Class DisasterZone

java.lang.Object
    DisasterZone

```
public class DisasterZone
extends java.lang.Object
```

DisasterZone provides a framework for an open-ended solution using object classes. The DisasterZone class acts as a simulator of the environment that the search and rescue robot will be entered into, for testing purposes. This class represents the unknown environment, the positioning of the robot in the environment, and the results of sensor readings and movement of the robot in the environment. Note - for the purposes of the simulation and for testing robot controller solutions the disaster zone environment is limited to a 102 meter square area.

Version:

(Jan 2022)

Author:

(Nick Kwan)

Constructor Summary

Constructors

Constructor	Description
`DisasterZone()`	Constructor for objects of class DisasterZone.
`DisasterZone (int debris)`	Constructor for objects of class DisasterZone; Intermediate constructor builds a disaster site including the Bot and Target, with randomly placed obstacles.
`DisasterZone (int debris, char type)`	Constructor for objects of class DisasterZone; Advanced constructor builds a disaster site including the Bot and Target, with randomly placed obstacles.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
void	**displayScore**()	public method to calculate and print out an overall evaluation or score for the run/scenario.
java.lang.String	**getBotStatus**()	public method to return the current status for the rescue bot.
java.lang.String	**move**(int dist)	public method to return the current status for the rescue bot after trying to move forward a given distance.
int	**ping**()	public method to return the approximate distance and general direction of the target.
boolean	**retrieveTarget**()	public method to return whether the target has been retrieved successfully or not.
void	**turnLeft**()	public method to check turn the bot 90-degrees to the left.
void	**turnRight**()	public method to check turn the bot 90-degrees to the right.

Methods inherited from class java.lang.Object

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

Constructor Detail

DisasterZone

```
public DisasterZone()
```

Constructor for objects of class DisasterZone.

Parameters:

`Default` - /Basic constructor builds an mostly empty disaster site except for the Bot and Target; Some random obstacles are also placed (roughly 5%); Entry point is the top-left corner;

DisasterZone

```
public DisasterZone(int debris)
```

Constructor for objects of class DisasterZone; Intermediate constructor builds a disaster site including the Bot and Target, with randomly placed obstacles. int parameter defines the density (as a percentage) of random obstacles (debris) placed in the site (debris = 0 means no random debris added); Entry point is a randomly chosen location along the top/north-wall (row 1);

DisasterZone

```
public DisasterZone(int debris,
                    char type)
```

Constructor for objects of class DisasterZone; Advanced constructor builds a disaster site including the Bot and Target, with randomly placed obstacles. int parameter defines the density (as a percentage) of random obstacles (debris) placed in the site (debris = 0 means no random debris added); char parameter defines which type of data file should be used (R = random config; B = building; S = structure); Entry point is a randomly chosen location along the top/north-wall (row 1) or left/west-wall (column 1);

Method Detail

turnLeft

```
public void turnLeft()
```

public method to check turn the bot 90-degrees to the left. each turn uses 1 unit of battery power;

turnRight

```
public void turnRight()
```

public method to check turn the bot 90-degrees to the right. each turn uses 1 unit of battery power;

displayScore

```
public void displayScore()
```

public method to calculate and print out an overall evaluation or score for the run/scenario. This method is provided to help evaluate and compare successful runs using different scenarios. This method only prints out a score - nothing is returned for use in the program. The score is calcuated based on three factors: finding/retrieving the target, exiting the disaster zone, and how much of the disaster zone has been visited/mapped for continued search and rescue missions.

getBotStatus

```
public java.lang.String getBotStatus()
```

public method to return the current status for the rescue bot. each status check uses 5 units of battery power;

Returns:

String -> representing the current state of the robot heading = N/E/S/W + location of bot = [row][col] + sensor readings around bot B's location = 8 chars showing the contents of the 8 spaces surrounding the bot (based on the current direction the bot is facing, moving clockwise around the bot) + battery/fuel levels remaining = [fuel] Example For example, if the bot B is at position 1,42 with 1000 units of battery charge, facing NORTH, where the sensor readings are: OO**.*.O This means the bot can only move to the left/WEST or down/SOUTH (moving NORTH is also possible but the bot would leave/exit the disaster zone) The return String in this case would be "N[1][42]OO**.*.O[1000]"

move

```
public java.lang.String move(int dist)
```

public method to return the current status for the rescue bot after trying to move forward a given distance. each move uses 2 units of battery power for each meter moved forward, plus 5 units for the

status check after the move;

Parameters:

`int` - -> representing the distance attempted to move forward Note - the bot will automatically stop if it runs into an obstacle or if the bot passes by the target

Returns:

String -> representing the current state of the robot

retrieveTarget

```
public boolean retrieveTarget()
```

public method to return whether the target has been retrieved successfully or not. retrieving the target can only be successfully executed if the bot is adjacent to the target; attempting to retrieve the target whether successful or not, uses 10 units of battery power

Returns:

boolean -> representing whether the target was retrieved or not

ping

```
public int ping()
```

public method to return the approximate distance and general direction of the target. each ping uses 100 units of battery power; the ping will only yield positive results if the target is ahead of the robot (90-degree wedge shape in the direction the robot is facing);

Returns:

int -> representing the approximate distance to the target provided that the target is ahead of the bot; 0 if the target has been found and retrieved; -1 otherwise