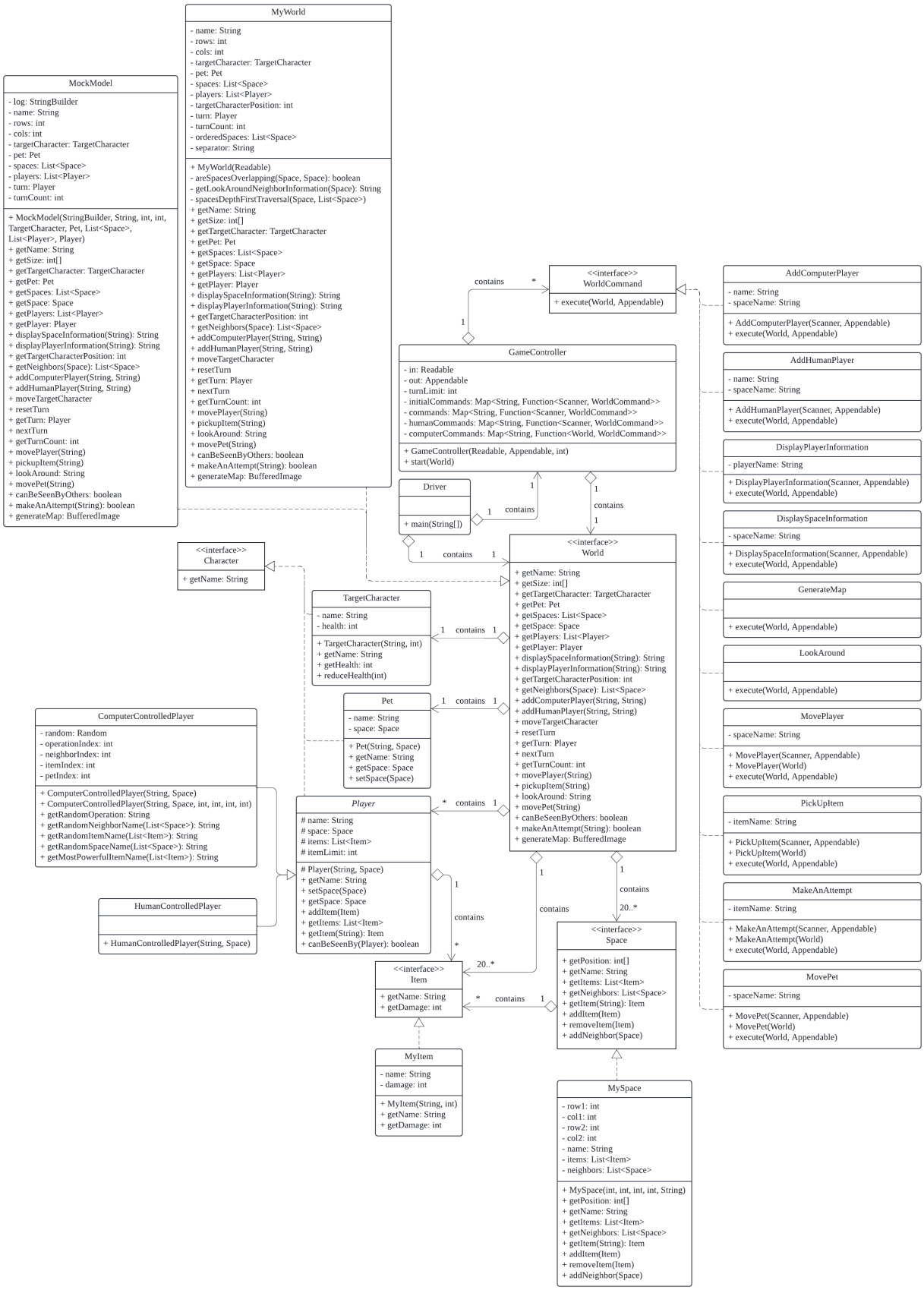
CS5010 Milestone3 Preliminary Design

By Linhao Qian (NUID: 002325915)

UML Diagram



Testing Plan

Testing design for TargetCharacter

Testing construction	Input	Expected Value
Constructor disallows null name	TargetCharacter(null, 50)	IllegalArgumentException
Constructor disallows empty name	TargetCharacter("", 50)	IllegalArgumentException
Constructor disallows non-positive health	TargetCharacter("doctor", 0)	IllegalArgumentException

