor Lucky's health is 50 Target's Position: 0 Number of Rooms: 3 Number of Items: 3 Number of Players: 0 Pet: Fortune the cat Pet’s position: 0 maxTurn: 5

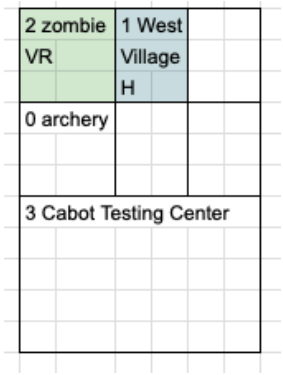
Test invalid arguments	**Input**	**Expected**
No cat inserted	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - In = new StringReader("15 16 Northeastern X Disney Land\n" + "50 Prof Jump\n" + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n")	IllegalArgumentException

Test zero space	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - In = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the Cat\n" + " 0\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n")	IllegalArgumentException
Room index greater than total Space	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat” + " 2\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 4\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 4 2 Motion sick VR headset\n");	IllegalArgumentException
Actual And Given Space Number Mismatch	- RandomIntGenerator ranGen = new RandomIntGenerator(5, 2, 3, 10, 24, 20); - maxTurn = 5; - in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat” + " 2\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n"	IllegalArgumentException
Zero maximum turn	- RandomIntGenerator ranGen = new RandomIntGenerator(0, 2, 3, 10, 24, 20); - maxTurn = 0; - in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat”	IllegalArgumentException

	+ " 2\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n"	
Test item room index greater than total space	- RandomIntGenerator ranGen = new RandomIntGenerator(0, 2, 3, 10, 24, 20); - maxTurn = 0; - in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat” + " 2\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 4\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 4 2 Motion sick VR headset\n")"	IllegalArgumentException

**1.3 getSpaceInfo()

Requirement changes: this command should only be a utility command for milestone 3. Once we included the View model, we should exclude this command once the game has started.



0 - Archery Range
1 - West Village H
2 - Zombie VR
3 - Cabot Testing Center

Test arguments	**Input**	**Expected**
1 neighbor	worldHelper(in, ranGen, maxTurn); worldDisnu.getSpaceInfo("Archery Range")	"0 Space name: Archery Range\n" + "======Items======:\n" + "-Item name: Suction Cup Arrows\n" + "Item Index: 0\n" + "Damage Point: 3\n" + "\n" + "======Neighbors==:\n" + "\n" + "% 3 Zombie VR %\n" + "neighbor's item:\n" + "-Item name: Motion Sick VR Headset\n" + "Item Index: 2\n" + "Damage Point: 2\n" + "neighbor's players:\n" + "-Player name: Sherly\n" + "current space: Zombie VR\n" + "max item: 5\n" + "items held:\n" + "[]\n" + "\n" + "\n" + "% 3 Cabot Testing Center %\n" + "neighbor's item:\n" + "neighbor's players:\n" + "\n" + "\n" + "======Players==:\n";
2 neighbors	worldDisnu.getSpaceInfo("Zombie VR");	"2 Space name: Zombie VR\n" + "Neighbors:\n" + "-Archery Range\n"

		+ "-West Village H\n" + "\n" + "Items:\n" + "-Item name: Motion Sick VR Headset\n" + "Room Index: 2\n" + "Damage Point: 2\n Players: "
No such space name	worldDisnu.getSpaceInfo("non existent space");	“No such name exists”

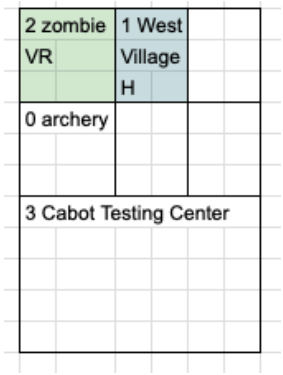
****1.4 makeHumanAttemptKill(int)**

- Players should be able to make an attempt on the target character's life if they are in the same space as the target character. *This represents a turn.*
    - A player who does not have any items, can always make an attempt on the target character's life by "poking him in the eye" which does 1 point of damage
- The results of an attempt are:
    - If an attack is *seen* by another player (human or computer), it is automatically stopped and no damage is done.
    - *Unseen* attacks are always successful in which case the appropriate number of hit points (determined by the item used in the attempt) is removed from the target character.

I design so that the item index starts from 1. If user doesn't want to use any weapon. Enter 0 to poke eye

Things to check: target’s health before and after. Weapon is removed from current player

Sample world:



0 - Archery Range
item: Suction cup arrow ,**damage point=3**

1 - West Village H
item: deadline extension, **damage point=10**

2 - Zombie VR
item: motion sick VR headset, **damage point=2**

3 - Cabot Testing Center

Test arguments	**Input**	**Expected**
Attempt successful - use weapon	worldDisnu.addHumanPlayerToList("Sherly", 1); // target is at 0, player at 1 // human pick deadline extension worldDisnu.makeHumanPickItem("Sherly", 1); // target is at 1, player at 1 String expected = worldDisnu.makeHumanAttemptToKill(1);	“Sherly has successfully attempted a murder, using arrow target's health is now 47. Sherly now hold these items:\n -”
Attempt failed- target not in same space	worldDisnu.addHumanPlayerToList("Sherly", 1); // target is at 0, player in 0; // human pick deadline extension String expected = worldDisnu.makeHumanAttemptToKill();	“Target is not in your current space”
Attempt successful - Poke eye	worldDisnu.addHumanPlayerToList("Sherly", 0); // target is at 0, player in 0; // human pick deadline extension String expected = worldDisnu.makeHumanAttemptToKill(0);	“Sherly has successfully attempted a murder by poking target’s eyes, target's health is now 49. Sherly now hold these items:\n -”
Attempt failed - seen by CPU player in rooms with pet	// CPU in room 2 worldDisnu.addCPUPlayerToList(); worldDisnu.addHumanPlayerToList("Sherly", 1);	“Attempt failed someone can see you. Target health is still 50. Sherly now hold these items: -”

	// CPU move pet to room 2 worldDisnu.makeCpuTakeTurn(); // CPU in room 2, pet in room 2, // Human in 1, target in room 1 worldDisnu.makeHumanAttemptToKill(1);	
Attempt failed attempt seen by Human player in rooms with pet	// Human opponent in room 2 worldDisnu.addHumanPlayerToList("Ai", 2); worldDisnu.addHumanPlayerToList("Sherly", 1); // Ai picked up item // target room 0 worldDisnu.makeHumanPickItem(0); // Sherly move pet to room 1 // target room 1 worldDisnu.makeHumanMovePet(1); // Ai make kill attempt // target room 2, Ai room 2, Sherly room 1, Pet room 1 worldDisnu.makeHumanAttemptToKill(1);	"Attempt failed someone can see you. Target health is still 50. Ai now hold these items: -"

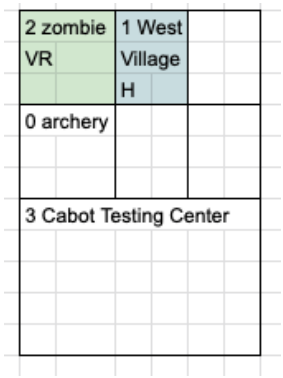
****1.5 makeHumanMovePet(int)**

- Players actions should include moving the pet to a specified space. *This represents a turn.*
- The pet can be moved to any space in the world (not necessarily neighbors).
- The pet can be moved by any player even when they are not in the same space as the pet. So moving pet is always successful unless its not a user turn

Test arguments	**Input**	**Expected**
Move Pet successful	worldDisnu.addHumanPlayerToList("Sherly", 1); // target is at 1, player at 1 String expected = worldDisnu.makeHumanMovePet(2);	"Fortune the cat have moved from Archery Range to Zombie VR."
Move Pet not their turn	worldDisnu.addCpuPlayerToList("Sherly", 1); worldDisnu.addHumanPlayerToList("Sherly", 1); // target is at 1, player at 1 String expected = worldDisnu.makeHumanMovePet(2);	"Moving pet failed. Its currently a CPU turn."

**1.6 makeCpuTakeTurn()

- Computer-controlled player always chooses to make an attempt on the target character's life (if they cannot be *seen* by others) using the item in their inventory that does the most damage.
    - A player who does not have any items, can always make an attempt on the target character's life by "poking him in the eye" which does 1 point of damage
- The results of an attempt are:
    - If an attack is *seen* by another player (human or computer), it is automatically stopped and no damage is done.
    - *Unseen* attacks are always successful in which case the appropriate number of hit points (determined by the item used in the attempt) is removed from the target character.



