

# **abhandar\_homework\_3: Roadrunner Coyote Simulation**

AUTHOR  
Version 1.0  
05/06/2019



# Table of Contents

Table of contents



# Hierarchical Index

## Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

agent.....	5
coyote.....	11
roadrunner.....	27
config.....	9
grid.....	16
simulation.....	32

# Class Index

## Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<b>agent (Agent class )</b> .....	5
<b>config (Configuration class )</b> .....	9
<b>coyote (Coyote class )</b> .....	11
<b>grid (Grid class of type singleton )</b> .....	16
<b>roadrunner (Road-runner class. A template for a road-runner )</b> .....	27
<b>simulation (Simulation class. A template for the simulation )</b> .....	32

# File Index

## File List

Here is a list of all files with brief descriptions:

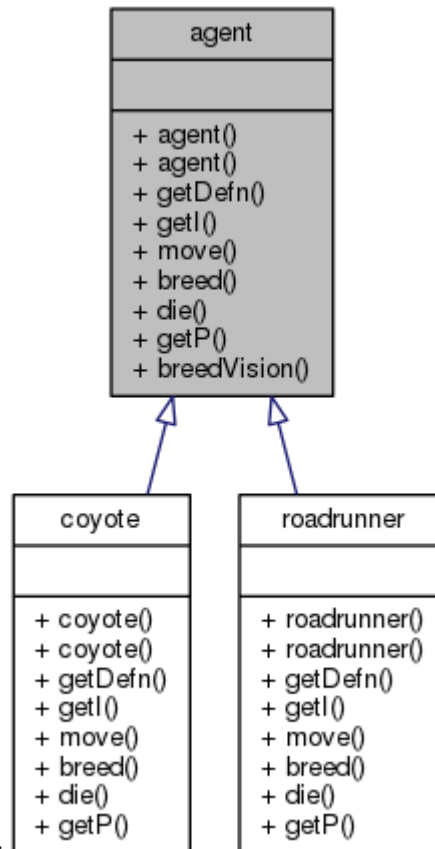
<b>agent.java</b>	.....35
<b>config.java</b>	.....36
<b>coyote.java</b>	.....37
<b>grid.java</b>	.....38
<b>roadrunner.java</b>	.....39
<b>simulation.java</b>	.....40

# Class Documentation

## agent Class Reference

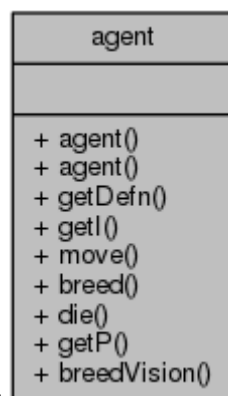
Agent class.

Inheritance diagram for agent:



IMAGE

Collaboration diagram for agent:



IMAGE

### Public Member Functions

**agent ()**

*Agent constructor without parameters Agents created this way are considered spaces.*

**agent (int iLoc)**



*Agent constructor with parameters. The agent will be defined as the parameter provided to this constructor.*

int **getDefn** ()

*Definition getter.*

int **getI** ()

*Gets the index of the agent.*

void **move** ()

*Handles the movement of the agent. For the agent class this function does nothing because we consider undefined agents to be spaces in this simulation.\ Function is defined for children.*

void **breed** ()

*Handles the breeding process of the agent For the agent class this function does nothing because we consider undefined agents to be spaces in this simulation. Function is defined for children.*

void **die** ()

*Handles the death of the agent For the agent class this function does nothing because we consider undefined agents to be spaces in this simulation. Function is defined for children.*

int **getP** ()

*Gets the previous location of the road-runner.*

int **breedVision** (int i)

*Function that helps the agent make a decision on where to breed.*

---

## Detailed Description

Agent class.

A template for agent. An agent in this simulation has a definition of 0 which is equivalent to a space in the grid.

Definition at line 8 of file agent.java.

---

## Constructor & Destructor Documentation

### agent.agent ()

Agent constructor without parameters Agents created this way are considered spaces.

Definition at line 22 of file agent.java.

### agent.agent (int iLoc)

Agent constructor with parameters. The agent will be defined as the parameter provided to this constructor.

#### Parameters:

<i>d</i>	The agent will be defined as d.
----------	---------------------------------

Note: Use of a custom definition requires manually changing configuration values. Use not advised.

Definition at line 32 of file agent.java.

---

## Member Function Documentation

### **void agent.breed ()**

Handles the breeding process of the agent For the agent class this function does nothing because we consider undefined agents to be spaces in this simulation. Function is defined for children.

Definition at line 64 of file agent.java.

Here is the caller graph for this function:



### **int agent.breedVision (int i)**

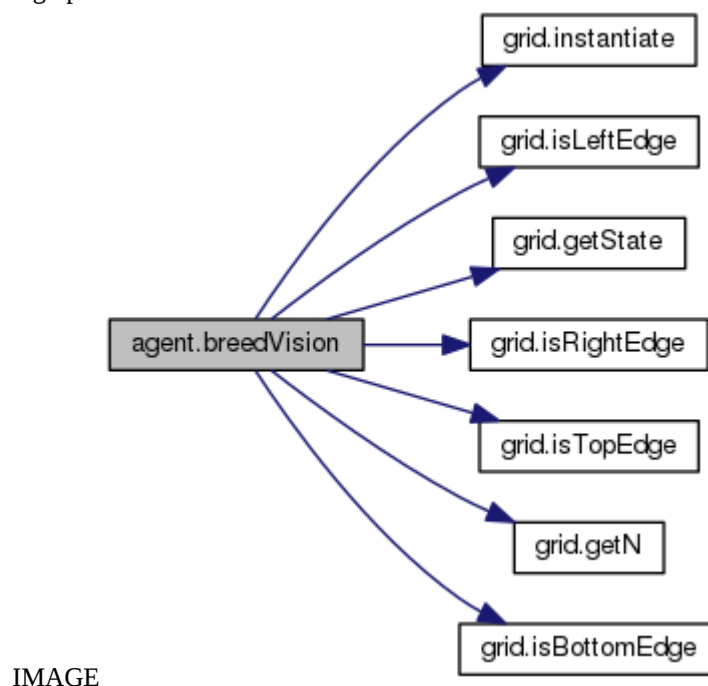
Function that helps the agent make a decision on where to breed.

#### **Returns:**

void

Definition at line 92 of file agent.java.

Here is the call graph for this function:



### **void agent.die ()**

Handles the death of the agent For the agent class this function does nothing because we consider undefined agents to be spaces in this simulation. Function is defined for children.

Definition at line 71 of file agent.java.

### **int agent.getDefn ()**

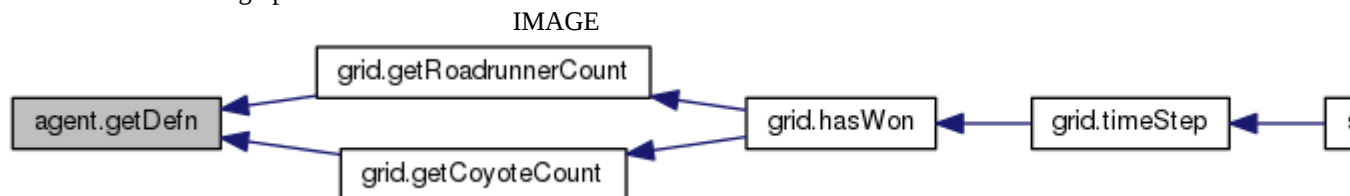
Definition getter.

