World

<<interface>>
World

+ getNeighbors(name: String): List<String>
+ getNeighbors(index: int): List<String>
+ getSpace(name: String): Space
+ getSpace(index: int): Space
+ getTargetPosition(): Space
+ moveTarget(): Space
+ showGraphicalImage(directory: String)

Target

- health: int
- name: String
- position: int

+ Target(health: int, name: String)
+ getHealth(): int
+ decreaseHealth(damage: int): int
+ getName(): String
+ getPosition(): int
+ setPosition(pos: int)

WorldImpl

- matrix: int[][]
- name: String
- target: Target
- spaces: List<Space>

+ WorldImpl(in: Readable)
+ WorldImpl(row: int, col: int, name: String, target: Target, baseSpaces: List<BaseSpace>, baseWeapons: List<BaseWeapon>)
+ getNeighbors(name: String): List<String>
+ getNeighbors(index: int): List<String>
+ getSpace(name: String): Space
+ getSpace(index: int): Space
+ getTargetPosition(): Space
+ moveTarget(): Space
+ showGraphicalImage(directory: String)
- initMatrixAndValidateSpaces(m: int, n: int, baseSpaces: List<BaseSpace>)
- initNeighborMapAndSpaces(m: int, n: int, baseSpaces: List<BaseSpace>, baseWeapons: List<BaseWeapon>)
- getNeighborsFormMatrix(baseSpace: baseSpace, baseSpaces: List<BaseSpace>, m: int, n: int): List<BaseSpace>
- getNeighborNamesFromSpace(space: Space): List<String>

BaseWeapon

- space: int
- damage: int
- name: String

+ BaseWeapon(s: int, d: int, str: String)
+ getSpaceIndex(): int
+ getDamage(): int
+ getName(): String

Weapon

- belongTo: BaseSpace

+ Weapon(base: BaseWeapon, sapce: BaseSapce)
+ getBelongTo(): BaseSpace
+ setBelongTo(space: BaseSpace)

Space

- neighbors: List<BaseSpace>
- weapons: List<Weapon>

+ Space(base: BaseSpace, neighbors: List<BaseSpace>, weapons: List<Weapon>)
+ getNeighbors(): List<Space>
+ setNeightbors(neighbors: List<Space>)
+ getWeapons(): List<Weapon>
+ setWeapons(weapons: List<Weapon>)

BaseSpace

- startX: int
- startY: int
- endX: int
- endY: int
- order: int
- name: String

+ BaseSpace(x1: int, y1: int, x2: int, y2: int, order: int, name: String)
+ getStart(): int[]
+ getEnd(): int[]
+ getOrder(): int
+ getName(): String

Test Plan for Project 1 (The World)

BaseSpace class:

Success cases:

1. Test construct BaseSpace with valid coordinates and valid order, expect success.
2. Test get name function of BaseSpace, expect returning the correct name.
3. Test get start coordinates function of BaseSpace, expect returning the correct start coordinates.
4. Test get end coordinates function of BaseSpace, expect returning the correct end coordinates.
5. Test get order function of BaseSpace, expect returning the correct order index.

Exception cases:

1. Test construct BaseSpace with invalid Coordinates and valid order, expect IllegalArgumentException.
2. Test construct BaseSpace with valid coordinates and invalid order, expect IllegalArgumentException.

Space class:

Success cases:

1. Test set neighbors function of Space, expect setting success.
2. Test get neighbors function of Space, expect returning the correct neighbors.
3. Test set weapons function of Space, expect setting success.
4. Test get weapons function of Space, expect returning the correct weapons.

BaseWeapon class:

Success cases:

1. Test construct BaseWeapon with valid space index and valid damage, expect success.
2. Test get space index function of BaseWeapon, expect returning the correct space index.
3. Test get damage function of BaseWeapon, expect returning the correct damage value.
4. Test get name function of BaseWeapon, expect returning the correct name.

Exception cases:

1. Test construct BaseWeapon with invalid space index and valid damage, expect IllegalArgumentException.
2. Test construct BaseWeapon with valid space index and invalid damage, expect IllegalArgumentException.

Weapon class:

Success cases:

1. Test set belong to space function of Weapon, expect setting success.
2. Test get belong to space function of Weapon, expect returning the correct space.

Target class:

Success cases:

1. Test construct Target with valid health, expect success.
2. Test get health function of Target, expect returning the correct health value.
3. Test decrease health function of Target, expect returning the correct health value after decrease.
4. Test decrease health function with damage more than current health, expect health to be 0 after decrease.
5. Test get name function of Target, expect returning the correct name.
6. Test get position function of Target, expect returning the correct space index.

7. Test set position function of Target, expect setting success.

Exception cases:

3. Test construct Target with invalid health, expect IllegalArgumentException.

WorldImpl calss:

Success cases:

1. Test construct WorldImpl with valid row, valid column and valid spaces, expect success
2. Test get space's neighbors function with valid space name of World, expect returning the correct neighbors' names.
3. Test get space's neighbors function with invalid space name of World, expect returning an empty list.
4. Test get space's neighbors function with valid space index of World, expect returning the correct neighbors' names.
5. Test get space's neighbors function with invalid space index of World, expect returning an empty list.
6. Test get space function with valid space name of World, expect returning the correct weapons and neighbors.
7. Test get space function with invalid space name of World, expect returning null.
8. Test get space function with valid space index of World, expect returning the correct weapons and neighbors.
9. Test get space function with invalid space index of World, expect returning null.
10. Test get target position function of World, expect returning the correct space info.
11. Test move target function of World, expect returning the correct space info after moving.
12. Test move target function of World, make moves at the last space, expect returning the 0^{th} space.
13. Test graphical image rendering of World, expect the correct image.

Exception cases:

1. Test construct WorldImpl with invalid row, valid column, valid spaces, expect IllegalArgumentException.
2. Test construct WorldImpl with valid row, invalid column, valid spaces, expect IllegalArgumentException
3. Test construct WorldImpl with valid row, valid column and invalid spaces, expect IllegalArgumentException. The space in the space list contains start coordinates which is more than the row or column.
4. Test construct WorldImpl with valid row, valid column and invalid spaces, expect IllegalArgumentException. The space in the space list contains end coordinates which is more than the row or column.
5. Test construct WorldImpl with valid row, valid column and invalid spaces, expect IllegalArgumentException. The space in the space list overlaps with another one in the list.