0 - Archery Range
item: Suction cup arrow ,**damage point=3**

1 - West Village H
item: deadline extension, **damage point=10**

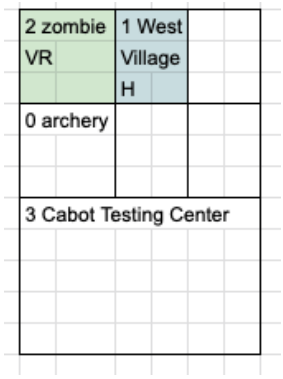
2 - Zombie VR
item: motion sick VR headset, **damage point=2**

Test arguments	**Input**	**Expected**
Successful attempt- poke eye	// make CPU in same space as target worldDisnu.addCpuPlayerToList(); // target is at 0, player in 0; String expected = worldDisnu.makeCpuTakeTurn();	“CPU_xyz has successfully attempted a murder by poking target’s eyes, target's health is now 49. CPU_xyz now hold these items:\n -”

Failed attempt seen by human with pet	worldDisnu.addHumanPlayerToList(2); // CPU in room 1 worldDisnu.addCpuPlayerToList(); worldDisnu.makeHumanMovePet(2); // target is at 1, cpu player in 1, human player in 2 with pet; String expected = worldDisnu.makeCpuTakeTurn();	"CPU_xyz failed to attempt murder with eye poke. Target health is still 50. CPU_xyz now hold these items:\n -"
CPU does not attempt murder if it's not in same space as target	worldDisnu.addCpuPlayerToList(); // target is at 0, player in 2; String expected = worldDisnu.makeCpuTakeTurn();	"CPU up picked deadline extension"
CPU move pet	worldDisnu.addCpuPlyaerToList(2); String expected = worldDisnu.makeCpuTakeTurn();	"CPU have successfully moved pet to room 1"
Attempt murder using max weapon	worldDisnu.addCpuPlayerToList(); // add to room 3 // make CPU picked up 3 items using random integer generator worldDisnu.makeCpuTakeTurn(); // target in room 0 worldDisnu.makeCpuTakeTurn(); // target in room 1 worldDisnu.makeCpuTakeTurn(); // target in room 2 String expected = worldDisnu.makeCpuTakeTurn(); // target in room 3	"CPU have successfully attempted murder with deadlin extension. Target health is now 40. CPU_xyz now hold these items: - Suction cup arrow - Motion Sick VR headset"
Test CPU Look around	worldDisnu.addCpuPlayerToList(); String expected = worldDisnu.makeCpuTakeTurn();	"CPU took a look around"

**1.7 makeHumanMove(Space space)

Requirement changes: human player can move to a room with a pet



0 - Archery Range
1 - West Village H
2 - Zombie VR
3 - Cabot Testing Center

Test arguments	**Input**	**Expected**
Test moving to a room with a pet	// pet is at position 2 worldDisnu.addHumanPlayerToList("Sherly", 0); worldDisnu.makeHumanMovePet(Pet,2); worldDisnu.makeHumanMove(2); // check if player is displayed from neighbor worldDisnu.getSpaceInfo("Archery Range")	→ "Sherly has successfuly moved to Zombie VR\n" →"0 Space name: Archery Range\n" + "======Items======:\n" + "-Item name: Suction Cup Arrows\n" + "Item Index: 0\n" + "Damage Point: 3\n" + "\n" + "======Neighbors==:\n" + "\n" + "% 3 Cabot Testing Center %\n" + "neighbor's item:\n" + "neighbor's players:\n" + "\n" + "\n" + "======Players==:\n";

Valid move	// player move from 0 to 2 worldDisnu.addHumanPlayerToList("Sherly", 0); worldDisnu.makeHumanPlayerMove(2); worldDisnu.getSpaceInfo("Archery Range") // space 0 should have no Sherly	→ "Sherly has successfuly moved to Zombie VR\n" →"0 Space name: Archery Range\n" + "======Items======:\n" + "-Item name: Suction Cup Arrows\n" + "Item Index: 0\n" + "Damage Point: 3\n" + "\n" + "======Neighbors==:\n" + "\n" + "% 2 Zombie VR %\n" + "neighbor's item:\n" + "-Item name: Motion Sick VR Headset\n" + "Item Index: 2\n" + "Damage Point: 2\n" + "neighbor's players:\n" + "-Player name: Sherly\n" + "current space: Zombie VR\n" + "max item: 5\n" + "items held:\n" + "[]\n" + "\n" + "\n" + "% 3 Cabot Testing Center %\n" + "neighbor's item:\n" + "neighbor's players:\n" + "\n" + "\n" + "======Players==:\n" + Target is here\n”;
Moving human with no player	worldDisnu.makeHumanPlayerMove(1);	"There is no player in the game or max turn reached " + "or current player turn is a cpu\n"
Moving human to space that is not a neighbor	worldDisnu.makeHumanPlayerMove(1);	"Sherly has failed to move because space is not neighbor"; // check status make sure target and turn has not changed "Current turn: 1, game's max turn: 5, target's space: 0\n" + "Whose turn: Sherly, player index: 0\n"

Moving human when it's cpu's turn	worldDisnu.addCpuPlayerToList(); worldDisnu.addHumanPlayerToList("Sherly", 0); worldDisnu.makeHumanPlayerMove(2);	"There is no player in the game or max turn reached " + "or current player turn is a cpu\n"

**1.8 makeHumanLook()

From milestone 2:

- update the *look around* command to make it worth using it to take a turn. The current player should only be able to *look around.*

the space that they are currently in. They cannot teleport to another area of the game and look around that space. If you imagine what you would be able to see if you were standing in a room in the real world, then:

- you would know where you were (the name of the space), who was in the room with you (the other players), what items were laying around the space, as well as information about other spaces that you could see from there.
- if you moved around the room, you would be able to *see* into neighboring spaces to be able to identify what space it was as well as what players and items were in the neighboring space. You might even be able to see what rooms were visible (but not necessarily).

Requirement changes:

- Any space that is occupied by the pet *cannot be seen* by its neighbors making it virtually invisible to the user.

Test valid arguments	**Input**	**Expected**
Look around-neighbors no pet	in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat” + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n" // Target and pet will be in Archery Range at the start of the game worldDisnu.addHumanPlayerToList("Sherly", 2); world1.makeHumanMovePet(1); world1.makeHumanLook();	"3 Space name: Zombie VR\n" + "======Items======:\n" + "-Item name: Motion Sick VR Headset\n" + "Item Index: 2\n" + "Damage Point: 2\n" + "\n" + "======Neighbors==:\n" + "\n" + "% 0 Archery Range %\n" + "neighbor's item:\n" + "-Item name: Suction Cup Arrows\n" + "Item Index: 0\n" + "Damage Point: 3\n" + "neighbor's players:\n" + "\n" + "% 1 West Village H %\n" + "neighbor's item:\n" + "-Item name: Deadline Extension\n" + "Item Index: 1\n" + "Damage Point: 10\n" + "neighbor's players:\n" + "\n" + "\n" + "======Players==:\n"

Look around-neighbor with pet	in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + “Fortune the cat” + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n" // Target and pet will be in Archery Range (0) at the start of the game // if we get check Zombie VR (room 2) we should not see Archery Range worldDisnu.addHumanPlayerToList("Sherly", 2); world1.makeHumanLookAround();	"3 Space name: Zombie VR\n" + "======Items======:\n" + "-Item name: Motion Sick VR Headset\n" + "Item Index: 2\n" + "Damage Point: 2\n" + "\n" + "======Neighbors==:\n" + "% 1 West Village H %\n" + "neighbor's item:\n" + "-Item name: Deadline Extension\n" + "Item Index: 1\n" + "Damage Point: 10\n" + "neighbor's players:\n" + "\n" + "\n" + "======Players==:\n"
Pet is in current space	worldDisnu.addHumanPlayerToList("Sherly", 0); worldDisnu.makeHumanLook();	3 Space name: Zombie VR\n" + "======Items======:\n" + "-Item name: Motion Sick VR Headset\n" + "Item Index: 2\n" + "Damage Point: 2\n" + "\n" + "======Neighbors==:\n" + "% 1 West Village H %\n" + "neighbor's item:\n" + "-Item name: Deadline Extension\n" + "Item Index: 1\n" + "Damage Point: 10\n" + "neighbor's players:\n" + "\n" + "\n" + "======Players==:\n" + “Fortune the cat is here”
Look around no neighbors	in = new StringReader("500 500 Northeastern X Disney Land\n" + "50 Prof Jump\n" + " 2\n" + " 3 1 5 2 Archery Range\n"); worldHelper(in, ranGen, 5); worldDisnu.makeHumanPlayerLook();	"0 Space name: Archery Range\n" + "======Items======:\n" + "-Item name: Suction Cup Arrows\n" + "Item Index: 0\n" + "Damage Point: 3\n" + "\n" + "======Neighbors==:\n" + "======Players==:\n";

**1.9 isGameOver()

Returns true if maximum turn has been reached

- Changes for milestone 3 : Game is over if target health is 0

**1.10 getStatus()

This is the string that is displayed to user at every turn:

- The location of the target so that it satisfies this requirement: the user has some way of knowing where the target player is in the world.
- the user is provided with some limited information about where they are in the world at the start of their turn.

Testing both is GameOver and getStatus

We'll make target's health = 2 so it dies quickly.

Test valid arguments	**Input**	**Expected**
Test computer wins	worldDisnu.addCpuPlayerToList(); // add CPU to room 1 worldDisnu.addHumanPlayerToList("Sherly", 0); in = new StringReader("15 16 Northeastern X Disney Land\n" + "2 Prof Jump\n" + "Fortune the cat\n" + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n"); worldDisnu.makeHumanAttemptKill(0); worldDisnu.makeCpuTakeTurn(); worldDisnu.isGameOver() String expected = worldDisnu.getCurrentStatus()	→ true → "Game is over, CPU_xyz wins by poking target in the eye"
Test human wins- by poke	worldDisnu.addCpuPlayerToList(); // add CPU to room 1 worldDisnu.addHumanPlayerToList("Sherly", 0); // target health 1 worldDisnu.makeCpuTakeTurn();	→ true → "Game is over, Sherly wins by poking target in the eye"

	// terget health 0 String expected = worldDisnu.makeHumanAttemptKill(0); worldDisnu.isGameOver() worldDisnu.getCurrentStatus()	
Test human wins- by weapon	// make target health 10 worldDisnu.addCpuPlayerToList(); // add CPU to room 1 worldDisnu.addHumanPlayerToList("Sherly", 0); // target health 9 worldDisnu.makeCpuTakeTurn(); String expected = worldDisnu.makeHumanPickItem(2); // cpu look around worldDisnu.makeCpuTakeTurn(); // player attempt to kill with weapon index 2 worldDisnu.makeHumanAttemptKill(2); worldDisnu.isGameOver() worldDisnu.getCurrentStatus()	→ true → “Game is over, Sherly wins by asking for deadline extension”
Test no one wins	// add CPU to room 1 worldDisnu.addCpuPlayerToList(); worldDisnu.addHumanPlayerToList("Sherly", 0); // target in room 0 worldDisnu.makeCpuTakeTurn(); // target in room 1 worldDisnu.makeHumanLook(); // target in room 2, cpu in room 1, cpu picks up item worldDisnu.makeCpuTakeTurn(); // target in room 3, human pick uip item worldDisnu.makeHumanPickItem(); // target in room 4 worldDisnu.isGameOver() worldDisnu.getCurrentStatus()	→ true → “Game is over, Nobody killed the target!”
Test get status at every user’s turn	// add CPU to room 1 worldDisnu.addHumanPlayerToList("Sherly", 0); worldDisnu.getStatus();	"Current turn: 1, game's max turn: 3, target's space: 0\n" + "Whose turn: Sherly, player index: 0\n. + “You are now in: 0”

