# 1.Class Checklist

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Attribute** | **method** | **method implement an interface** | **method explanation** |
| Space | id | Space(int,String,int[],int[]) - constructor |  | create a space by pass id, size info,and name |
| name | getID():int | SpaceInterface | retrieve Space Index(0toN) |
| upleft | getUpLeft():Int[2] | SpaceInterface | retrieve upleft corner of space |
| lowright | getDownRight():int[2] | SpaceInterface | retrieve downright corner |
| items:List[Item] | getName():String | SpaceInterface | retrieve name of space |
| neighbors:list[Space] | setItem(Item) |  | add Item into the space, add once every time |
| seens:list[Space] | getItems():List[Item] |  | retrieve item list in the space |
|  | removeItem(Item) |  | remove an item from the space |
|  | calcNeighbors(List[]:Space) |  | calculate and set the neighbors of this space |
|  | calcSeens(List[]:Space) |  | set which space can be seen from this space |
|  | getNeighbors():List[] |  | get list of neighbor spaces |
|  | getSeens():List[] |  | get list of spaces can be seen from this space |
|  | toString():String |  | Return string format “id:{\id} name:{\name} leftcorne:{\upleft} rightcorner:{\lowright}” |
| spaceFactory |  | createSpace()-static method |  | use createSpace to create a space through Factory class rather than directly using constructor |
| Item | id | Item(id,String,int) - constructor |  | create a Item by passing id,name,damage |
| name | getID():int | ItemInterface | retrieve Index of this Item |
| damage:int | getName():String | ItemInterface | retrieve name of this Item |
| space:Space | getDamage():int | ItemInterface | retrieve damage value of this Item |
|  | setSpace(Space) |  | set in which the space should be placed the item |
|  | getSpace():Space | ItemInterface | get space of which the item placed in |
|  | toString():String |  | Return string format “id:{\id} name:{\name} damage:{\damage}” |
| ItemFactory |  | createItem()-static method |  | use createItem to create a item through Factory class rather than directly using constructor |
| Target | name | Target(String,int) - constructor |  | create target character by passing name,health |
| health:int | getName():String | CharacterInterface | retrieve name of character |
| space:Space | getSpace():Space | CharacterInterface | retrieve space that character settling in at the moment |
|  | getHealth():int | CharacterInterface | retrieve the health value of the character |
|  | move(index:int,List spaces) | CharacterInterface | Move the target to the index specified space of space list and settle the attribute space |
|  | toString():String |  | Return string format “name:{\name} health:{\health}” |
| TheWorld | name | TheWorld(String,int,int) - constructor |  | Create the game:Spaces/Charactor/items by a specification file |
| row | getColumn():int | MapInterface | retrieve the total column of map |
| column | getRow():int | MapInterface | retrieve the total row of map |
| spaces:List[Space] | getName():String |  | retrieve the name of this game |
|  | addSpace(Space) |  | add a space to the world |
| items:List[Item] | getSpaces():List[Space] | MapInterface | retrieve the 0-indexed space list int the map |
|  | addItem(Item) |  | add an item to the world |
| target::Target | getItems():List[Item] |  | retrieve the 0-indexed item list int this game |
|  | setTarget():void |  | set target character to the world |
|  | getTarget():Target |  | retrieve the target charactor int this game |
|  | draw() | MapInterface | use BufferImage to draw the image of the map and store it in a file |
|  | moveTarget(List spaces, int stop,Enum mode) |  | Move the target by specified mode , under sequence mode, the character move from start index space to the last index space by sequential order,we should provide scalability to other mode(like random etc.) Stop arg controls how many steps does target move, target will stop at specified step, default is never stop. |
|  | toString():String |  | Return string format “name:{\name} row:{\row} column:{\column}” |

# 2.Test cases design

## 2.1 Space

firstly create a new space by call constructor function Space( id:int , name:String, upLeft:int , downRight:int ) by pass different arguments to test constructor and toString method.