#### **Returns:**

Integer definition of the agent

Definition at line 40 of file agent.java.

Here is the caller graph for this function:



### **int agent.getI ()**

Gets the index of the agent.

#### **Returns:**

Integer index of the agent

Definition at line 48 of file agent.java.

Here is the caller graph for this function:



### **int agent.getP ()**

Gets the previous location of the road-runner.

Definition at line 84 of file agent.java.

### **void agent.move ()**

Handles the movement of the agent. For the agent class this function does nothing because we consider undefined agents to be spaces in this simulation.\ Function is defined for children.

Definition at line 57 of file agent.java.

Here is the caller graph for this function:



---

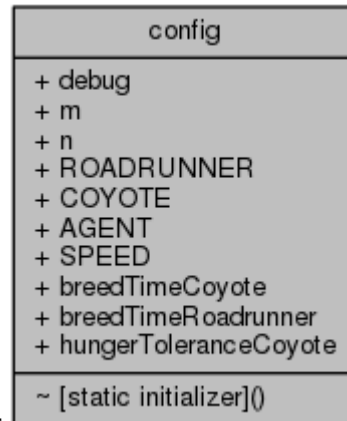
The documentation for this class was generated from the following file:

0 agent.java

## config Class Reference

Configuration class.

Collaboration diagram for config:



IMAGE

### Static Public Attributes

static int **debug**  
static int **m**  
static int **n**  
static int **ROADRUNNER**  
static int **COYOTE**  
static int **AGENT**  
static int **SPEED**  
static int **breedTimeCoyote**  
static int **breedTimeRoadrunner**  
static int **hungerToleranceCoyote**

---

### Detailed Description

Configuration class.

Definition at line 9 of file config.java.

---

### Member Data Documentation

**int config.AGENT[static]**

Definition at line 15 of file config.java.

**int config.breedTimeCoyote[static]**

Definition at line 17 of file config.java.

**int config.breedTimeRoadrunner[static]**

Definition at line 18 of file config.java.

**int config.COYOTE[static]**

Definition at line 14 of file config.java.

**int config.debug[static]**

Definition at line 10 of file config.java.

**int config.hungerToleranceCoyote[static]**

Definition at line 19 of file config.java.

**int config.m[static]**

Definition at line 11 of file config.java.

**int config.n[static]**

Definition at line 12 of file config.java.

**int config.ROADRUNNER[static]**

Definition at line 13 of file config.java.

**int config.SPEED[static]**

Definition at line 16 of file config.java.

---

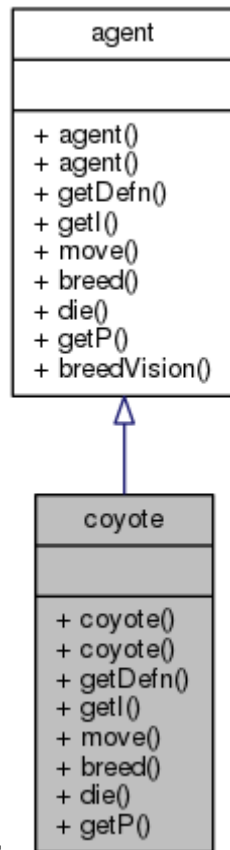
**The documentation for this class was generated from the following file:**

1 config.java

## coyote Class Reference

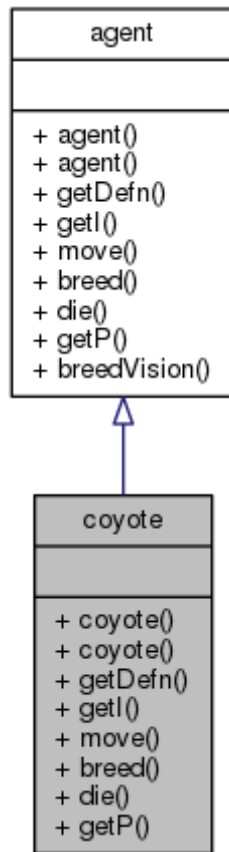
Coyote class.

Inheritance diagram for coyote:



IMAGE

Collaboration diagram for coyote:



IMAGE

### Public Member Functions

**coyote** (int iLoc)

*Coyote constructor with parameters.*

**coyote** ()

*Road-runner constructor without parameters. Reports error if invoked without the location.*

int **getDefn** ()

*Definition getter.*

int **getI** ()

*Gets the index for the coyote.*

void **move** ()

*Function that helps the coyote decide where to move and move there.*

void **breed** ()

*Handles the breeding action of the coyote.*

void **die** ()

*Coyote dies if it hasn't eaten a road-runner in 4 time-steps.*

int **getP** ()

*Gets the previous location of the road-runner.*

---

### Detailed Description

Coyote class.

A template for a coyote. A coyote in this simulation inherits from agent and has a definition of 2.

Definition at line 10 of file coyote.java.

---

## Constructor & Destructor Documentation

### **coyote.coyote (int *iLoc*)**

Coyote constructor with parameters.

#### **Parameters:**

<i>iLoc</i>	The location of the coyote in the grid.
-------------	---

Definition at line 42 of file coyote.java.

### **coyote.coyote ()**

Road-runner constructor without parameters. Reports error if invoked without the location.

Definition at line 50 of file coyote.java.

---

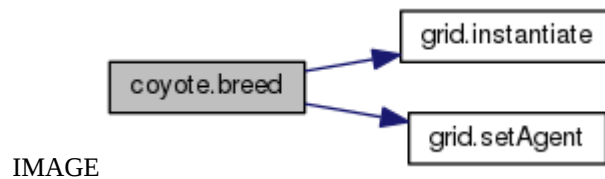
## Member Function Documentation

### **void coyote.breed ()**

Handles the breeding action of the coyote.

Definition at line 218 of file coyote.java.

Here is the call graph for this function:

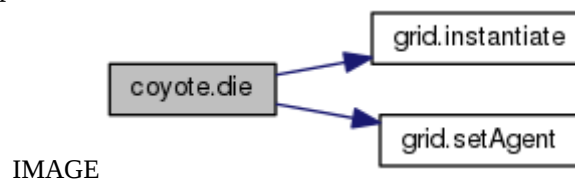


### **void coyote.die ()**

Coyote dies if it hasn't eaten a road-runner in 4 time-steps.

Definition at line 239 of file coyote.java.

Here is the call graph for this function:



### **int coyote.getDefn ()**

Definition getter.



**Returns:**

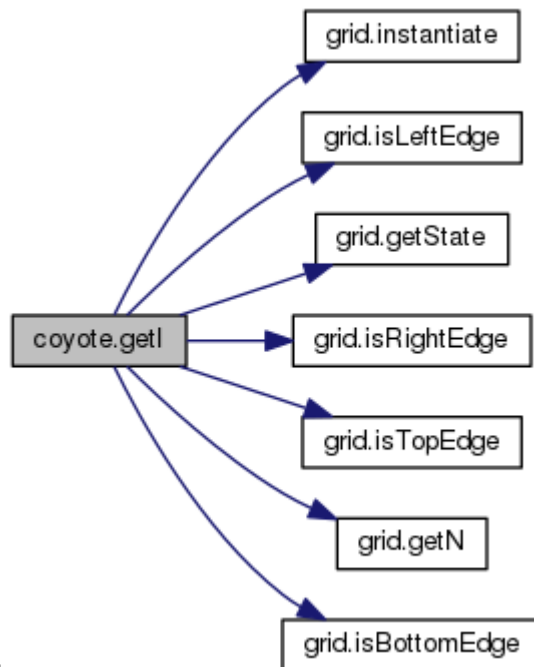
Integer definition of the coyote  
Definition at line 58 of file coyote.java.