		+ “Space name: Archery Range\n" + "======Items======:\n" + "-Item name: Suction Cup Arrows\n" + "Item Index: 0\n" + "Damage Point: 3\n" + "\n" + "======Players==:\n" + "-Player name: CPU_abcde\n" + "current space: Archery Range\n" + "max item: 3\n" + "items held:\n" + "[]\n" + "-Player name: CPU_fghij\n" + "current space: Archery Range\n" + "max item: 3\n" + "items held:\n" + "[]\n" + "-Player name: Sherly\n" + "current space: Archery Range\n" + "max item: 2\n" + "items held:\n" + "[]\n" + Target Character is here” + Fortune the car is here”
Status after 3 player turns	// add 1 in = new StringReader("15 16 Northeastern X Disney Land\n" + "50 Prof Jump\n" + "Fortune the cat\n" + " 4\n" + " 3 1 5 2 Archery Range\n" + " 0 3 2 4 West Village H\n" + " 0 0 2 2 Zombie VR\n" + " 6 1 10 8 Cabot Testing Center\n" + " 3\n" + " 0 3 Suction Cup Arrows\n" + " 1 10 Deadline Extension\n" + " 2 2 Motion sick VR headset\n"); ranGen = randomIntGenHelper(3, 0, 1, 2, 3, 4, 0, // 1st cpu max item 3, start from space 0 3, 5, 6, 7, 8, 9, 0, // 2nd cpu max item 3, start from space 0 2, // human player max item 0, 0, 0, // get 0th neighbor, get 0th item, random move 0. pick item 0, 1, 1); // get 1th neighbor, get 0th item, random move 0. move neighbor worldDisnu = worldHelper(in, ranGen, 3);	"Current turn: 3, game's max turn: 3, target's space: 2\n" + "Whose turn: Sherly, player index: 2\n"

	worldDisnu.addCpuPlayerToList(); worldDisnu.addCpuPlayerToList(); worldDisnu.addHumanPlayerToList("Sherly", 0); worldDisnu.makeCpuTakeTurn(); // turn 1 worldDisnu.makeCpuTakeTurn(); // turn 2	

1.11 makeHumanPlayerPickItem(int itemIndex)

Human player can pick an item that is present in his current space. This represent a turn.

Test arguments	**Input**	**Expected**
Valid	worldDisnu.addHumanPlayerToList("Sherly", 0); Item banana = itemHelper("Banana", 6, 1, 3); String actual = worldDisnu.makeHumanPlayerPickItem(0);	"Sherly has successfuly picked\n" + "Item name: Suction Cup Arrows\n" + "Item Index: 0\n" + "Damage Point: 3"
More than max item	worldDisnu.makeHumanPlayerPickItem(0); worldDisnu.makeHumanPlayerPickItem(1); String actual = worldDisnu.makeHumanPlayerPickItem(2);	"Sherly has failed to pick item. " + "Either item is not in space, or maximum item held is reached"

Pick up item not present	String actual worldDisnu.makeHumanPlayerPickItem(20);	“Sherly failed to pick item. " + "Either item is not in space, or maximum item held is reached

1.12 getTargetRoomIdx() and moveTargetCharc()

Requirements:

- Start with space index 0 (already covered in previous test)
- Move character up to 1 space (check with toString())
- Cannot move more than the number of Spaces minus 1 IllegalStateException

Test valid arguments	**Input**	**Expected**
Move once	displayTargetRoomIdx(); moveTargetCharc(); - once displayTargetRoomIdx();	target is at room: 0 Archery Range target is at room: 1 West Village H
Move multiple times and go back to room 0	getTargetRoomIdx(); moveTargetCharc(); getTargetRoomIdx(); moveTargetCharc(); getTargetRoomIdx(); moveTargetCharc(); getTargetRoomIdx(); moveTargetCharc();	0 1 2 0

1.13 getMaxTurn()

Test arguments	**Input**	**Expected**
Valid - small numbers	worldHelper(in, ranGen, 4); worldDisnu.getMaxTurn()	4

Valid - big numbers	worldHelper(in, ranGen, 100); worldDisnu.getMaxTurn()	100

1.14 getPlayerInfo()

Will return the player info at index thats

Test arguments	**Input**	**Expected**
Valid - carry no item	worldHelper(worldInfo, spacesInfo, itemsInfo) worldDisnu.getPlayerInfo(“Sherly”)	“Player name: Sherly Current space: Living Room Item held:”
Valid - carry multiple items	worldHelper(worldInfo, spacesInfo, itemsInfo) worldDisnu.getPlayerInfo(“Sherly”)	“Player name: Sherly Current space: Living Room Item held: Knife Baseball bat Covid virus Syringe”
Valid - carry 1 item	worldHelper(worldInfo, spacesInfo, itemsInfo) worldDisnu.getPlayerInfo(“Sherly”)	“Player name: Sherly Current space: Living Room Item held: Knife”

1.15 addPlayerToList()

To add a cpu just take a “cpu” string
Will return a string of the added player

Test arguments	Input	Expected
Valid - add 1 player	worldHelper(worldInfo, spacesInfo, itemsInfo) worldDisnu.addPlayerToList(“Sherly\nLiving Room”)	“Player name: Sherly Current space: Living Room Item held: Knife Baseball bat Covid virus Syringe”
Valid - add 2 players	worldHelper(worldInfo, spacesInfo, itemsInfo) worldDisnu.addPlayersToList(“Sherly\nLiving Room”) worldDisnu.addPlayersToList(“cpu”)	“Player name: Sherly Current space: Living Room Item held: Knife Baseball bat Covid virus Syringe” “Player name: some random Current space: Zombie VR Item held: Basketball”
Add human player toInvalid space	worldHelper(worldInfo, spacesInfo, itemsInfo) worldDisnu.addPlayersToList(“Sherly\nLiving”)	“Space name does not exists”
Add cpu player toInvalid space	worldHelper(worldInfo, spacesInfo, itemsInfo) worldDisnu.addPlayersToList(“cpu”)	“Space name does not exists”
Empty argument	worldHelper(worldInfo, spacesInfo, itemsInfo) worldDisnu.addPlayersToList(“”)	“Empty string is invalid”

2. DefaultTargetCharc

Will add the following methods to support new mileston requirement as now target character can be attacked:

+ reduceHealth(int) : void
+ isDead() : boolean
+ getCurrentSpaceIdx(): int
+ move(): void

If damage point is greater than target ‘s health it will just set the health to 0.

2 - Zombie VR
Item 3, motion sick VR headset, **damage point=2**

3 - Cabot Testing Center

**2.1 reduceHealth() and getHealth()

Test arguments	**Input**	**Expected**
Reduce by 10	Target unlucky = targetHelper(“Prof Unlucky”, 50); unlucky.reduceHealth(10) int health = unlucky.getHealth()	40
Reduce by 60	Target unlucky = targetHelper(“Prof Unlucky”, 50); unlucky.reduceHealth(60) int health = unlucky.getHealth()	0
Reduce by 0	Target unlucky = targetHelper(“Prof Unlucky”, 50); unlucky.reduceHealth(0) int health = unlucky.getHealth()	0

**2.2 isDead()

Test arguments	**Input**	**Expected**
dead	Target unlucky = targetHelper(“Prof Unlucky”, 1); player1.attemptToKill(unlucky) Boolean isdead = unlucky.isDead()	true
Not dead attack with weapon	Target unlucky = targetHelper(“Prof Unlucky”, 20); player1.attemptToKill(unlucky, 1) // use weapon of damage point 3	false

	Boolean isdead = unlucky.isDead()	

2.3 getCurrentSpaceIndex()

Test arguments	**Input**	**Expected**
Valid not move	Target unlucky = targetHelper(“Prof Unlucky”, 100); unlucky.getCurrentSpaceIndex()	0
Move once	targetHelper(“Li”, 100); unlucky.move() unlucky.getCurrentSpaceIndex()	1

2.5 Constructor

2.6 toString()

2.1 and 2.2 Test constructor and TargetCharacter’s `toString()`

Test arguments	**Input**	**Expected**
Start health valid	targetHelper(“Prof Unlucky”, 100);	“Prof Unlucky’s health is 100”
Short character names	targetHelper(“Li”, 100);	Li’s health is 100”
Long Charc names	targetHelper(“Prof Takayamapemi Sparkskull Tokagepemi”, 100);	“Prof Takayamapemi Sparkskull Tokagepemi’s health is 100”
Small Health	targetHelper(“Prof Unlucky”, 1);	“Prof Unlucky’s health is 1”
Large Health	targetHelper(“Prof Unlucky”, 19999);	“Prof Unlucky’s health is 19999”

Test invalid arguments	**Input**	**Expected**
Empty name	targetHelper(“”, 100);	IllegalArgumentException
Health is 0	targetHelper(“Prof Unlucky”, 0);	IllegalArgumentException
Health is negative	targetHelper(“Prof Unlucky”, -1);	IllegalArgumentException

2.7 equals()

Test arguments	**Input**	**Expected**
Same character	**// test new object vs previously created object** assertTrue(targetHelper(“Prof Unlucky”, 100).equals(unlucky100)); assertTrue(unlucky100.equals(targetHelper(“Prof Unlucky”, 100)));	True
Different character	**// Different health** targetHelper(“Prof Unlucky”, 101).equals(unlucky100);	False