|  |  |  |
| --- | --- | --- |
| **Test constructor and toString()** | **Input** | **Expected Value** |
| Normal case | Space(1,”bathroom”,new int[]{2,3},new int[]{5,8}).toString() | “id:1 name:bathroom leftcorner:2,3 rightcorner:5,8” |
| Space index 0 | Space(0,”bathroom”,new int[]{2,3},new int[]{5,8}).toString() | “id:0 name:bathroom leftcorner:2,3 rightcorner:5,8” |
| Left corner 0,0 | Space(1,”bathroom”,new int[]{0,0},new int[]{5,8}).toString() | “id:1 name:bathroom leftcorner:0,0 rightcorner:5,8” |
| Id < 0 | Space(-1,”bathroom”,new int[]{0,0},new int[]{5,8}).toString() | Throws IllegalArgumentsException |
| No name pass in | Space(1,,new int[]{0,0},new int[]{5,8}).toString() | Throws IllegalArgumentsException |
| Left corner <0 | Space(1,”bathroom”,new int[]{-1,0},new int[]{5,8}).toString() | Throws IllegalArgumentsException |
| Left corner is righter than right corner | Space(1,”bathroom”,new int[]{-10,0},new int[]{5,8}).toString() | Throws IllegalArgumentsException |
| Left corner is higher than right corner | Space(1,”bathroom”,new int[]{-1,10},new int[]{5,8}).toString() | Throws IllegalArgumentsException |

Create a normal Space instance of **Space(1,”bathroom”,new int[]{2,3},new int[]{5,8})** and test get,set method

|  |  |  |
| --- | --- | --- |
| **Test setItem removeItem and get method** | **Input** | **Expected Value** |
| getID() | Space(1,”bathroom”,new int[]{2,3},new int[]{5,8}) | 1 |
| getName() | above | “bathroom” |
| getUpLeft() | above | Int[]{2,3} |
| getDownRight() | above | Int[]{5,8} |
| Set an Item into space:  setItem(Item) | setItem(New Item(0,”clamper”,5)) | “id:0 name:clamper,damage:5” |
| getItems()[0].toString() |
| multi items case:  getItems()[0].toString()  getItems()[1].toString() | setItem(New Item(0,”clamper”,5))  setItem(New Item(2,”mop”,2)) | “id:0 name:clamper,damage:5”  “id:2 name:mop,damage:2” |
| Remove item case：  removeItem(Item) | item1=setItem(New Item(0,”clamper”,5))  item2=setItem(New Item(2,”mop”,2));  removeItem(item1) | **getItems()[0].toString()**:  “id:2 name:mop,damage:2” |
| Remove all cases | item1=setItem(New Item(0,”clamper”,5))  item2=setItem(New Item(2,”mop”,2));  removeItem(item1)  removeItem(item2) | **getItems().size()**:  0 |
| No item in the room , but try to  Print items info | Space(1,”bathroom”,new int[]{2,3},new int[]{5,8})  Then directly:  Print(getItems[0].toString()) | Throws NullPointerException |

Create more than 1 spaces ,and make sure some of them are neighbor(share at least one “wall”), some of them are not . test calcNeighbors() which function set neighbors to a space, and getNeighbors();

Create more tha 1 spaces , make sure some of them has continous ID , some are not , if 2 space have continous ID, that means they can be seen by each other, otherwise , they can’t. Test calcSeens() which set spaces can be seen by specified space, then getSeens().