**int coyote.getI ()**

Gets the index for the coyote.

**Returns:**

The index of the coyote  
Definition at line 66 of file coyote.java.  
Here is the call graph for this function:



IMAGE

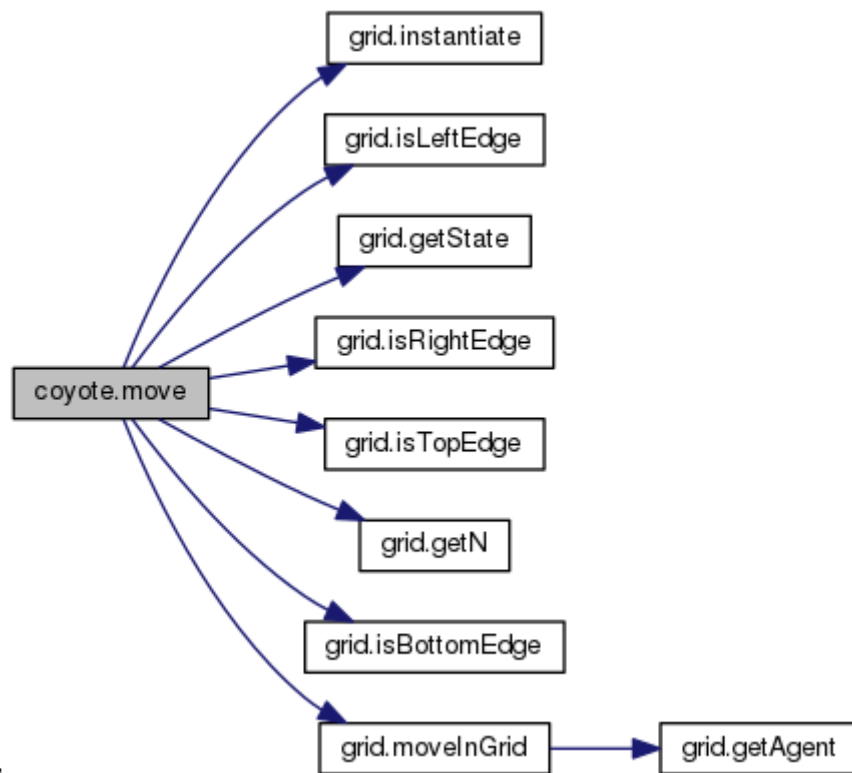
**int coyote.getP ()**

Gets the previous location of the road-runner.  
Definition at line 251 of file coyote.java.

**void coyote.move ()**

Function that helps the coyote decide where to move and move there.

Print statement for debugging purposes  
Print statement for debugging purposes  
Definition at line 131 of file coyote.java.  
Here is the call graph for this function:



IMAGE

---

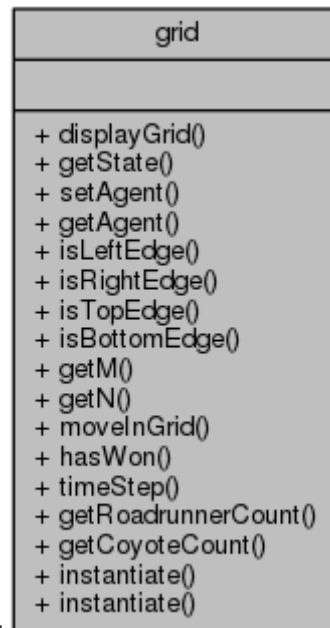
The documentation for this class was generated from the following file:

2 coyote.java

## grid Class Reference

Grid class of type singleton.

Collaboration diagram for grid:



IMAGE

### Public Member Functions

void **displayGrid** ()

*Displays the grid on-screen.*

int **getState** (int j)

*Returns the state at a certain index. Useful when checking what kind of agent occupies the space.*

void **setAgent** (int j, int defn)

*Sets the state at a certain index.*

agent **getAgent** (int j)

*Returns a reference to the agent at certain index.*

boolean **isLeftEdge** (int j)

*Whether or not the given index is a left edge in the grid.*

boolean **isRightEdge** (int j)

*Whether or not the given index is a right edge in the grid.*

boolean **isTopEdge** (int j)

*Whether or not the given index is a top edge in the grid.*

boolean **isBottomEdge** (int j)

*Whether or not the given index is a bottom edge in the grid.*

int **getM** ()

*Gets the m dimension of the grid.*

int **getN** ()

*Get the n dimension of the grid.*

void **moveInGrid** (int pLoc, int iLoc)

*Updates the grid for new updated indexes of the agents.*

boolean **hasWon** ()

*Checks if someone has won the simulation. A win is considered if all coyote or all road-runner remains in the space.*

void **timeStep** (int speed)

*Take a time step. Iterates through the entire grid and invokes agent actions.*

int **getRoadrunnerCount** ()

*Get number of road-runners.*

int **getCoyoteCount** ()

*Get the number of coyotes.*

### Static Public Member Functions

static **grid instantiate** (int j, int k)

*Instantiates a grid object.*

static **grid instantiate** ()

*Returns error if dimension is not provided as argument or else returns the singleton grid instance already created.*

---

### Detailed Description

Grid class of type singleton.

A template for the grid.

Definition at line 10 of file grid.java.

---

### Member Function Documentation

void **grid.displayGrid** ()

Displays the grid on-screen.

Definition at line 119 of file grid.java.

Here is the caller graph for this function:



agent **grid.getAgent** (int j)

Returns a reference to the agent at certain index.

#### Parameters:

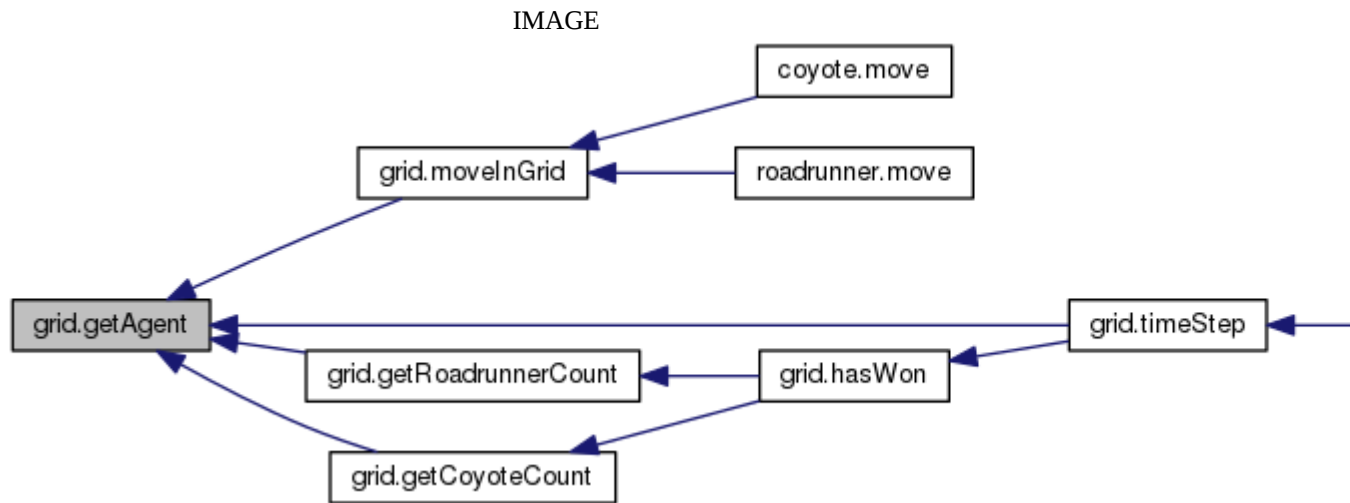
j	The index for which the agent is to be returned
---	---

#### Returns:

The reference to the agent at index j

Definition at line 166 of file grid.java.