2.8 hashCode()

Test arguments	**Input**	**Expected**
Same character	**// test new object vs previously created object** targetHelper(“Prof Unlucky”, 100).hashCode == unlucky100.hashCode	True
Different character	**// Slightly Different health** targetHelper(“Prof Unlucky”, 101).hashCode() == unlucky100.hashCode(); **// Slightly different name** targetHelper(“Prof Lucky”, 100).hashCode() == unlucky100.hashCode();	False

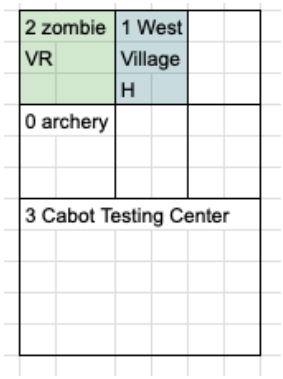
DefaultSpace

Will add the following method for mileston3 to support the requirements of :

- determine if player A can *see* player B. Player A can be see player B if they are in the same space or if player A is in one of the neighboring spaces of space that player B is in.
    - doNeighborsHavePlayers() : bool
    - doPetlessNeighborsHavePlayers() : bool
    - toStringHidePetRoom() : String

Sample world:

```
in = new StringReader(
        "15 16 Northeastern X Disney Land\n"
        + "2 Prof Jump\n"
        + "Fortune the cat\n"
        + " 4\n"
        + " 3  1  5  2 Archery Range\n"
        + " 0  3  2  4 West Village H\n"
        + " 0  0  2  2 Zombie VR\n"
        + " 6  1 10  8 Cabot Testing Center\n"
        + " 3\n"
        + " 0 3 Suction Cup Arrows\n"
        + " 1 10 Deadline Extension\n"
        + " 2 2 Motion sick VR headset\n");
```



0 - Archery Range (3,1 , 5,2)
item: Suction cup arrow ,**damage point=3**
1 - West Village H (0,3 , 2,4)
item: deadline extension, **damage point=10**
2 - Zombie VR (0,0 , 2,2)
item: motion sick VR headset, **damage point=2**
3 - Cabot Testing Center (6,1 , 10, 8)

**3.1 doNeighborsHavePlayers() :bool

Will check for all neighbors players including the one with cat.
This returns the true visibility. This allows to determine if a player can be seen

Test valid arguments	**Input**	**Expected**
Neighbors have no pet	// neighbors have been added with 2 players Space archery = new Space(1, "Archery Range", new RowCol(3, 1), new RowCol(5, 2), items); String expected = zombie.doNeighboursHavePlayers();	→ true
Neighbors have pet	// neighbors have been added with 2 players in room with cat Space archery = new Space(1, "Archery Range", new RowCol(3, 1), new RowCol(5, 2), items); zombie.addPet(cat) String expected = zombie.doNeighboursHavePlayers();	→ true

**3.2 doPetlessNeighborsHavePlayers() :bool

Only check for neighbors without pet. This is needed to check if player can see other players

Test valid arguments	**Input**	**Expected**
Neighbors have no pet	// neighbors have been added with 2 players in room with cat Space zombie = new Space(3, "Zombie VR", new RowCol(3, 1),new RowCol(5, 2), items); String expected = zombie.doNeighboursHavePlayers()	true
neighbors have pet	// neighbors have been added with 2 players in room with cat Space archery = new Space(1, "Archery Range", new RowCol(3, 1), new RowCol(5, 2), items); zombie.addPet(cat) String expected = zombie.doNeighboursHavePlayers();	False // players in room with cat has been hidden

**3.3 toStringHidePetRoom() :bool

Return space info for look around (including neighbors but hide the pet room)

Test valid arguments	**Input**	**Expected**
neighbors have pet	// current space (archery) Space archery = new Space(1, "Archery Range", new RowCol(3, 1), new RowCol(5, 2), items);	"3 Space name: Zombie VR\n" + "======Items======:\n" + "-Item name: Motion Sick VR Headset\n" + "Item Index: 2\n" + "Damage Point: 2\n"

	westVillage.addPet(cat); String expected = archery.toStringHidePetRoom();	+ "\n" + "======Neighbors==:\n" + "% 1 West Village H %\n" + "neighbor's item:\n" + "-Item name: Deadline Extension\n" + "Item Index: 1\n" + "Damage Point: 10\n" + "neighbor's players:\n" + "\n" + "\n" + "======Players==:\n"
Neighbors have no pet	// current space (archery) Space archery = new Space(1, "Archery Range", new RowCol(3, 1), new RowCol(5, 2), items); String expected = archery.toStringHidePetRoom();	"3 Space name: Zombie VR\n" + "======Items======:\n" + "-Item name: Motion Sick VR Headset\n" + "Item Index: 2\n" + "Damage Point: 2\n" + "\n" + "======Neighbors==:\n" + "\n" + "% 0 Archery Range %\n" + "neighbor's item:\n" + "-Item name: Suction Cup Arrows\n" + "Item Index: 0\n" + "Damage Point: 3\n" + "neighbor's players:\n" + "\n" + "% 1 West Village H %\n" + "neighbor's item:\n" + "-Item name: Deadline Extension\n" + "Item Index: 1\n" + "Damage Point: 10\n" + "neighbor's players:\n" + "\n" + "\n" + "======Players==:\n"

**3.4 addPet() :bool

Test valid arguments	**Input**	**Expected**
Add valid	// current space (archery) Space archery = new Space(1, "Archery Range", new RowCol(3, 1), new RowCol(5, 2), items);	"3 Space name: Zombie VR\n" + "======Items======:\n" + "-Item name: Motion Sick VR Headset\n" + "Item Index: 2\n"

	archery.addPet(cat);	+ "Damage Point: 2\n" + "\n" + "======Neighbors==:\n" + "% 1 West Village H %\n" + "neighbor's item:\n" + "-Item name: Deadline Extension\n" + "Item Index: 1\n" + "Damage Point: 10\n" + "neighbor's players:\n" + "\n" + "\n" + "======Players==:\n" + Fortune the cat is here
Add null	// current space (archery) Space archery = new Space(1, "Archery Range", new RowCol(3, 1), new RowCol(5, 2), items); archery.addPet(cat);	IllegalArgumentException

3.5 Constructor

3.5 toString()

@Before
// make list of items
Set<Item> items = new HashSet<Item> ();
Item knife = Item(“Sharp Knife”, 1, 12)
Item ball = Item(“Slippery Ball”, 1, 5)
Item painting= Item(“Ugly Painting”, 1, 3)
Items.add(knife);
Items.add(ball);
Items.add(paining);

// constructor
Space(int spaceIndex, String name, RowCol upLeft, RowCol lowRight, Set<Item> items)

3.4 and 3.5 Test `constructor` and `toString()` valid arguments

Test valid arguments	**Input**	**Expected**

Long name ()	spaceHelper(1, “Behrakis Health and Science Center”, new RowCol(10,25), new RowCol(12,30), items)	“1 Space name: Behrakis Health and Science Center Neighbor: Items: -Item name: Sharp Knife Room Index: 1 Damage Point: 12 -Item name: Slippery Ball Room Index: 1 Damage Point: 5 -Item name: Ugly Painting Room Index: 1 Damage Point: 3 Players: ”
Empty items	spaceHelper(1, “Behrakis Health and Science Center”, new RowCol(10,25), new RowCol(12,30), new HashSet<Item> ())	“1 Space name: Behrakis Health and Science Center Neighbor: Items:
Small room	spaceHelper(1, “Behrakis Health and Science Center”, new RowCol(0,2), new RowCol(1,3), items)	“1 Space name: Behrakis Health and Science Center Neighbor: Items: -Item name: Sharp Knife Room Index: 1 Damage Point: 12 -Item name: Slippery Ball Room Index: 1 Damage Point: 5” -Item name: Ugly Painting Room Index: 1 Damage Point: 3 Players: ”

3.1 and 3.2 Test `constructor` and `toString()` invalid arguments

Test invalid arguments	**Input**	**Expected**

Empty space name	spaceHelper(1,"", new RowCol(10, 25), new RowCol(12, 30), items)	IllegalArgumentException
Negative space index	spaceHelper(-1, "Behrakis Health and Science Center", new RowCol(10,25), new RowCol(12, 30), items)	IllegalArgumentException
upLeft row < lowRight row	spaceHelper(1, "Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(9, 30), items);	IllegalArgumentException
upLeft col < lowRight col	spaceHelper(1, "Behrakis Health and Science Center" new RowCol(10, 25), new RowCol(12, 24), items)	IllegalArgumentException
upeLeftRow = lowRightRow	spaceHelper(1, "Behrakis Health and Science Center" new RowCol(10, 25), new RowCol(10, 25), items)	IllegalArgumentException
Room and item index does not match	Item ball = Item("Slippery Ball", 0, 5) items.add(item) spaceHelper(1, "Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(12, 24), items)	IllegalArgumentException
Null rowCol	RowCol upLeft; RowCol lowRight; spaceHelper(1, "Behrakis Health and Science Center", newRowCol(10, 25) new RowCol(12, 24), items)	NullPtrException
Null items	items = null; spaceHelper(1, "Behrakis Health and Science Center", newRowCol(10, 25), new RowCol(12, 30), items)	NullPtrException
Null item	Item nullitem = null; items.add(nullitem); spaceHelper(1, "Behrakis Health and Science Center", newRowCol(10, 25), new RowCol(12, 30), items)	NullPtrException

3.6 getName()

Test arguments	**Input**	**Expected**
long Name	spaceHelper(1,"Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(12, 30), items)	“Behrakis Health and Science Center”

Short Name	spaceHelper(1, "B, new RowCol(10, 25), new RowCol(12, 30), items)	“B”

3.7 getUpLeft()

Test arguments	**Input**	**Expected**
0 X 0	spaceHelper(1,"Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(12, 30), items).getUpLeft().toString()	“0 X 0”
large	spaceHelper(1, "Behrakis, new RowCol(1231, 231), new RowCol(12, 30), items).getUpLeft().toString()	“1231 X 231”