|  |  |  |
| --- | --- | --- |
| **Test calcNeighbors(), getNeighbors(),calcSeens() and getSeens()** | **Input** | **Expected Value** |
| calcNeighbors(List spaces) | space=new Space(1,”bathroom”,new int[]{2,3},new int[]{5,8});  list={new Space(2,”kitchen”,new int[]{5,3},new int[]{10,8}),  New Space(3,”living room”,new int[]{15,9},new int[]{22,11})};  space.calcNeighbors(list) | **space.getNeighbors(list).size():**  1  **space..getNeighbors(list)[0].toString():**  “id:2 name:kitchen leftcorner:5,3 rightcorner:10,8” |
| getNeighbors() |
| 0 neighbor | space=new Space(1,”bathroom”,new int[]{2,3},new int[]{5,8});  list={new Space(2,”kitchen”,new int[]{11,3},new int[]{12,8}),  New Space(3,”living room”,new int[]{15,9},new int[]{22,11})}; | **space.getNeighbors(list).size():**  0 |
| More than 1 neighbor | space=new Space(1,”bathroom”,new int[]{2,3},new int[]{5,8});  list={new Space(2,”kitchen”,new int[]{5,3},new int[]{10,8}),  New Space(3,”living room”,new int[]{3,8},new int[]{20,11})}; | **space.getNeighbors(list).size():**  2  **space..getNeighbors(list)[0].toString():**  “id:2 name:kitchen leftcorner:5,3 rightcorner:10,8”  **space..getNeighbors(list)[1].toString():**  “id:3 name:living room leftcorner:3,8 rightcorner:20,11” |
| calcSeens:  list[0].calcSeens(list)  list[1].calcSeens(list)  List[2].calcSeens(list) | list={new Space(1,”bathroom”,new int[]{2,3},new int[]{5,8}),  {new Space(2,”kitchen”,new int[]{5,3},new int[]{10,8}),  New Space(3,”living room”,new int[]{3,8},new int[]{20,11})}; |  |
| getSeens(list[0]) | above | **getSeens(list[0]).size():**  1  **getSeens(list[0])[0].getName():**  “kitchen” |
| getSeens(list[1]) | above | **getSeens(list[1]).size():**  2  **getSeens(list[1])[0].getName():**  “bathroom”  **getSeens(list[1])[1].getName():**  “living room” |
| getSeens(list[2]) | above | **getSeens(list[2]).size():**  1  **getSeens(list[2])[0].getName():**  “kitchen” |

## 2.2 Item

Create an item by constructor, then test get , set method

|  |  |  |
| --- | --- | --- |
| **Test constructor, get and set method of Item class** | **Input** | **Expected Value** |
| Test constructor:normal case | Item(1,”mop”,2) | **Item(1,”mop”,2).toString():**  “id:1 name:mop damage:2” |
| Id < 0 | Item(-1,”mop”,2) | Throws IllegalArgumentsException |
| No name passed | Item(-1,,2) | Throws IllegalArgumentsException |
| Damage <0 | Item(1,”mop”,-2) | Throws IllegalArgumentsException |
| Test get method:  getID() | Item(1,”mop”,2) | 1 |
| getName() | above | “mop” |
| getDamage() | above | 2 |
| Set in which space the item placed:  setSpace(Space space) | setSpace(new Space(1,”bathroom”,new int[]{2,3},new int[]{5,8}) | **getSpace().getName():**  “bathroom” |
| getSpace() |
| No space been set | Item(1,”mop”,2)  Then call getSpace() | Throws NullPointerException |

## 2.3 Target

Create a target(implements character interface) By constructor, test constructor and get methods

|  |  |  |
| --- | --- | --- |
| **Test constructor, get class** | **Input** | **Expected Value** |
| Test constructor:normal case | Target(”lucky”,200) | **Target(“lucky”,200).toString():**  “name:lucky health:200” |
| No name passed | Target(””,200) | Throws IllegalArgumentsException |
| Health <1 | Target(”lucky”,0) | Throws IllegalArgumentsException |
| Health <0 | Target(”lucky”,-1) | Throws IllegalArgumentsException |
| getName() | Target(”lucky”,200) | “lucky” |
| getHealth() | above | 200 |

Then, we can test move() , the character can change the space he/she settled through move(). He/She change the private attribute “space” by a specified mode when everytime we call the move(), that means move() can return a Space and fill the space attribute of the target instance.The modes of move() are stored in a enum, but by far, there is only one mode, sequence mode, that allow target move in the spaces by increasing order from 0 to N(N is the last index of all spaces).

|  |  |  |
| --- | --- | --- |
| **Test move, getSpace** | **Input** | **Expected Value** |
| move from space to no.1, by sequence mode, then getSpace | list = {new Space(0,”bathroom”,new int[]{2,3},new int[]{5,8}),new Space(1,”kitchen”,new int[]{5,3},new int[]{10,8}),  New Space(2,”living room”,new int[]{3,8},new int[]{20,11})}  space =move(1,list) | **target.getSpace().toString():**  “id:1 name:kitchen leftcorner:5,3 rightcorner:10,8” |
| Move twice | For (i=0,i<2,i++){  Space=move(i,list)  } | **target.getSpace().toString():**  “id:2 name:living room leftcorner:3,8 rightcorner:20,11” |
| Move thrice, out of the boundry of space list | For (i=0,i<3,i++){  Space=move(i,list)  } | **Throws IllegalArgumentsException** |