Testing getName()	Input	Expected Value
TargetCharacter with normal name	TargetCharacter("Leo", 50)	"Leo"

Testing getHealth()	Input	Expected Value
TargetCharacter with positive health	TargetCharacter("Leo", 50)	50

Testing reduceHealth(int damage)	Input	Parameter	Actual Testing
Reduce positive health value	TargetCharacter("Leo", 50)	3	assertEquals(getHealth(), 47)
Test invalid damage	TargetCharacter("Leo", 50)	-3	IllegalArgumentException

Testing design for Pet

Testing construction	Input	Expected Value
Constructor disallows null name	Pet(null, new MySpace(0, 0, 3, 3, "Kitchen"))	IllegalArgumentException
Constructor disallows empty name	Pet("", new MySpace(0, 0, 3, 3, "Kitchen"))	IllegalArgumentException
Constructor disallows null space	Pet("cat", null)	IllegalArgumentException

Testing getName()	Input	Expected Value

Pet with normal name	Pet(“cat”, new MySpace(0, 0, 3, 3, “Kitchen”))	“cat”

Testing getSpace()	**Input**	**Expected Value**
Pet with normal space	Pet(“cat”, new MySpace(0, 0, 3, 3, “Kitchen”))	new MySpace(0, 0, 3, 3, “Kitchen”)

Testing setSpace(Space space)	**Input**	**Parameter**	**Expected Value**
Set a normal space	Pet(“cat”, new MySpace(0, 0, 3, 3, “Kitchen”))	new MySpace(0, 4, 3, 6, “Parlor”)	new MySpace(0, 4, 3, 6, “Parlor”)

Testing design for HumanControlledPlayer

Testing construction	**Input**	**Expected Value**
Constructor disallows null name	HumanControlledPlayer (null, new MySpace(0, 0, 3, 3, “Kitchen”))	IllegalArgumentException
Constructor disallows empty name	HumanControlledPlayer (“”, new MySpace(0, 0, 3, 3, “Kitchen”))	IllegalArgumentException
Constructor disallows null space	HumanControlledPlayer (“Leon”, null)	IllegalArgumentException

Testing getName()	**Input**	**Expected Value**
HumanControlledPlayer with normal name	HumanControlledPlayer (“Leo”, new MySpace(0, 0, 3, 3, “Kitchen”))	“Leo”

Testing addItem(Item item) and getItems()	**Input**	**Expected Value**
Add and get an item	setItem(new MyItem(“Revolver”, 3)); getItems().get(0);	new MyItem(“Revolver”, 3)

Testing setSpace(Space space) and getSpace()	**Operation**	**Expected Value**

Set and get a space	setSpace(new MySpace(0, 0, 3, 3, “Kitchen”)); getSpace()	MySpace(0, 0, 3, 3, “Kitchen”)

Testing design for ComputerControlledPlayer

Testing construction	Input	Expected Value
Constructor disallows null name	ComputerControlledPlayer (null, new MySpace(0, 0, 3, 3, “Kitchen”))	IllegalArgumentException
Constructor disallows empty name	ComputerControlledPlayer (“”, new MySpace(0, 0, 3, 3, “Kitchen”))	IllegalArgumentException
Constructor disallows null space	ComputerControlledPlayer (“Leon”, null)	IllegalArgumentException

Testing getName()	Input	Expected Value
ComputerControlledPlayerwith normal name	ComputerControlledPlayer (“Leo”, new MySpace(0, 0, 3, 3, “Kitchen”))	“Leo”

Testing addItem(Item item) and getItems()	Input	Expected Value
Add and get an item	setItem(new MyItem(“Revolver”, 3)); getItems();	ArrayList<Item>(){ new MyItem(“Revolver”, 3)}

Testing setSpace(Space space) and getSpace()	Operation	Expected Value
Set and get a space	setSpace(new MySpace(0, 0, 3, 3, “Kitchen”)); getSpace()	MySpace(0, 0, 3, 3, “Kitchen”)

Testing getRandomOperation()	Input	Expected Value
Get a random operation	ComputerControlledPlayer (“Leo”, new MySpace(0, 0, 3, 3, “Kitchen”), 0, 0, 0)	''automaticMovePlayer''

Testing getRandomNeighborName()	Input	Expected Value
Get a random neighbor name	Space space = new MySpace(0, 0, 3, 3, “Kitchen”) ComputerControlledPlayer (“Leo”, space, 0, 0, 0) space.addNeighbor(new MySpace(0, 4, 3, 6, “Parlor”))	"Parlor"