3.8 getLowRight()

// same as above but 1 X 1 instead of 0 X 0

3.9 equals()

Spaces with same names are equal

Test arguments	**Input**	**Expected**
Same every fields	**// test new object vs previously created object and symmetry** assertTrue(behrakis).equals(new Space(1, "Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(12, 30), items));	True
Different items	assertTrue(behrakis).equals(new Space(1, "Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(12, 30), emptyItems)); // add symmetry	True

3.10 hashCode()

Test arguments	**Input**	**Expected**

Same every fields	**// test new object vs previously created object and symmetry** behrakis.hashCode() ==(new Space(1, "Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(12, 30), items).hashCode()	True
Different items	behrakis.hashCode() == (new Space(1, "Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(12, 30), emptyItems).hashCode(); // add symmetry	True

3.8 playersAsString()

Test arguments	**Input**	**Expected**
Same every fields	**// test new object vs previously created object and symmetry** behrakis.hashCode() ==(new Space(1, "Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(12, 30), items).hashCode()	True
Different items	behrakis.hashCode() == (new Space(1, "Behrakis Health and Science Center", new RowCol(10, 25), new RowCol(12, 30), emptyItems).hashCode(); // add symmetry	True

3. DefaultPet

- The pet enters the game in the same space as the target character.
  It should be included in the description of the space that it occupies.
- Any space that is occupied by the pet *cannot be seen* by its neighbors making it virtually invisible to the user.
- Players actions should include moving the pet to a specified space. *This represents a turn.*

- the user has some way of knowing where the target player is in the world

Own requirements: String cannot be null or empty

4.1 Constructor and to String

Test valid arguments	**Input**	**Expected**
Valid cat	Pet cat = new Pet(“Fortune the cat”, 0); cat.toString()	“Fortune the cat is in room 0”
Empty String	Pet cat = new Pet(“”, 0); cat.toString()	Illegal Argument Exception
Null name	Pet cat = new Pet(null, 0); cat.toString()	Illegal Argument Exception
Negative room index	Pet cat = new Pet(null, -1); cat.toString()	Illegal Argument Exception

4.2 getCurrentSpaceIndex() and moveSpace(int)

Test valid arguments	**Input**	**Expected**
Move Space	Pet cat = new Pet(“Fortune the cat”, 2); String expected1 = cat.getCurrentSpaceIndex() cat.moveSpace(3) String expected = cat.getCurrentSpaceIndex()	→ 2 → 3
Move to invalid space (space does not exists)	Pet cat = new Pet(“Fortune the cat”, 2); String expected1 = cat.getCurrentSpaceIndex() cat.moveSpace(200) String expected = cat.getCurrentSpaceIndex()	Illegal Argument Exception
Move to invalid space (negative index)	Pet cat = new Pet(“Fortune the cat”, 2); String expected1 = cat.getCurrentSpaceIndex() cat.moveSpace(-1) String expected = cat.getCurrentSpaceIndex()	Illegal Argument Exception
Move multiple times	Pet cat = new Pet(“Fortune the cat”, 2); String expected1 = cat.getCurrentSpaceIndex() cat.moveSpace(3) String expected = cat.getCurrentSpaceIndex() cat.moveSpace(1) String expected = cat.getCurrentSpaceIndex() cat.moveSpace(0) String expected = cat.getCurrentSpaceIndex()	→ 2 → 3 → 1 → 0

4. DefaultItem

WIll add a getter getDamagePoint() to satisfy milestone 3 requirement in order to reduce target's health with the correct damage.

4.0 getDamagePoint()

Test case	**Input**	**Expected**
valid	Item knife = Item("Pink knife", 4, 32) item.getDamagePoint()	→ 32
Small damage point	Item knife = Item("Pink knife", 4, 1) item.getDamagePoint()	→ 1

4.1 Constructor

Requirements:

- Two items cannot have the same name
- Item name's length should be between 2-50
- names should contain A-Z, a-z, 0-9, ' only
- roomIndex cannot be greater than world's total space minus 1
- Damage point cannot be greater than 99
- Damage point cannot be less than 1

4.2 toString()

Requirements:

Will return string the summarizes the item as follows:

"Item name: sharp knife
Room Index: 4
Damage Point: 32"

4.1 and 4.2 Test `constructor` **and** `toString()`

Test valid arguments	**Input**	**Expected**
Long item name (50 characters	Item(“Pink Bad Tight Silken Red Billiard Sharp Rat Knife”, 4, 32)	“Space Name: Tin Room Item: Pink Bad Tight Silken Red Billiard Sharp Rat Knife Room Index: 4 Damage Point: 32”
Short item name (2 characters)	Item(“Ax”, 4, 40)	“Item name: Ax Room Index: 4 Damage Point: 32”
roomIndex border point (0)	Item(“Sharp Knife”, 0, 30)	“Space Name: Tin Room Item: -”
Damage point low border (1)	Item(“Sharp Knife”, 10, 1)	“Item name: Sharp Knife Room Index: 4 Damage Point: 0”

Test invalid arguments	**Input**	**Expected**
Empty item name	itemHelper("", 0, 21);	IllegalArgumentException
Negative room index	itemHelper("", -1, 21);	IllegalArgumentException
Damage point 0	itemHelper("", 13, 0);	IllegalArgumentException
Damage point negative	itemHelper("", 14, -1);	IllegalArgumentException

4.3 hashCode() and equals()

Requirements:

- Two items are the same if they have the same name, damage point, and room index
- We will have helper method itemHelper() that will create Item object

Test arguments	**Input**	**Expected**

Same item	broom021 = itemHelper("broom", 0, 21); stinkyTofu2037 = itemHelper("stinky tofu", 20, 37); **equals()** itemHelper("broom", 0, 21).equals(broom021) broom021.equals(itemHelper("broom", 0, 21))	True True
	hashCode() assertTrue(itemHelper("stinky tofu", 20, 37).hashCode() == stinkyTofu2037.hashCode()); assertTrue(itemHelper("broom", 0, 21)).hashCode() == broom021.hashCode());	True True
Different Item (different room index and different damage point	**equals()** 2. test different room index assertFalse((itemHelper("broom", 1, 21).equals(broom021))); 3. test different damage point assertFalse((itemHelper("broom", 0, 20).equals(broom021))); **hashCode()** 2. test different room index assertFalse((itemHelper("broom", 1, 21).hashCode() == broom021.hashCode())); assertFalse((itemHelper("stinky tofu", 19, 37).hashCode() == stinkyTofu2037.hashCode())); 2. test different damage point assertFalse((itemHelper("broom", 0, 22).hashCode() == broom021.hashCode())); assertFalse((itemHelper("stinky tofu", 20, 36).hashCode() == stinkyTofu2037.hashCode()));	False False False False

4.4 getRoomIndex()

Test arguments	**Input**	**Expected**
Big index	itemHelper("Stinky Tofu", 1231, 1).getRoomIndex()	1231

Small index	itemHelper("Stinky Tofu", 0, 1).getRoomIndex() itemHelper("Stinky Tofu", 1, 1).getRoomIndex()	0 1

5. **RowCol**
   - **5.1 Constructor**
     Requirements: Row and Col cannot be negative
   - **5.2 getRow()**
   - **5.3 getCol()**
   - **5.4 toString()**

Test arguments	**Input**	**Expected**
Valid	rowCol1025 = rowColHelper(10, 25); rowCol1025.getRow() rowCol1025.getCol() rowCol1025.toString()	 10 25 “10 X 25”
Zero Row and Col	rowColHelper(0, 0).getRow(); rowColHelper(0, 0).getCol(); rowColHelper(0,0).toString();	0 0 “0 X 0”
Negative Row and col	rowColHelper(-1, 25).getRow(); rowColHelper(10, -1.getCol(); rowColHelper(-25, -25).getRow();	IllegalArgumentException IllegalArgumentException IllegalArgumentException

5.5 equals()

Test arguments	**Input**	**Expected**

Same RowCol	**// test new object vs previously created object** rowColHelper(10, 25).equals(rowCol1025) rowCol1025.equals(rowColHelper(10,25))	True
Different RowCol	**// Different row** rowColHelper(11, 25).equals(rowCol1025); **// Different col** rowColHelper(10, 24).equals(rowCol1025);	False False
rowCol from space VS rowCol from newly created	Item knife = new Item("Sharp Knife", 1, 12); Item ball = new Item("Slippery Ball", 1, 5); Set<Item> items = new HashSet<Item>(); Set<Item> emptyItems = new HashSet<Item>(); items.add(knife); items.add(ball); Space space = new Space(1, "Behrakisr", new RowCol(10, 25), new RowCol(12, 30), items); Space space2 = new Space(1, "Behr", new RowCol(10, 25), new RowCol(12, 30), emptyItems); // different object same value assertTrue(space.getUpLeft().equals(space2.getUpLeft())); assertTrue(space.getLowRight().equals(space2.getLowRight())); // newly created vs get from space RowCol spaceUpLeftRowCol = new RowCol(10, 25); assertTrue(spaceUpLeftRowCol.equals(space2.getUpLeft()));	True

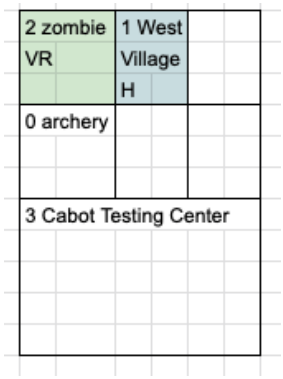
5.6 hashCode()

Test arguments	**Input**	**Expected**
Same RowCol	**// test new object vs previously created object** rowColHelper(10, 25).hashCode() == rowCol1025 .hashCode()	True
Different RowCol	**// Different Row** rowColHelper(9, 25).hashCode() == rowCol1025 .hashCode() **// Different col** rowColHelper(10, 24).hashCode() == rowCol1025.hashCode()	False False