Here is the caller graph for this function:



**int grid.getCoyoteCount ()**

Get the number of coyotes.

**Returns:**

The integer of the number of coyotes

Definition at line 344 of file grid.java.

Here is the call graph for this function:

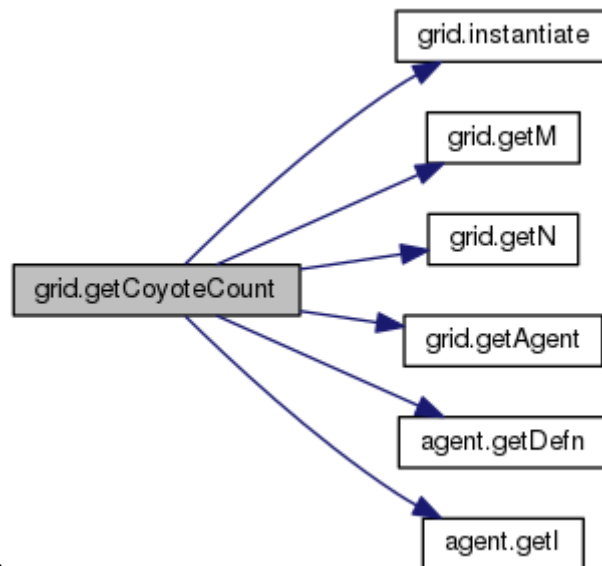


IMAGE  
Here is the caller graph for this function:



**int grid.getM ()**

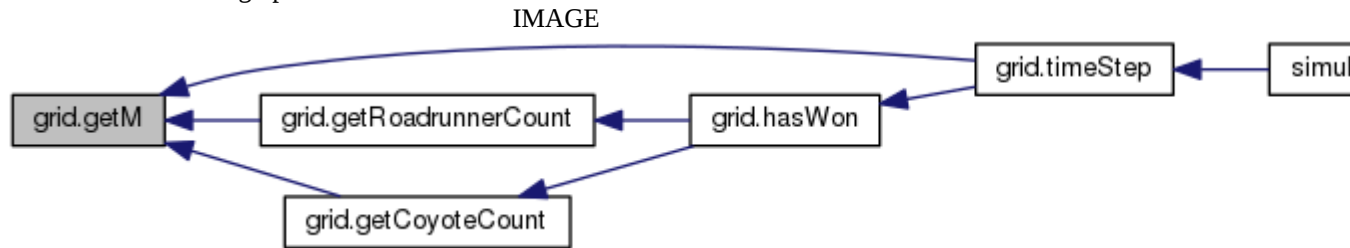
Gets the m dimension of the grid.

**Returns:**

Integer variable representing the dimension

Definition at line 249 of file grid.java.

Here is the caller graph for this function:

**int grid.getN ()**

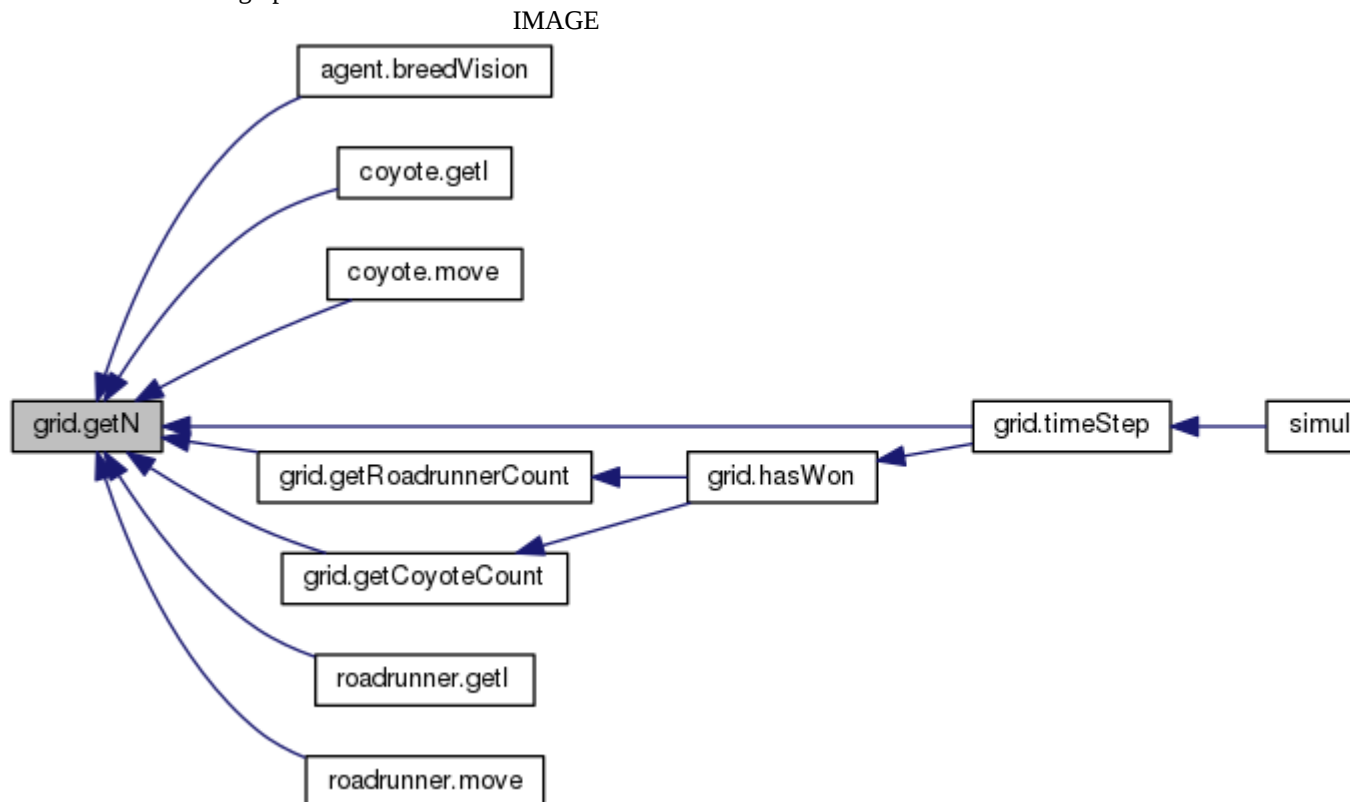
Get the n dimension of the grid.

**Returns:**

Integer variable representing the dimension

Definition at line 257 of file grid.java.