## 2.4 TheWorld

Create TheWorld by constructor that pass into a simple specification file , the file is like this(3 spaces):

It is a 5 spaces mansion with 7 items, it is saved as Mansion.txt, In Driver class ,we parser the file by line number, and construct TheWorld instance and Spaces/items/target that associated with TheWorld

|  |  |  |
| --- | --- | --- |
| **Test constructor, get/set class** | **Input** | **Expected Value** |
| Test constructor:normal case | world=TheWorld(”lucky’s mansion”,200,100) | **world.toString():**  “name:lucky’s mansion rows:200 columns:100” |
| No name passed | TheWorld(””,200,100) | Throws IllegalArgumentsException |
| rows<1 | TheWorld(”lucky’s mansion”,0,50) | Throws IllegalArgumentsException |
| rows<0 | TheWorld(”lucky’s mansion”,-1,50) | Throws IllegalArgumentsException |
| columns<1 | TheWorld(”lucky’s mansion”,100,0) | Throws IllegalArgumentsException |
| columns<0 | TheWorld(”lucky’s mansion”,100,-1) | Throws IllegalArgumentsException |
| Test get method:  getName() | world=TheWorld(”lucky’s mansion”,200,100) | “lucky’s mansion” |
| getRows() | above | 200 |
| getColumns() | above | 100 |
| addSpace(space) | addSpace(new Space(0,”bathroom”,new int[]{2,3},new int[]{5,8})) | **getSpaces()[0].toString():**  “id:0 name:bathroom leftcorner:2,3 rightcorner:5,8” |
| getSpaces | getSpaces()[0] | above |
| add more than 1 space | addSpace(new Space(0,”bathroom”,new int[]{2,3},new int[]{5,8}));  addSpace(new Space(1,”kitchen”,new int[]{8,15},new int[]{9,19})); | **getSpaces()[0].toString():**  “id:0 name:bathroom leftcorner:2,3 rightcorner:5,8”  **getSpaces()[1].toString():**  “id:1 name:kitchen  leftcorner:8,15 rightcorner:9,19” |
| addItem(space) | addItem(new Item(0,”mop”,10)) | **getItems()[0].toString():**  “id:0 name:mop damage:10” |
| getItems | above | above |
| Add more than 1 item | addItem(new Item(0,”mop”,10))  addItem(new Item(1,”helmet”,2)) | **getItems()[0].toString():**  “id:0 name:mop damage:10”  **getItems()[1].toString():**  “id:0 name:helmet damage:2” |
| addTarget(Target) | addTarget(new Target(“Lucky”,200)) | **getTarget().toString():**  “name:Lucky health:200” |
| getTarget | above | above |

The target can move through the spaces in specified mode , right now there is only one moving mode called “sequence”, it stored in a enum named Mode

|  |  |  |
| --- | --- | --- |
| **Test moveTarget()** | **Input** | **Expected Value** |
| Test moveTarget(list,enum),stop at first step | list = {new Space(0,”bathroom”,new int[]{2,3},new int[]{5,8}),new Space(1,”kitchen”,new int[]{5,3},new int[]{10,8}),  New Space(2,”living room”,new int[]{3,8},new int[]{20,11})}  space =moveTarget(list,Mode.Sequence,1) | **space.toString():**  “id:0 name:bathroom leftcorner:2,3 rightcorner:5,8” |
| Test moveTarget(list,enum),stop at second step | space =moveTarget(list,Mode.Sequence,2) | **space.toString():**  “id:1 name:kitchen  leftcorner:5,3 rightcorner:10,8” |
| Stop at 5 step, move back to kitchen | space =moveTarget(list,Mode.Sequence,5) | **space.toString():**  “id:1 name:kitchen  leftcorner:5,3 rightcorner:10,8” |
| Stop at 7 step, move back to kitchen | space =moveTarget(list,Mode.Sequence,7) | **space.toString():**  “id:0 name:bathroom leftcorner:2,3 rightcorner:5,8” |
| Stop at 9 step, move back to kitchen | space =moveTarget(list,Mode.Sequence,9) | **space.toString():**  “id:2 name:living room leftcorner:3,8 rightcorner:20,11” |