6. DefaultPlayer

The below methods highlighted in yellow will be added to support the requirements:

- determine if player A can *see* player B. Player A can be see player B if they are in the same space or if player A is in one of the neighboring spaces of space that player B is in.
- Players should be able to make an attempt on the target character's life if they are in the same space as the target character. *This represents a turn.*
    - Computer-controlled players always choose to make an attempt on the target character's life (if they cannot be *seen* by others) using the item in their inventory that does the most damage.
    - A player who does not have any items, can always make an attempt on the target character's life by "poking him in the eye" which does 1 point of damage



0 - Archery Range
item1: Suction cup arrow ,**damage point=3**
1 - West Village H
item2: deadline extension, **damage point=10**
2 - Zombie VR
item3: motion sick VR headset, **damage point=2**
3 - Cabot Testing Center

** 6.1 getItemIndexWithMostDamage() :int

Test arguments	**Input**	**Expected**

Item exists	// pick item by index player1.pickUpItem(1); player1.moveToNeighbor(1); player1.pickUpItem(2); player1.moveToNeighbor(2); player1.pickUpItem(3); Item item = player1.getItemIndexWithMostDamage() String expected = item.toString();	2
Player has no item	Item item = player1.getItemIndexWithMostDamage()	0
Only 1 item	// pick item by index player1.pickUpItem(1);	1

**6.2 attemptToKill(Target, int) : boolean

If item index is not carried by the player, it will poke eye instead

Test arguments	**Input**	**Expected**
Item exists	// target health is 50 // will attempt to kill with weapon index 2 - damage point 10 Boolean success = player1.attemptToKill(drUnlucky, 2); String items left = player1.itemsAsString(); Strin target health = drUnlucky.getHealth();	→ true → player1 is carrying no Item → drUnlucky's health is now 40
Item does not exists - will poke eye instead	// target health is 50 // will attempt to kill with weapon index 20 Boolean success = player1.attemptToKill(drUnlucky, 20); String items left = player1.itemsAsString(); String target health = drUnlucky.getHealth();	→ true → player1 is carrying no Item → drUnlucky's health is now 49
Target is null	// target health is 50 // will attempt to kill with weapon index 20 Boolean success = player1.attemptToKill(null, 20);	IllegalArgumentException

Use 0 for poke eye	// target health is 50 // will attempt to kill with weapon index 20 Boolean success = player1.attemptToKill(null, 0); String items left = player1.itemsAsString(); String target health = drUnlucky.getHealth();	→ true → player1 is carrying no Item → drUnlucky's health is now 49

** 6.3 movePet(Pet, int) : boolean

Move pet to space index stated in parameter

Test arguments	**Input**	**Expected**
Move successful	player1.movePet(lucky, 0, 4)	→ true
Move failed, invalid destination room number	player1.movePet(lucky, 0, 100)	false
Invalid pet	player1.movePet(null, 0, 100)	IllegalArgumentException

**6.4 canSeeOthers() : boolean

Will return true if rooms with no cats have players. Return false is room with no cat have no players

Test arguments	**Input**	**Expected**
No cat - with players	// neighbor Zombie has 2 players archery.addPlayer(player1); player1.canSeeOthers();	→ true
No cat - no players	// neighbor Zombie has no players archery.addPlayer(player1); player1.canSeeOthers();	→ false
With cat - no players	// neighbor Zombie has no players and a cat archery.addPlayer(player1); player1.canSeeOthers();	→ false

With cat - with 2players	// neighbor Zombie has 1 player with cat archery.addPlayer(player1); player1.canSeeOthers();	→ false
With cat - with 2 players - 1 in other room	// neighbor Zombie has 1 player with cat // neighbor Calbot has 1 player with no cat archery.addPlayer(player1); player1.canSeeOthers();	→ true

** 6.5 itemsAsString() : String

Test arguments	**Input**	**Expected**
0 item	player1.itemsToString()	“Player 1 has no item”
2 items	player1.pickUpItem(1); player1.pickUpItem(2); player1.itemsToString()	“Player 1 has: Item name: Arrow Item index: 1 Damage Point: 2 Item name: Motion Sick VR Headset Item index: 2 Damage Point: 4”
3 items	player1.pickUpItem(1); player1.pickUpItem(2); player1.pickUpItem(2); player1.itemsToString()	“Player 1 has: Item name: Arrow Item index: 1 Damage Point: 2 Item name: Motion Sick VR Headset Item index: 2 Damage Point: 4” Item name: Deadline Extension Item index: 3 Damage Point: 10”

6.7 Constructor and toString

Test using helper method

Test arguments	**Input**	**Expected**

Valid - short name	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.toString()	“Player name: Sherly Current space: Living Room Item held:
Valid - long name	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.toString()	“Player name: Sherly Hartono 123XX0K Current space: Living Room Item held:
Invalid - empty string name	Player sherly = HumanPlayerHelper(“”, space1) sherly.toString()	IllegalArgumentException
Space is null	Player sherly = HumanPlayerHelper(“Sherly”, null) sherly.toString()	IllegalArgumentException

6.8 MoveToNeighbour(Space) : String

will return the current player info string

Test arguments	**Input**	**Expected**
Valid - move 1 time	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.moveToNeighbour(spaceBilliard) sherly.toString()	“Player name: Sherly Hartono 123XX0K Current space: Billiard Room Item held: “
Valid - move two times	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.moveToNeighbour(spaceBilliard) sherly.moveToNeighbour(spaceKitchen) sherly.toString()	“Player name: Sherly Hartono 123XX0K Current space: Kitchen Item held: “
Valid - move to current place	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.moveToNeighbour(space1) sherly.toString()	“Player name: Sherly Hartono 123XX0K Current space: Kitchen Item held: “

Invalid - space given is not a neighbor of current space	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.moveToNeighbour(notNeighbour)	IllegalArgumentException - space you want to move is not a neighbor Or does not exists
Invalid - space is null	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.moveToNeighbour(null)	IllegalArgumentException

6.9 pickUpItem(Item) : String

Test arguments	**Input**	**Expected**
Valid - pick up 3 items	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.pickUpItem(knife) sherly.pickUpItem(rope) sherly.pickUpItem(gum) // also check currentSpace items Items should be missing one at a time	“Player name: Sherly Current space: Kitchen Item held: Knife Gum“ “1 Space name: Behrakis Health and Science Center Neighbor: Items: Knife Gum Rope” “1 Space name: Behrakis Health and Science Center Neighbor: Items: Gum Rope” “1 Space name: Behrakis Health and Science Center Neighbor: Items: Gum” “1 Space name: Behrakis Health and Science Center Neighbor:

		Items: ”
Invalid - item is null	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.pickUpItem(null)	IllegalArgumentException

6.10 LookAround() : String

Test arguments	**Input**	**Expected**
Valid 1 neighbors	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.lookAround()	3. Space name: Behrakis Health and Science Center Items: -Item name: Sharp Knife Damage Point: 12 -Item name: Slippery Ball Damage Point: 5 -Item name: Ugly Painting Damage Point: 3 Players: Maria”
Valid 2 neighbors	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.lookAround()	“1. Space name: Behrakis Health and Science Center Items: -Item name: Sharp Knife Damage Point: 12 -Item name: Slippery Ball Damage Point: 5 -Item name: Ugly Painting Damage Point: 3 Players: Maria _____ 0 Space name: Archery Range Neighbors: -Zombie VR Items: -Item name: Suction Cup Arrows

		Room Index: 0 Damage Point: 3 Players: Sherly”
Valid 3 neighbors	Player sherly = HumanPlayerHelper(“Sherly”, space1) sherly.lookAround()	“1. Space name: Behrakis Health and Science Center Items: -Item name: Sharp Knife Damage Point: 12 -Item name: Slippery Ball Players: Maria _____ 0 Space name: Archery Range Neighbors: -Zombie VR Items: -Item name: Suction Cup Arrows Room Index: 0 Damage Point: 3 Players: Sherly” _____ 0 Space name: Dining Hall Neighbors: -Zombie VR Items: -Item name: fork Room Index: 0 Damage Point: 3 Players: Bruce”

6.10 equals() : boolean

sherlyKitchen = HumanPlayerHelper(“Sherly”, kitchen)

Test arguments	Input	Expected
Same player	**// test new object vs previously created object** assertTrue(HumanPlayerHelper(“Sherly”, kitchen).equals(sherlyKitchen)); assertTrue(sherlyKitchen.equals(targetHelper(“Sherly”, sherlyKitchen)));	True
Different character	**// Different space** HumanPlayerHelper(“Sherly”, room101).equals(sherlyKitchen);	False

6.11 hashCode(): int

Test arguments	Input	Expected
Same character	**// test new object vs previously created object** HumanPlayerHelper(“Sherly”, room101).hashCode == sherlyKitchen.hashCode	True
Different Player	**// Slightly Different health** HumanPlayerHelper(“Sherly”, room101).hashCode() == sherly.hashCode(); **// Slightly different name** HumanPlayertHelper(“Sherl”, room101).hashCode() == sherly.hashCode();	False

7. DefaultCpu

Computer-controlled players always choose to make an attempt on the target character's life (if they cannot be *seen* by others) using the item in their inventory that does the most damage.

