Package FunctionC_TomCatchJerry

Class Character

java.lang.Object
    FunctionC_TomCatchJerry.Character

```
public class Character
extends Object
```

Character class is used to record and update the location of an instance according to the key pressing or algorithm.

Field Summary

Fields

Modifier and Type	Field	Description
boolean	Game_state	
int	lastPos	
int[]	location	
int	newCol	
int	newRow	
int	speed	

Constructor Summary

Constructors

Constructor	Description
Character()	Constructor of Character Initialize all the attributes

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
boolean	IsPath(int row, int col)	Check whether the new location is valid for the object to move
boolean	IsWithinBoundary(int coordinate)	Check whether the new coordinate of the location is valid in maze
void	move()	Player controls the direction of the moving object using keyboard 1.
void	MoveWithShortestPath()	Computer controls the movement of the moving object using algorithm from function B 1.
void	reset(int row, int col)	Reset the status of the character when the game is restarted 1.
static int	toIndex(int[] location)	Convert the 2D location to 1D index based on the length of the maze

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

newCol

```
public int newCol
```

newRow

```
public int newRow
```

lastPos

```
public int lastPos
```

location

```
public int[] location
```

Game_state

```
public boolean Game_state
```

speed

```
public int speed
```

Constructor Details

Character

```
public Character()
```

Constructor of Character Initialize all the attributes

Method Details

move

```
public void move()
```

Player controls the direction of the moving object using keyboard 1. Check whether the new location is valid for moving 1.1 If valid, 1.1.1 Store the previous location 1.1.2 Update the new location 1.1.3 Update the current location to the algorithm for calculating the new shortest path between Tom and Jerry 1.2 If invalid, location remains unchanged

MoveWithShortestPath

```
public void MoveWithShortestPath()
```

Computer controls the movement of the moving object using algorithm from function B 1. Get the current NEXT shortest path 2. Check whether the next step is valid 2.1 If valid, 2.1.1 Update the last position and the current location 2.1.2 Update Tom's Location to the algorithm to find the newest shortest path 2.2 If invalid, location remains unchanged

IsWithinBoundary

```
public boolean IsWithinBoundary(int coordinate)
```

Check whether the new coordinate of the location is valid in maze

Parameters:

`coordinate` - The coordinate of the row/column of the location

Returns:

true if the coordinate is non-negative and within the maze size

IsPath

```
public boolean IsPath(int row,
                      int col)
```

Check whether the new location is valid for the object to move

Parameters:

`row` - The row value of the new location

`col` - The column value of the new location

Returns:

true if the location on the maze is not a BLOCK

toIndex

```
public static int toIndex(int[] location)
```

Convert the 2D location to 1D index based on the length of the maze

Parameters:

`location` - An int array storing the row and column value of a location

Returns:

the index value in 1D

reset

```
public void reset(int row,
                  int col)
```

Reset the status of the character when the game is restarted 1. Store the current location to last position 2. Reset the character's location back to spawn point 3. Reset the newRow and newCol to 0 indicating no movement exists 4. Set the game state to false The value of lastPos and location is used to update the game maze interface

Parameters:

`row` - The row value of the spawn point

`col` - The column value of the spawn point