Here is the caller graph for this function:

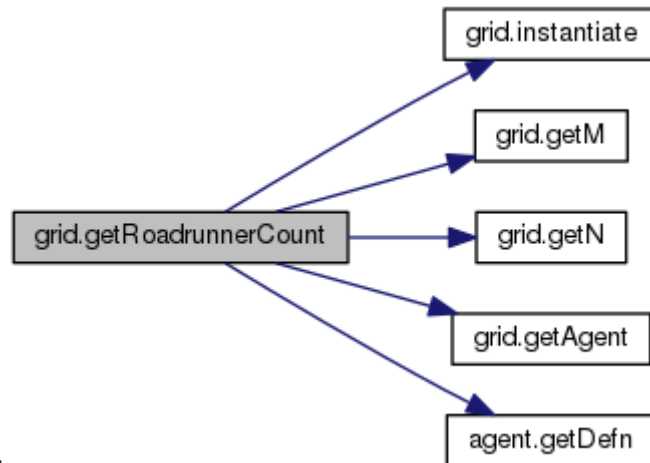
**int grid.getRoadrunnerCount ()**

Get number of road-runners.

**Returns:**

Integer of the number of road-runners

Definition at line 329 of file grid.java.  
Here is the call graph for this function:



IMAGE

Here is the caller graph for this function:

IMAGE



**int grid.getState (int j)**

Returns the state at a certain index. Useful when checking what kind of agent occupies the space.

**Parameters:**

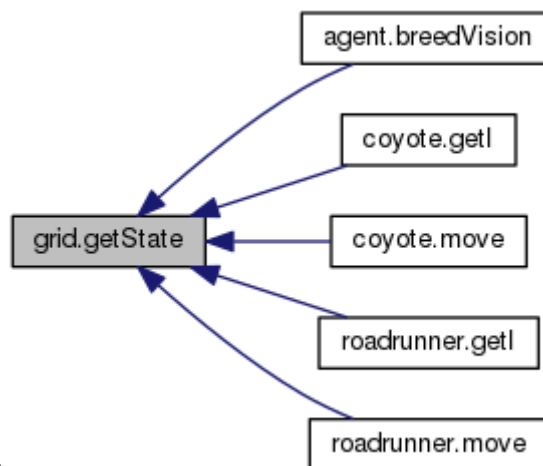
<code>j</code>	The location for which the state is asked for
----------------	---

**Returns:**

The definition of the agent.

Definition at line 136 of file grid.java.

Here is the caller graph for this function:



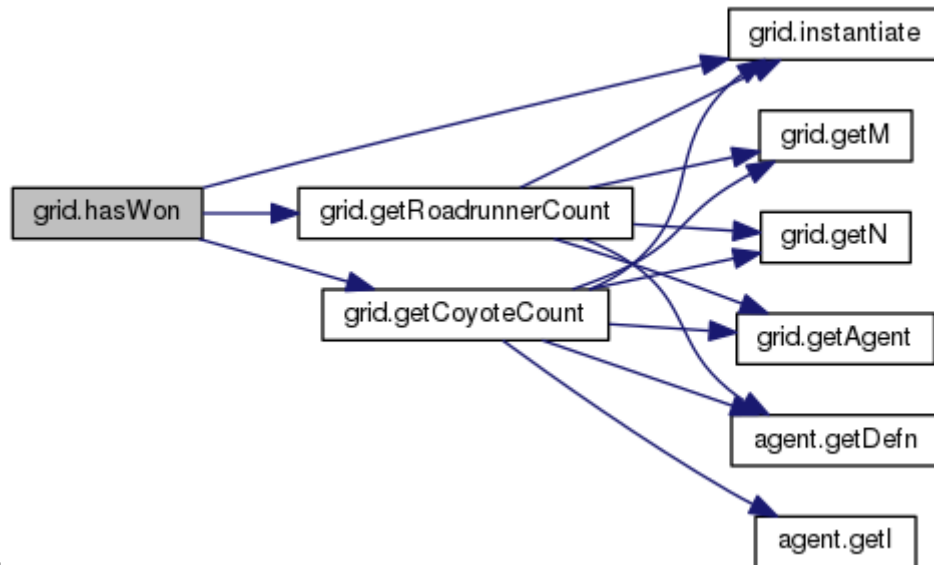
IMAGE

**boolean grid.hasWon ()**

Checks if someone has won the simulation. A win is considered if all coyote or all roadrunner remains in the space.

**Returns:**

Boolean of true or false  
Definition at line 280 of file grid.java.  
Here is the call graph for this function:



IMAGE

Here is the caller graph for this function:

IMAGE



**static grid.grid.instantiate (int j, int k)[static]**

Instantiates a grid object.