This is almost identical cases as the one in the world. But it will also be unit tested here. The arguments and expected will be the same as string returned in DefaultWorld Class

Test arguments	Input	Expected

Successful attempt- poke eye	// make CPU go to space 0 Player cpu1 = CpuPlayerHelper() // create a space Space archery = new Space(1, "Archery Range", new RowCol(3, 1), new RowCol(5, 2), items); zombie.addPlayer(cpu1) // Create world and make target in space 0 // make murder attempt String = cpu1.takeCpuTurn()	“CPU_xyz has successfully attempted a murder by poking target’s eyes, target's health is now 49. CPU_xyz now hold these items:\n -”
Check get item with most damage	// make CPU go to space 0 Player cpu1 = CpuPlayerHelper() // create a space Space archery = new Space(1, "Archery Range", new RowCol(3, 1), new RowCol(5, 2), items); zombie.addPlayer(cpu1) cpu1.takeCpuTurn(); // make pick item cpu1.takeCpuTurn(); // make pick item cpu1.takeCpuTurn(); // make pick item // Create world and make target in space 0 // make murder attempt String = cpu1.takeCpuTurn()	“Attempt successful poked target in the eye. Target’s health is now 49”
Successful attempt- poke eye	// make CPU in same space as target worldDisnu.addCpuPlayerToList(); // target is at 0, player in 0; String expected = cpu1. takeCpuTurn();	“CPU_xyz has successfully attempted a murder by poking target’s eyes, target's health is now 49. CPU_xyz now hold these items:\n -”
Failed attempt seen by human with pet	worldDisnu.addHumanPlayerToList(2); // CPU in room 1 worldDisnu.addCpuPlayerToList(); worldDisnu.makeHumanMovePet(2); // target is at 1, cpu player in 1, human player in 2 with pet; String expected = cpu1. takeCpuTurn();	“CPU_xyz failed to attempt murder with eye poke. Target health is still 50. CPU_xyz now hold these items:\n -”
CPU does not attempt murder if it’s not in same space as target	worldDisnu.addCpuPlayerToList(); // target is at 0, player in 2; String expected = cpu1. takeCpuTurn();;	“CPU up picked deadline extension”

CPU move pet	worldDisnu.addCpuPlyaerToList(2); String expected = cpu1. takeCpuTurn();	"CPU have successfully moved pet to room 1"
Attempt murder using max weapon	worldDisnu.addCpuPlayerToList(); // add to room 3 // make CPU picked up 3 items usinr random inte generator cpu1. takeCpuTurn(); // target in room 0 cpu1. takeCpuTurn();// target in room 1 cpu1. takeCpuTurn(); // target in room 2 String expected = cpu1. takeCpuTurn();// target in room 3	"CPU have successfully attempted murder with deadlin extension. Target health is now 40. CPU_xyz now hold these items: - Suction cup arrow - Motion Sick VR headset"
Test CPU Look around	worldDisnu.addCpuPlayerToList(); String expected =cpu1. takeCpuTurn();	"CPU took a look around"

7.1 Constructor and toString

Test using helper method
Current space for the test case is living room

Test arguments	**Input**	**Expected**
Valid- check random ness	Player cpu1 = CpuPlayerHelper() cpu1.toString()	"Player name: Some Random name Current space: Living Room Item held:
	Player cpu2 = CpuPlayerHelper() cpu1.toString()	"Player name:Another random name Current space: Dining Room Item held:
	Player cpu3 = CpuPlayerHelper() cpu1.toString()	"Player name: Johnny Current space: Archery Range Item held:

7.2 MoveToNeighbour(Space) : String

Test arguments	**Input**	**Expected**
Valid - move 1 time	Player cpu1 = CpuPlayerHelper() cpu1.moveToNeighbour(spaceBilliard) cpu1.toString()	“Player name: Some Random name Current space: Billiard Room Item held:
Valid - move two times	Player cpu1 = CpuPlayerHelper() cpu1.moveToNeighbour(spaceBilliard) cpu1.moveToNeighbour(spaceKitchen) cpu.toString()	“Player name: Some Random name Current space: Kitchen Item held:
Valid - move to current place	Player cpu1 = CpuPlayerHelper() cpu1.moveToNeighbour(Living Room) cpu.toString()	“Player name: Some Random name Current space: Living Room Item held:

7.3 pickUpItem(Item) : String

Test arguments	**Input**	**Expected**
Valid - pick up 2 items	Player cpu1 = CpuPlayerHelper() cpu1.pickUpItem() cpu1.pickUpItem() cpu1.pickUpItem() //Check current space’s item space1.toString()	“1 Space name: Behrakis Health and Science Center Neighbor: Items: Knife Gum

	space1.toString() space1.toString()	Rope” “1 Space name: Behrakis Health and Science Center Neighbor: Items: Gum Rope” “1 Space name: Behrakis Health and Science Center Neighbor: Items: Gum” “1 Space name: Behrakis Health and Science Center Neighbor: Items: ”
	// Check final list of items in the cpu	“Player name: Some Random name Current space: Kitchen Item held: rope”

7.4 LookAround() : String

Test arguments	**Input**	**Expected**
Valid 1 neighbors	Player cpu1 = CpuPlayerHelper() cpu1.lookAround()	3. Space name: Behrakis Health and Science Center Items: -Item name: Sharp Knife Damage Point: 12 -Item name: Slippery Ball Damage Point: 5 -Item name: Ugly Painting Damage Point: 3

		Players: Maria”
Valid 2 neighbors	Player cpu1 = CpuPlayerHelper() cpu1.lookAround()	“1. Space name: Behrakis Health and Science Center Items: -Item name: Sharp Knife Damage Point: 12 -Item name: Slippery Ball Damage Point: 5 -Item name: Ugly Painting Damage Point: 3 Players: Maria _____ 0 Space name: Archery Range Neighbors: -Zombie VR Items: -Item name: Suction Cup Arrows Room Index: 0 Damage Point: 3 Players: Sherly”
Valid 3 neighbors	Player cpu1 = CpuPlayerHelper() cpu1.lookAround()	“1. Space name: Behrakis Health and Science Center Items: -Item name: Sharp Knife Damage Point: 12 -Item name: Slippery Ball Players: Maria _____ 0 Space name: Archery Range Neighbors: -Zombie VR Items:

		-Item name: Suction Cup Arrows Room Index: 0 Damage Point: 3 Players: Sherly” _____ 0 Space name: Dining Hall Neighbors: -Zombie VR Items: -Item name: fork Room Index: 0 Damage Point: 3 Players: Bruce”

7.6 equals() : boolean

cpu1 = CpuPlayerHelper(“Random”, kitchen)

Test arguments	**Input**	**Expected**
Same player	**// test new object vs previously created object** assertTrue(CpuPlayerHelper(“Random”, kitchen) .equals(cpu1)); assertTrue(cpu1.equals(CpuPlayerHelper(“Random”, kitchen)));	True
Different character	**// Different space** CpuPlayerHelper(“Random”, kitchen) .equals(cpu1);	False

7.7 hashCode(): int

Test arguments	Input	Expected
Same character	**// test new object vs previously created object** CpuPlayerHelper(“Random”, kitchen).hashCode == cpu1.hashCode	True
Different Player	**// Slightly Different health** CpuPlayerHelper(“Random”, kitchen) .hashCode() == cpu1.hashCode(); **// Slightly different name** CpuPlayerHelper(“Random”, kitchen).hashCode() == cpu1.hashCode();	False

8. ** DefaultWorldController

Will add two commands to support the added requirements so that human player can move pet and human player can attempt a murder. This is tested as a part of controller.

8.0 HumanMovePet() and HumanKillAttempt()

User will enter ‘11’ to move a pet and enter ‘12’ to attempt a kill. Followed by respective space or item index.

This will be tested using Mock class that has these methods:

```
 @Override
 public String makeHumanMovePet(int spaceIndex) throws IllegalArgumentException {
        inlog.append("space index is: ").append(spaceIndex);
        return outString;
 }
```

```
@Override
public String makeHumanKillAttempt(int itemIndex) throws IllegalArgumentException {
        inlog.append("item index is: ").append(itemIndex);
        return outString;
}
```

Test arguments	**Input**	**Expected**
Test make human move pet	in = new StringReader("11 3\n"); contr = new WorldControllerImpl(in, out); contr.playGame(mockModel);	In expected : " item index 3" Out expected: "Game starts. Enter your option followed by argument: \n" + "----> 11. Moving Pet Result: <----\n" + "1234567\n" + "\n" + "\n" + "---- Game Status ---- \n" + "1234567\n" + "\n" + "Enter your option followed by argument: quitting program...";
Test make human move pet	in = new StringReader("12 1\n"); contr = new WorldControllerImpl(in, out); contr.playGame(mockModel);	In expected : "space index 1" Out expected: + "Game starts. Enter your option followed by argument: \n" + "----> 12. Kill Attempt: <----\n" + "1234567\n" + "\n" + "\n" + "---- Game Status ---- \n" + "1234567\n" + "\n" + "Enter your option followed by argument: quitting program...";

8.1 Constructor

```
private String worldInfo = "15 16 Northeastern X Disney Land\n"
        + "50 Prof Jump";
        " 3  1  5  2 Archery Range\n"
        + " 0  2  2  4 West Village H\n"
        + " 0  0  2  2 Zombie VR"
        "0 3 Suction Cup Arrows\n"
        + "1 10 Deadline Extension\n"
        + "2 2 Motion sick VR headset\n";
```

Test arguments	**Input**	**Expected**

Test invalid World model	StringBuffer out = new StringBuffer(); Reader in = new StringReader(); Controller c = new WorldController(in, out); World m = null c.playGame(m)	IllegalArgumentEx
Invalid commands	StringBuffer out = new StringBuffer(); Reader in = new StringReader(“dis”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m) out.toString()	“Command does not exists”
Invalid commands - empty	StringBuffer out = new StringBuffer(); Reader in = new StringReader(“”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m) out.toString()	“Command does not exists”
Mock test	StringBuffer out = new StringBuffer(); Reader in = new StringReader("display space info\nArchery Range"); StringBuilder log = new StringBuilder(); World m = new MockModel(log, 1234321); WorldController controller = new WorldController(in, out); controller.playGame(m); assertEquals("display space info: archery range", log.toString()); // input reaches model correctly assertEquals("1234321\n1234321\n1234321\n", out.toString()); // output from model received correctly	