Testing getRandomItemName()	Input	Expected Value
Get a random item name	Space space = new MySpace(0, 0, 3, 3, “Kitchen”) ComputerControlledPlayer (“Leo”, space, 0, 0, 0) space.addItem(MyItem(“Revolver”, 3))	" Revolver "

Testing getRandomSpaceName()	Input	Expected Value
Get a random space name	Space space = new MySpace(0, 0, 3, 3, “Kitchen”) ComputerControlledPlayer (“Leo”, space, 0, 0, 0)	"Kitchen"

Testing getMostPowerfulItemName ()	Input	Expected Value
Get the most powerful item name	ComputerControlledPlayer (“Leo”, space, 0, 0, 0) player.addItem(MyItem(“Revolver”, 3))	" Revolver "

Testing design for MyItem

Testing construction	Input	Expected Value
Constructor disallows empty name	MyItem(“”, 3)	IllegalArgumentException
Constructor disallows null name	MyItem(null, 3)	IllegalArgumentException
Constructor disallows non-positive damage	MyItem(“Revolver”, 0)	IllegalArgumentException

Testing getName()	Input	Expected Value
Item with normal name	MyItem(“Revolver”, 0, 3)	“Revolver”

Testing getDamage()	Input	Expected Value
Item with positive damage	MyItem(“Revolver”, 0, 3)	3

Testing design for MySpace

Testing construction	Input	Expected Value
Constructor disallows negative row1	MySpace(-1, 0, 3, 3, “Kitchen”)	IllegalArgumentException
Constructor disallows negative col1	MySpace(0, -1, 3, 3, “Kitchen”)	IllegalArgumentException
Constructor disallows the value of row2 to be less than the value of row1	MySpace(0, 0, -3, 3, “Kitchen”)	IllegalArgumentException
Constructor disallows the value of col2 to be less than the value of col1	MySpace(0, 0, 3, -3, “Kitchen”)	IllegalArgumentException
Constructor disallows empty name	MySpace(0, 0, 3, 3, “”)	IllegalArgumentException
Constructor disallows null name	MySpace(0, 0, 3, 3, null)	IllegalArgumentException

Testing getPosition()	Input	Expected Value
Space with correct position	MySpace(0, 0, 3, 3, “Kitchen”)	new int[]{0, 0, 3, 3}

Testing getName()	Input	Expected Value
Space with normal name	MySpace(0, 0, 3, 3, “Kitchen”)	“Kitchen”

Testing addItem(Item item), removeItem(Item item), getItem(String itemName), and getItems()	Operation	Actual Testing

Test multiple methods	Item item = new Item(“Revolver”, 0, 3);	addItem(item);
		assertTrue(getItems().get(0).equals(item));
		assertEquals(getItem(“Revolver”, item)
		removeItem(item);
		assertEquals(getItems().size, 0);

Testing	Operation	Actual Testing

addNeighbor(Space space) and getNeighbors()		
Add a neighbor and get the neighbors list	Space space = new MySpace(0, 0, 3, 3, “Kitchen”);	Space newSpace = new MySpace(0, 4, 3, 6, “Parlor”); space.addNeighbor(newSpace); assertEquals(getNeighbors ().get(0), newSpace)

Testing design for MyWorld

Testing construction	**Input**	**Expected Value**
Constructor disallows empty name	MyWorld(“”, 40, 30, new TargetCharacter(“Leo”, 50), new ArrayList<Space>(20), new ArrayList<Item>(20), 0)	IllegalArgumentException
Constructor disallows non-positive rows	MyWorld(“Mansion”, 0, 30, new TargetCharacter(“Leo”, 50), new ArrayList<Space>(20), new ArrayList<Item>(20), 0)	IllegalArgumentException
Constructor disallows non-positive cols	MyWorld(“Mansion”, 40, 0, new TargetCharacter(“Leo”, 50), new ArrayList<Space>(20), new ArrayList<Item>(20), 0)	IllegalArgumentException

Testing construction	**Testing ideas**
Constructor disallows items with incorrect position	The position of an item should be less than the length of the spaceList.
Constructor disallows overlapping spaces	A space should not be overlapped by any other one. (i.e., the row1(row2) value of a space should not be between any other space’s row1 and row2 if its col1(col2) value is between any other space’s col1 and col2.
Constructor disallows space beyond boundaries	The row2 of a space should be less than the rows of the world, and the col2 of a space should be less than the cols of the world.