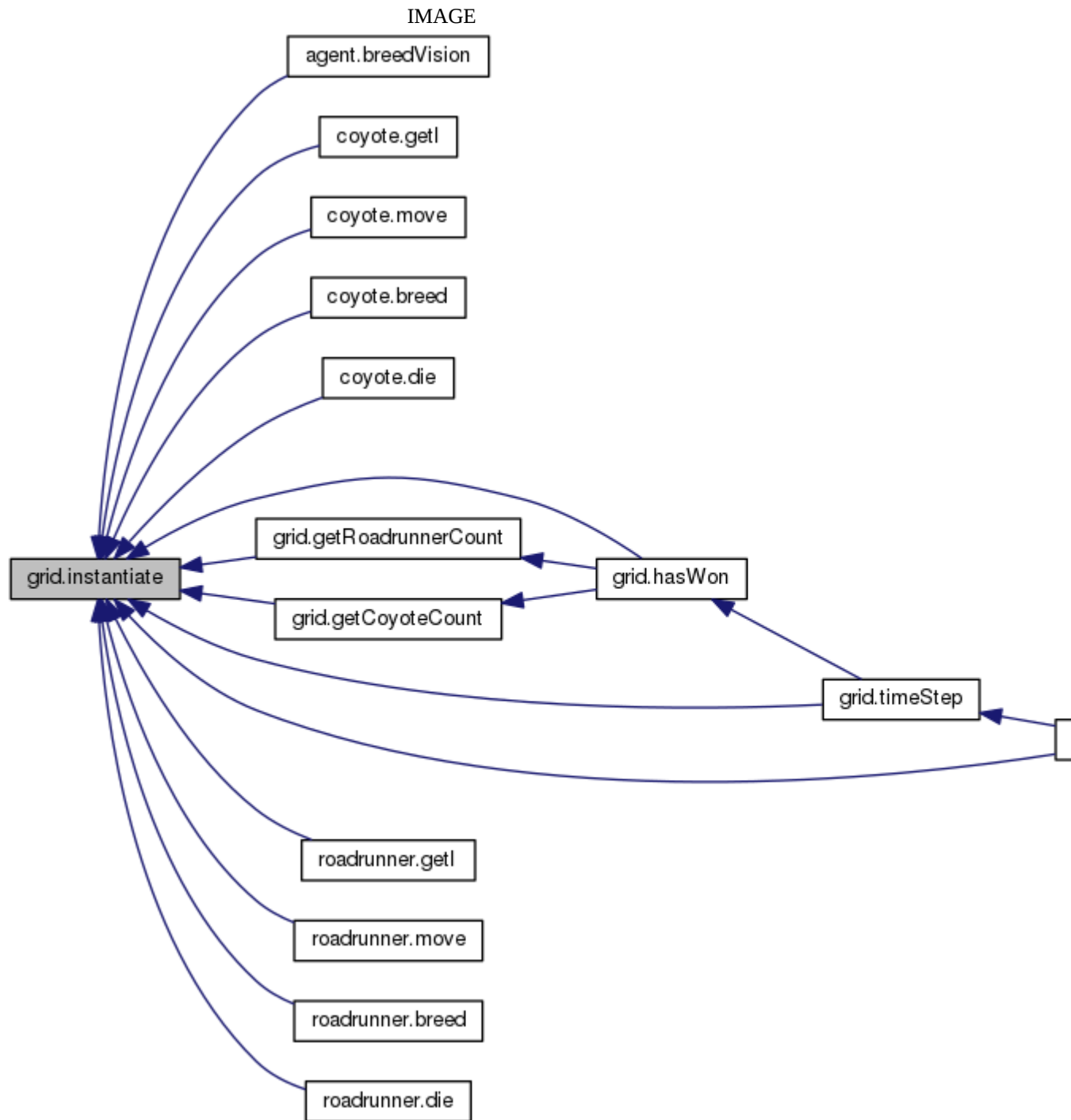
**Parameters:**

<i>j</i>	The n dimension
<i>k</i>	The m dimension

**Returns:**

The singleton grid instance  
Definition at line 49 of file grid.java.  
Here is the caller graph for this function:





**static grid grid.instantiate ()[static]**

Returns error if dimension is not provided as argument or else returns the singleton grid instance already created.

**Returns:**

The singleton grid instance.  
Definition at line 60 of file grid.java.

### boolean grid.isBottomEdge (int *j*)

Whether or not the given index is a bottom edge in the grid.

#### Parameters:

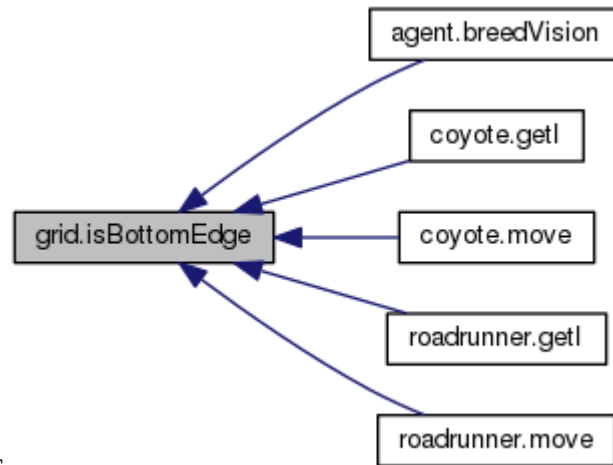
<i>j</i>	The index for which to check
----------	------------------------------

#### Returns:

A boolean value of true or false

Definition at line 235 of file grid.java.

Here is the caller graph for this function:



IMAGE

### boolean grid.isLeftEdge (int *j*)

Whether or not the given index is a left edge in the grid.

#### Parameters:

<i>j</i>	The index for which to check
----------	------------------------------

#### Returns:

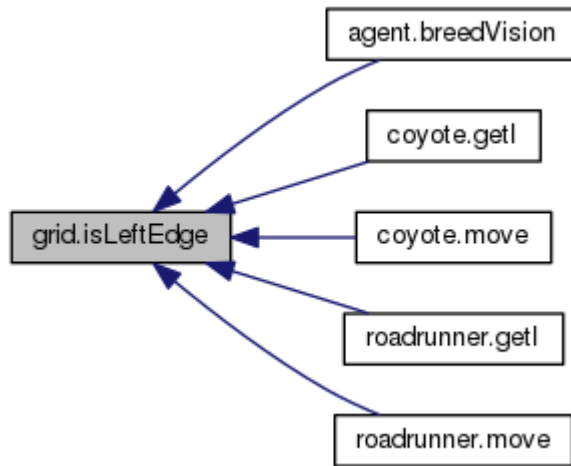
A boolean value of true or false

If the index is 0, it is a left edge

If an out-of-range index is given, this function simply returns false

Definition at line 175 of file grid.java.

Here is the caller graph for this function:



IMAGE

### boolean grid.isRightEdge (int *j*)

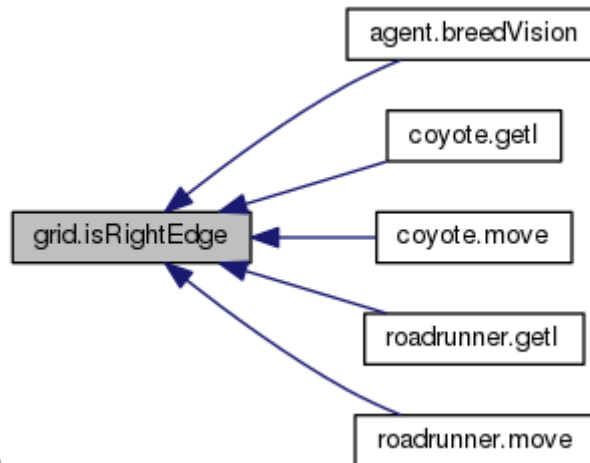
Whether or not the given index is a right edge in the grid.

#### Parameters:

<i>j</i>	The index for which to check
----------	------------------------------

#### Returns:

A boolean value of true or false  
 Definition at line 200 of file grid.java.  
 Here is the caller graph for this function:



IMAGE

### boolean grid.isTopEdge (int *j*)

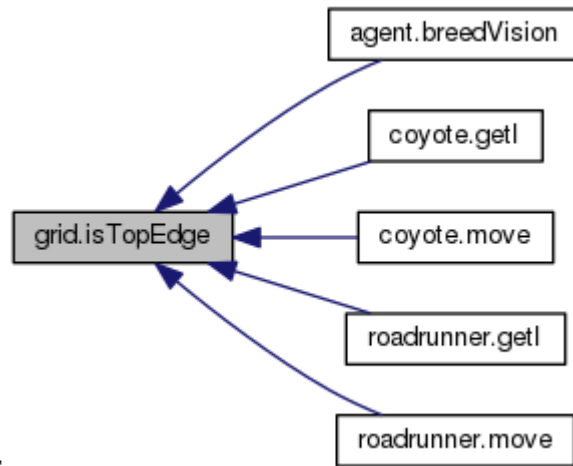
Whether or not the given index is a top edge in the grid.

#### Parameters:

<i>j</i>	The index for which to check
----------	------------------------------

#### Returns:

A boolean value of true or false  
 Definition at line 220 of file grid.java.  
 Here is the caller graph for this function:



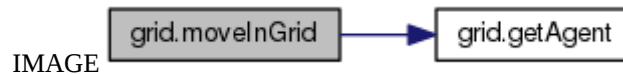
IMAGE

**void grid.moveInGrid (int *pLoc*, int *iLoc*)**

Updates the grid for new updated indexes of the agents.

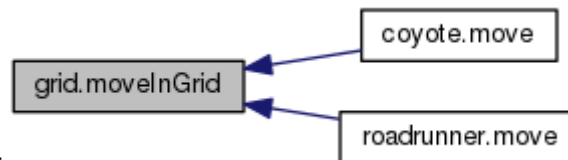
Definition at line 264 of file grid.java.