8.2 Display space info

To display a space type “disp space” followed by the space name

Test arguments	**Input**	**Expected**

Display Space Info 1	StringBuffer out = new StringBuffer(); Reader in = new StringReader(“disp space\nZombie VR”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m) out.toString()	“1 Space name: Zombie VR Neighbor: Items: -Item name: Sharp Knife Room Index: 1 Damage Point: 12 -Item name: Slippery Ball Room Index: 1 Damage Point: 5 -Item name: Ugly Painting Room Index: 1 Damage Point: 3 Players: ”
Display Space Info - invalid space	StringBuffer out = new StringBuffer(); Reader in = new StringReader((“disp space\nNo space); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m) out.toString()	Space does not exists

8.3 Add cpu player

To add a cpu just type in ““cpu”.
When added will return a string of the player info

Test arguments	**Input**	**Expected**
Add a single cpu player	StringBuffer out = new StringBuffer(); Reader in = new StringReader(“cpu\n”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	“Player name: Some Random name Current space: Living Room Item held:

Add a multiple cpu player	StringBuffer out = new StringBuffer(); Reader in = new StringReader(“cpu\ncpu\n”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	“Player name: Some Random name Current space: Living Room Item held: “Player name: Another Random name Current space: Random Room Item held:
Add with invalid cpu command	StringBuffer out = new StringBuffer(); Reader in = new StringReader((“cp\n”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	“Invalid command”

8.4 Add human player

To add a human play just type in “man”. Then type in the name and space you want to be
When added will return a string of the player info

Test arguments	**Input**	**Expected**
Add a single human player	StringBuffer out = new StringBuffer(); Reader in = new StringReader(“man\nSherly\nLiving Room”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	“Player name: Sherly Current space: Living Room Item held:
Add a multiple human player	StringBuffer out = new StringBuffer(); Reader in = new StringReader(“Man\n Sherly\n Living Room\n Man\n Maria\n Dining Room”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	“Player name: Sherly Current space: Living Room Item held: “Player name: Maria Current space: Dining Room Item held:

Add with invalid space name	StringBuffer out = new StringBuffer(); Reader in = new StringReader((“man\nSherly\nNonexistant”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	“Space name does not exists”

8.5 Move a player and display player info

The player that moves are generated by the order that they are added
To move a player just type “move” followed by the “space name” if it’s human
To display a player type “disp player” followed by the player name

Test arguments	**Input**	**Expected**
Move a player	StringBuffer out = new StringBuffer(); Reader in = new StringReader(“display target\nmove\nKitchen\ndisp\nSherly\ndisplay target”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	target is at room: 0 Archery Range Player name: Sherly Current space: Kitchen Item held: target is at room: 1 West Village H
Move a cpu player	Will be generated inside the playGame logic Will check in the list if the next player is CPU, then check CPU play before and after moving	target is at room: 0 Archery Range // before “Player name: Some random Cpu Current space: Kitchen Item held: Knife” // after Player name: Some random Cpu Current space: Zombie VR Item held: Knife” target is at room: 1 West Village H

Move a human player to invalid space	StringBuffer out = new StringBuffer(); Reader in = new StringReader((“display target\nmove\nNon existing space\ndisplay target”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	target is at room: 0 Archery Range “Space does not exists or is not a neighbor” target is at room: 1 West Village H

8.6 Player pick up item

For a player to pick up an item, type “pick” followed by the item name. To display a player type “disp player” followed by the player name

Test arguments	**Input**	**Expected**
Human player pick up an item	StringBuffer out = new StringBuffer(); Reader in = new StringReader((“display target\pick\nBaseball bat\ndisplay target”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	target is at room: 0 Archery Range “Player name: Sherly Current space: Living Room Item held: Knife Baseball bat” target is at room: 1 West Village H
Human pick up invalid item	StringBuffer out = new StringBuffer(); Reader in = new StringReader((“display target\npick\nSome non existing weapon\ndisplay target”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	Item does not exists

8.7 Player look around

For a player to pick up an item, type “look”.

Test arguments	Input	Expected
player look around	StringBuffer out = new StringBuffer(); Reader in = new StringReader((“display target\nlook\ndisplay target”); Controller c = new WorldController(in, out); World m = new WorldImpl(worldInfo) c.playGame(m)	target is at room: 8 Archery Range “1. Space name: Behrakis Health and Science Center Items: -Item name: Sharp Knife Damage Point: 12 -Item name: Slippery Ball Damage Point: 5 -Item name: Ugly Painting Damage Point: 3 Players: Maria _____ 1. Space name: Zombie VR Items: -Item name: broken VR google Damage Point: 3 Players: Some Random CPU” target is at room: 0 West Village H

8.8 target character move

For every move above we’ve checked that target character moves to a new room

9. DefaultWorldCommand

9.1 Display space info

To display a space type “disp space” followed by the space name

Test arguments	**Input**	**Expected**
Display Space Info 1	World m = new WorldImpl(worldInfo) cmd = displaySpace(“Zombie VR”) cmd.go(m)	“1 Space name: Zombie VR Neighbor: Items: -Item name: Sharp Knife Room Index: 1 Damage Point: 12 -Item name: Slippery Ball Room Index: 1 Damage Point: 5 -Item name: Ugly Painting Room Index: 1 Damage Point: 3 Players: ”
Display Space Info - invalid space	World m = new WorldImpl(worldInfo) WorldCommand cmd = displaySpace(“No space”) cmd.go(m)	Space does not exists

9.2 Add cpu player

Test arguments	**Input**	**Expected**
Add a single cpu player	World m = new WorldImpl(worldInfo) WorldCommand cmd = addPlayers(“cpu”) cmd.go(m)	“Player name: Some Random name Current space: Living Room Item held:

Add multiple cpu player	World m = new WorldImpl(worldInfo) WorldCommand cmd = addPlayers(“cpu”) cmd.go(m) cmd = addPlayers(“cpu”) cmd.go(m)	“Player name: Some Random name Current space: Living Room Item held: “Player name: Another Random name Current space: Random Room Item held:
Add with invalid cpu command	World m = new WorldImpl(worldInfo) WorldCommand cmd = addPlayers(“cp”) cmd.go(m)	“Invalid command”

9.3 Add human player

Test arguments	**Input**	**Expected**
Add a single human player	WorldCommand cmd1 = AddPlayers(“Sherly\nLiving Room”) cmd.go(m)	“Player name: Sherly Current space: Living Room Item held:

Add a multiple human player	WorldCommand cmd1 = addPlayers(“Sherly\nLiving Room”) cmd.go(m) cmd = AddPlayers(“Maria\nDining Room”) cmd.go(m)	“Player name: Sherly Current space: Living Room Item held: “Player name: Maria Current space: Dining Room Item held:
Add with invalid space name	WorldCommand cmd1 = AddPlayers(“Sherly\nSome X space”) cmd.go(m)	“Space name does not exists”

9.4 PlayerTakeATurn - Move a player and DisplayPlayer

Test arguments	**Input**	**Expected**
Move a player	WorldCommand cmd = PlayerTakeAturn(0, “move\nKitchen”) cmd.go(m) cmd = DisplayPlayer(“Sherly”) cmd.go(m)	target is at room: 0 Archery Range Player name: Sherly Current space: Kitchen Item held: target is at room: 1 West Village H

Move a cpu player	WorldCommand cmd = PlayerTakeAturn(2, “move\nKitchen”) cmd.go(m) cmd = DisplayPlayer(“Some random Cpu”) cmd.go(m)	target is at room: 0 Archery Range // before “Player name: Some random Cpu Current space: Kitchen Item held: Knife” // after Player name: Some random Cpu Current space: Zombie VR Item held: Knife” target is at room: 1 West Village H
Move a human player to invalid space	WorldCommand cmd = PlayerTakeAturn(0, “move\nSome space X”) cmd.go(m)	“Space does not exists or is not a neighbor”

9.5 PlayerTakeATurn - Player pick up item

For a player to pick up an item, type “pick” followed by the item name. To display a player type “disp player” followed by the player name

Test arguments	**Input**	**Expected**
Human player pick up an item	WorldCommand cmd = PlayerTakeAturn(2, “pick\nBaseball bat”) cmd.go(m) cmd = DisplayPlayer(“Sherly”) cmd.go(m)	“Player name: Sherly Current space: Living Room Item held: Knife Baseball bat”

Human pick up invalid item	WorldCommand cmd = PlayerTakeAturn(2, “pick\nSome non existing weapon”) cmd.go(m)	Item does not exists
CPU player pick up an item	WorldCommand cmd = PlayerTakeAturn(3,, “pick\nBaseball bat”) cmd.go(m) cmd = DisplayPlayer(“some random cpu”) cmd.go(m)	“Player name: Some random cpu Current space: Living Room Item held: Knife Baseball bat”

9.6 PlayerTakeATurn - Player look around

For a player to pick up an item, type “look”.

Test arguments	**Input**	**Expected**

player look around	WorldCommand cmd = PlayerTakeAturn(2, “look”) cmd.go(m)	“1. Space name: Behrakis Health and Science Center Items: -Item name: Sharp Knife Damage Point: 12 -Item name: Slippery Ball Damage Point: 5 -Item name: Ugly Painting Damage Point: 3 Players: Maria _____ 1. Space name: Zombie VR Items: -Item name: broken VR google Damage Point: 3 Players: Some Random CPU”

Last player look around	WorldCommand cmd = PlayerTakeAturn(4, “look”) cmd.go(m)	“1. Space name: Behrakis Health and Science Center Items: -Item name: Sharp Knife Damage Point: 12 -Item name: Slippery Ball Damage Point: 5 -Item name: Ugly Painting Damage Point: 3 Players: Maria _____ 1. Space name: Zombie VR Items: -Item name: broken VR google Damage Point: 3 Players: Some Random CPU”
Invalid player index	WorldCommand cmd = PlayerTakeAturn(100, “look”) cmd.go(m)	Player at index 100 does not exists

Video Prompt

Give an overall explanation of your design with your particular focus on the implementation of the Milestone 3 requirements. You should include any changes you made because of feedback in previous milestones.