Testing getName()	**Input**	**Expected Value**
World with normal name	MyWorld(“Mansion”, 40, 30, new TargetCharacter(“Leo”, 50), new ArrayList<Space>(20), new	“Mansion”

	ArrayList<Item>(20), 0)	

Testing getSize()	**Input**	**Expected Value**
World with positive size	MyWorld (“Mansion”, 40, 30, new TargetCharacter(“Leo”, 50), new ArrayList<Space>(20), new ArrayList<Item>(20), 0)	new int[]{40, 30}

Testing getTargetCharacterPosition()	**Input**	**Expected Value**
The target character starts in space 0	MyWorld (“Mansion”, 40, 30, new TargetCharacter(“Leo”, 50), new ArrayList<Space>(20), new ArrayList<Item>(20), 0)	0

Testing getNeighbors(Space space)	**Input**	**Expected Value**
Get the neighbors of a specified Space	getNeighbors(MySpace mySpace)	ArrayList of the neighbors of mySpace on the generated world map

Testing getPlayers()	**Input**	**Expected Value**
Get the players of the game	getPlayers()	ArrayList of the players of the game

Testing getTargetCharacter()	**Input**	**Actual Testing**
Get the target character	MyWorld (“Mansion”, 40, 30, new TargetCharacter(“Leo”, 50), new ArrayList<Space>(20), new ArrayList<Item>(20), 0)	assertEquals (getTargetCharacter. getName(), “Leo”); assertEquals (getTargetCharacter. getHealth(), 50);

Testing moveTargetCharacter()	**Input**	**Operation**	**Actual Testing**
Move the target character from	MyWorld (“Mansion”, 40, 30, new TargetCharacter(“Leo”, 50),	move Target	assertEquals (getTargetCharacterP

space to space in order	new ArrayList<Space>(20), new ArrayList<Item>(20), 0)	Chara cter()	osition(), 1)

Testing movePlayer()	**Testing ideas**
Move the player from current space to a neighbor space	Test the player position before and after moving

Testing pickUpItem()	**Testing ideas**
Pick up an item from current space	Test the items of current player before and after picking up the item

Testing lookAround()	**Testing ideas**
Look around the space the player are currently occupying	Test if the displayed information is equal to expected texts

Testing movePet()	**Testing ideas**
Move the pet to a specified space	Test if the pet is in the specified space after being moved

Testing makeAnAttempt()	**Testing ideas**
Make an attempt on the target character's life	Test if the target character's life is reduced after being attacked

Testing displaySpaceInformation()	**Testing ideas**
Display the information of a specified space	Test if the displayed information is equal to expected texts

Testing displayPlayerInformation()	**Testing ideas**
Display the information of a specified player	Test if the displayed information is equal to expected texts

Testing generateMap()	**Testing ideas**
Generate a map of current world	Test if the map is successfully generated

Testing design for GameController

Testing start()	Testing ideas
Start the game	Use an expected text to test whether the game can successfully complete a run

Testing design for AddComputerPlayer

Testing execute()	Testing ideas
Add a computer controlled player	Test whether a computer controlled player is successfully added to the game after execute()

Testing design for AddHumanPlayer

Testing execute()	Testing ideas
Add a human controlled player	Test whether a human controlled player is successfully added to the game after execute()

Testing design for DisplayPlayerInformation

Testing execute()	Testing ideas
Display a player's information	Test whether a player's information is correctly displayed after execute()

Testing design for DisplaySpaceInformation

Testing execute()	Testing ideas
Display a space's information	Test whether a space's information is correctly displayed after execute()

Testing design for GenerateMap

Testing execute()	Testing ideas
Generate the world map	Test whether the map is successfully generated after execute()

Testing design for LookAround

Testing execute()	Testing ideas
Display the neighboring information of the space the player are currently occupying	Test whether the current space's neighboring information is correctly displayed after execute()

Testing design for MovePlayer

Testing execute()	Testing ideas
Move the player from current space to a neighbor space	Test whether the player moves to expected space after execute()

Testing design for PickUpItem

Testing execute()	Testing ideas
Pick up an item from current space	Test whether the chosen item is successfully picked up by the player after execute()

Testing design for movePet

Testing movePet()	Testing ideas
Move the pet to a specified space	Test if the pet is in the specified space after execute()

Testing design for makeAnAttempt

Testing makeAnAttempt()	Testing ideas
Make an attempt on the target character's life	Test if the target character's life is reduced after execute()