Here is the call graph for this function:



IMAGE

Here is the caller graph for this function:



IMAGE

**void grid.setAgent (int *j*, int *defn*)**

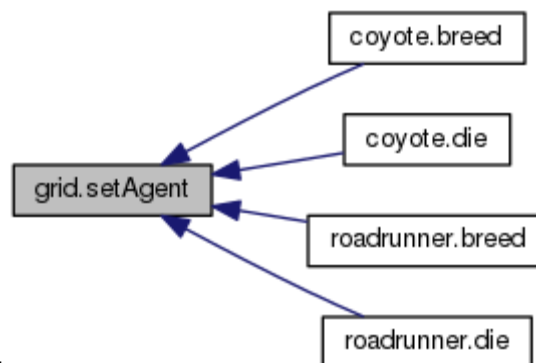
Sets the state at a certain index.

#### Parameters:

<i>j</i>	The location for which the state is to be set
<i>defn</i>	The definition of the agent

Definition at line 145 of file grid.java.

Here is the caller graph for this function:



IMAGE

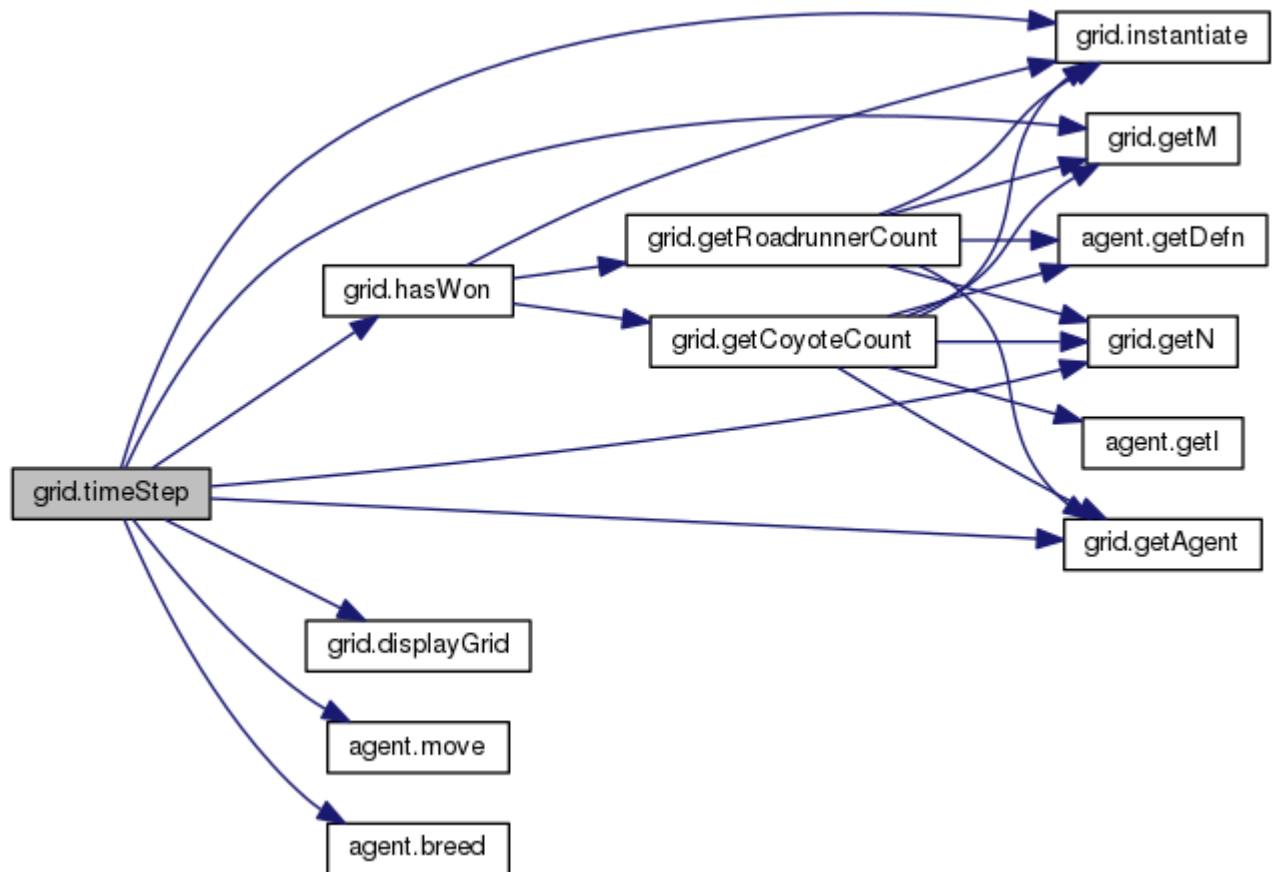
**void grid.timeStep (int *speed*)**

Take a time step. Iterates through the entire grid and invokes agent actions.

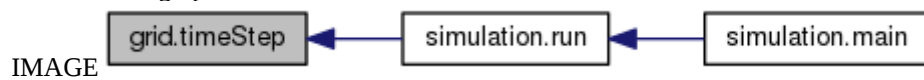
Definition at line 306 of file grid.java.

Here is the call graph for this function:

IMAGE



Here is the caller graph for this function:



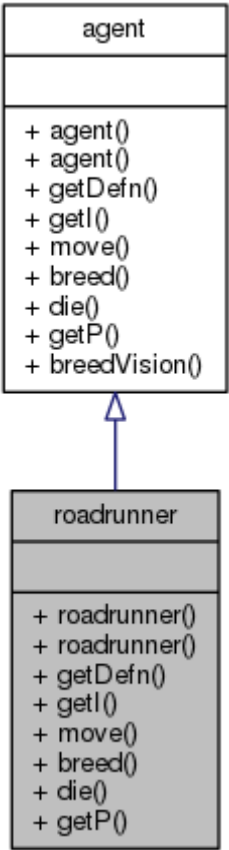
The documentation for this class was generated from the following file:

3 `grid.java`

# roadrunner Class Reference

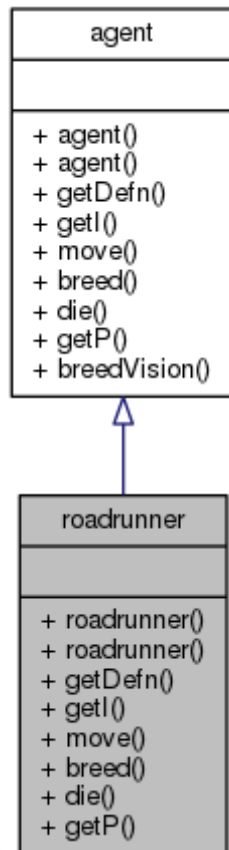
Road-runner class. A template for a road-runner.

Inheritance diagram for roadrunner:



IMAGE

Collaboration diagram for roadrunner:



IMAGE

### Public Member Functions

**roadrunner** (int iLoc)

*Keeps track of the steps a road-runner has taken.*

**roadrunner** ()

*Road-runner constructor without parameters. Reports error if invoked without the location.*

int **getDefn** ()

*Definition getter.*

int **getI** ()

*Gets the index of the agent.*

void **move** ()

*Function that helps the road-runner make a decision on where to move and move there.*

void **breed** ()

*Handles the breeding action of the road-runner.*

void **die** ()

*Kills a road-runner NOTE: This function is only useful for manual kills.*

int **getP** ()

*Gets the previous location of the road-runner.*

---

### Detailed Description

Road-runner class. A template for a road-runner.

Definition at line 9 of file roadrunner.java.

---

## Constructor & Destructor Documentation

### roadrunner.roadrunner (int *iLoc*)

Keeps track of the steps a road-runner has taken.

