Class DisasterZone

java.lang.Object
    DisasterZone

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

DisasterZone provides a framework for an open-ended student solution as an APCS final project.

Version:

(May 2018)

Author:

(Nick Kwan)

Constructor Summary

Constructors

Constructor and Description
DisasterZone() Constructor for objects of class DisasterZone; Default/Basic constructor builds an mostly empty disaster site except for the Bot and Target; Some random obstacles are also placed (roughly 2%); Entry point is the top-left corner;
DisasterZone(int debris) Constructor for objects of class DisasterZone; Intermediate constructor builds a disaster site including for the Bot and Target, with randomly placed obstacles; int arg defines the density 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(int debris, char type) Constructor for objects of class DisasterZone; Advanced constructor builds a disaster site including for the Bot and Target, with randomly placed obstacles; int arg defines the density of random obstacles (debris) placed in the site (debris = 0 means no random debris added); char arg 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 Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
void	**displayScore**()

	public method to print out an overall "score" for the run This method is provided to help students evaluate and compare their successful solutions over a number of runs with different scenarios.
java.lang.String	**getBotStatus**() public method to return the current status for the rescue bot each status check uses 1 unit of battery power
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 a unit for the status check after the move
int	**ping**() public method to return the approximate distance and general direction of the target each ping uses 10 units of battery power
void	**turnLeft**() public method to check turn the bot 90-degrees to the left; each turn uses 1 unit of battery power;
void	**turnRight**() public method to check turn the bot 90-degrees to the right; each turn uses 1 unit of battery power;

Methods inherited from class java.lang.Object

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

Constructor Detail

DisasterZone

```
public DisasterZone()
```

Constructor for objects of class DisasterZone; Default/Basic constructor builds an mostly empty disaster site except for the Bot and Target; Some random obstacles are also placed (roughly 2%); 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 for the Bot and Target, with randomly placed obstacles; int arg defines the density 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)
             throws java.io.IOException
```

Constructor for objects of class DisasterZone; Advanced constructor builds a disaster site including for the Bot and Target, with randomly placed obstacles; int arg defines the density of random obstacles (debris) placed in the site (debris = 0 means no random debris added); char arg 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);

Throws:

`java.io.IOException`

Method Detail

displayScore

```
public void displayScore()
```

public method to print out an overall "score" for the run This method is provided to help students evaluate and compare their successful solutions over a number of runs with different scenarios. This method only prints out a score - nothing is returned for use in the program.

Parameters:

-

getBotStatus

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

public method to return the current status for the rescue bot each status check uses 1 unit of battery power

Parameters:

-

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 812 (based on the current direction of the bot 7B3 moving clockwise around the bot) 654 + battery/fuel levels remaining = [fuel] Ex For example, a bot B could start at position 1,42 with 1000 units facing NORTH of battery charge with the sensor readings of OOO .B* *.* where it can only move to the left/WEST or down/SOUTH The return String 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 a unit 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`

ping

```
public int ping()
```

public method to return the approximate distance and general direction of the target each ping uses 10 units of battery power

Parameters:

`-`

Returns:

`int -> representing the approximate distance to the target provided that the target is ahead of the bot -1 otherwise`

turnLeft

```
public void turnLeft()
```

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

Parameters:

`-`

turnRight

```
public void turnRight()
```

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

Parameters:

`-`