rspec

Ryan Durling

February 3, 2012

Contents

1 String

1.1 #is$_i$

1.1.1 SUCCESS Check to see if string is an Integer

1.2 #is$_s$?

1.2.1 SUCCESS Check to see if string is string

2 Fixnum

2.1 #is$_i$?

2.1.1 SUCCESS Check to see if integer is integer

3 Array

3.1 #s$_{toi}$

3.1.1 SUCCESS Change the contents of the array with integers and/or strings

and converts the integers to integers

3.2 #add$_{value}$

3.2.1 SUCCESS "Add value at index 'num' by 'some other num'

3.3 #subtract$_{value}$

3.3.1 SUCCESS "Subtract value at index 'num' by 'some other num'

3.4 #shift$_{right}$

3.4.1 SUCCESS Move elements of the array to the right

by removing the last element and putting it at the beginning

3.5 #shift$_{left}$

3.5.1 SUCCESS Move elements of the array to the left

by removing the first element and putting it at the end

4 Command$_{\text{File}}$

4.1 #new

4.1.1 SUCCESS Returns a new Command$_{\text{File}}$ object

4.1.2 SUCCESS Throws an ArgumentError if passed more than one arguments

4.1.3 SUCCESS It fills an array with lines of text

4.1.4 #process$_{\text{commandfile}}$

- **SUCCESS** Processes the lines of text into a command structure by gathering the commands and creating a command structure
- #gather$_{\text{command}}$
  - **SUCCESS** Gather the lines and splits them into a command array checking for integers and converting them
- #create$_{\text{commandstructre}}$
  - **SUCCESS** Takes the the first element, which is the size of the field, and then takes every odd index putiing every two commands into a command object and pushing them into the command$_{\text{structure}}$ array

4.2 #valid$_{\text{commandstructure}}$?

4.2.1 SUCCESS Return true if commands within the

command$_{\text{structure}}$ array are valid by checking the xy coordinates, text within the file and the command objects

4.2.2 #valid$_{\text{xy}}$?

- **SUCCESS** Return true if xy coordinates given are valid

4.2.3 #valid$_{\text{text}}$?

- **SUCCESS** Return true if the text in the file is correct

4.2.4 #valid$_{\text{commands}}$?

- **SUCCESS** Return true if the command objects within the command$_{\text{structure}}$ array are valid by checking the position and movent
- #valid$_{\text{position}}$?

  - **SUCCESS** Return true if the positin in the command object is valid

- $\#\text{valid}_{\text{movement}}$?
  - **SUCCESS** Return true if the movement in the command object is valid

5 Command

5.1 #new

5.1.1 SUCCESS Returns a new Command object

5.1.2 SUCCESS Throws an ArgumentError if passed less than two arguments

5.2 #position

5.2.1 SUCCESS Returns an array of the position

5.3 #movement

5.3.1 SUCCESS Returns an array of the movement

6 My_{Matrix}

6.1 #new

6.1.1 SUCCESS Returns a new My_{Matrix} object

6.1.2 SUCCESS Throws an ArgumentError if passed less than two arguments

6.1.3 SUCCESS Creates a two dimmensional array of nil

6.2 #[]

6.2.1 SUCCESS Return the xy given within the two dimensional array

6.3 #[]=

6.3.1 SUCCESS Place the value at the xy given within the two dimensional array

6.4 $\#my_{matrix}$

6.4.1 SUCCESS Returns the two dimensinal matrix

7 Navigation

7.1 $\#Navigation.VALID_{COMMANDS}$

7.1.1 SUCCESS Returns array of $VALID_{COMMANDS}$

7.2 #Navigation.COMPASS

7.2.1 SUCCESS Returns array COMPASS

7.3 #Navigation.DIRECTION

7.3.1 SUCCESS Returns array DIRECTION

7.4 $\#Navigation.execute_{move}$[8]

7.4.1 SUCCESS Returns new coordinate and heading based upon given instruction

7.4.2 #Navigation.move

- **SUCCESS** Moves coordinate based on heading

7.4.3 #Navigation.change$_{\text{heading}}$

- **SUCCESS** Turns the coordinate left or right
  based on current heading and instruction

- #Navigation.calibrate$_{\text{compass}}$

  - **SUCCESS** Move the compass heading so it matches
    the heading of the current coordinate

8 Plateau

8.1 #new

8.1.1 SUCCESS Returns a new Plateau object

and adds one to the x,y

8.1.2 SUCCESS Throws an ArgumentError if passed the wrong number of arguments

9 Rover$_{\text{Controller}}$

9.1 #new

9.1.1 SUCCESS Returns a new instance of a Rover$_{\text{Controller}}$ object

9.1.2 SUCCESS It takes the first element of the command$_{\text{structure}}$ array

and creates a new two dimensional array

9.1.3 #create$_{\text{rover}}$

- **SUCCESS** Create the rovers based on the number of command objects
  given

9.1.4 #initial$_{\text{placement}}$

- **SUCCESS** Place the rovers based on the coordinates
  provided by the command object

- #pass$_{\text{initialplacementcheck}}$

  - **SUCCESS** Return true if not placing one rover atop another

  - **SUCCESS** Throw error if there is a collision in placement

- #pass$_{\text{collisioncheck}}$
  - **SUCCESS** Return true if there is no collision

9.2 #explore

9.2.1 SUCCESS Each rover moves its full path one at a time

9.2.2 #pass$_{\text{pathcheck}}$?

- **SUCCESS** Check to see if the path is valid

- **SUCCESS** Raise an error if path is not within the boundary or if there is a collision along the path

- #pass$_{\text{boundarycheck}}$?
  - **SUCCESS** Return true if the coordinate is within the prescribed boundary

9.2.3 #move

- **SUCCESS** Move a rover based on the command list given from the file by removing it from the map and placing it in its new spot

- #remove$_{\text{rover}}$
  - **SUCCESS** Remove the rover from the map coordinate it inhabits

- #place$_{\text{rover}}$
  - **SUCCESS** Place the rover in its new coordinates

10 Rover

10.1 #new

10.1.1 SUCCESS Return a new Rover object

10.1.2 SUCCESS Throws an ArgumentError if wrong number of arguments

10.2 #move

10.2.1 SUCCESS Logs the place the rover was in and

Return new coordinates based on instruction

10.3 #home$_{\text{coordinate}}$

10.3.1 SUCCESS Returns the coordinates of where the rover started

10.4 #tag

10.4.1 SUCCESS Return the tag number of the rover

10.5 #command

10.5.1 SUCCESS Return the list of instructions for the rover

10.6 #name

10.6.1 SUCCESS Returns the name of the rover

10.7 #coordinate

10.7.1 SUCCESS Returns the coordinates of rover

10.8 #coordinate=

10.8.1 SUCCESS Sets the coordinate of the rover

11 Summary

Finished in **0.015143 seconds** 62 examples, 0 failures