Road-runner constructor with parameters

#### Parameters:

<i>iLoc</i>	The location of the road-runner.
-------------	----------------------------------

Definition at line 29 of file roadrunner.java.

### roadrunner.roadrunner ()

Road-runner constructor without parameters. Reports error if invoked without the location.

Definition at line 37 of file roadrunner.java.

---

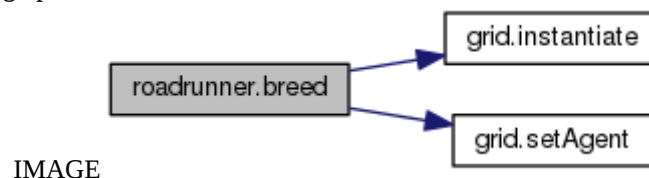
## Member Function Documentation

### void roadrunner.breed ()

Handles the breeding action of the road-runner.

Definition at line 282 of file roadrunner.java.

Here is the call graph for this function:



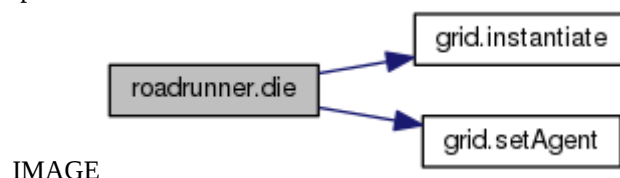
IMAGE

### void roadrunner.die ()

Kills a road-runner NOTE: This function is only useful for manual kills.

Definition at line 299 of file roadrunner.java.

Here is the call graph for this function:



IMAGE

### int roadrunner.getDefn ()

Definition getter.

#### Returns:

Integer definition of the road-runner

Definition at line 45 of file roadrunner.java.



### **int roadrunner.getI ()**

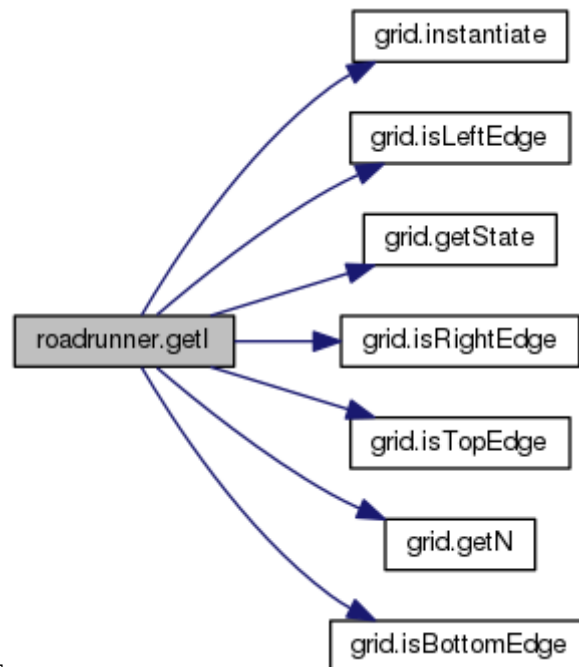
Gets the index of the agent.

#### **Returns:**

Integer index of the agent

Definition at line 53 of file roadrunner.java.

Here is the call graph for this function:



IMAGE

### **int roadrunner.getP ()**

Gets the previous location of the road-runner.

Definition at line 307 of file roadrunner.java.

### **void roadrunner.move ()**

Function that helps the road-runner make a decision on where to move and move there.

#### **Returns:**

void

Make a random one cell movement if no coyote was seen around

If a coyote is seen, decide the best possible location to move to

Left movement is not required for the left edge

Look left to see if a move there is appropriate

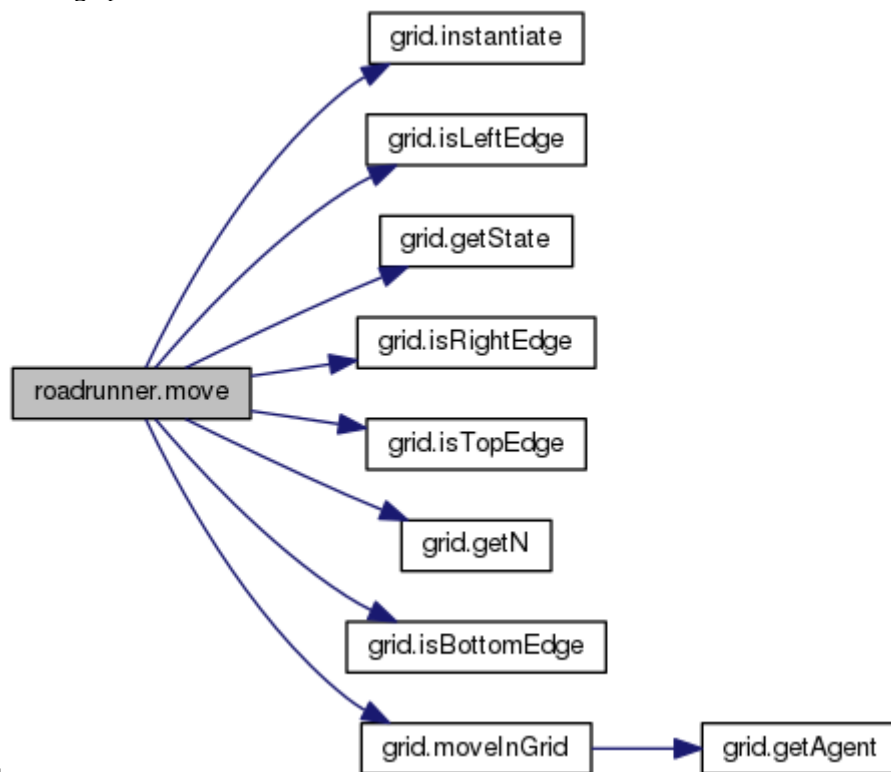
Right movement is not required for the right edge

Look right to see if a move there is more appropriate

Up movement is not required for the top edge

Look up to see if a move there is more appropriate

Up-right movement is not required for the top-right edge  
 Look up-right to see if a move there is more appropriate  
 Up-left movement is not required for the top-left edge  
 Look up-left to see if a move there is more appropriate  
 Down movement is not required for the bottom edge  
 Look down to see if a move there is more appropriate  
 Down-right movement is not required for the bottom-right edge  
 Look down-right to see if a move there is more appropriate  
 Down-left movement is not required for the bottom-left edge  
 Look down-left to see if a move there is more appropriate  
 If an appropriate move is found  
 Update the road-runner's index  
 Update the timeStep for the road-runner  
 Print statement for debugging purposes  
 Definition at line 123 of file roadrunner.java.  
 Here is the call graph for this function:



IMAGE

---

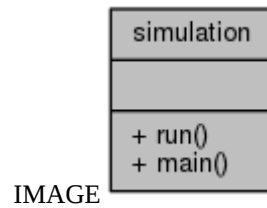
The documentation for this class was generated from the following file:

4 roadrunner.java

## simulation Class Reference

Simulation class. A template for the simulation.

Collaboration diagram for simulation:



### Static Public Member Functions

static void **run** ()

*Instantiates a grid, taking the dimensions from the configuration class and runs a timeStep at certain speed.*

static void **main** (String[] args)

---

### Detailed Description

Simulation class. A template for the simulation.

Definition at line 6 of file simulation.java.

---

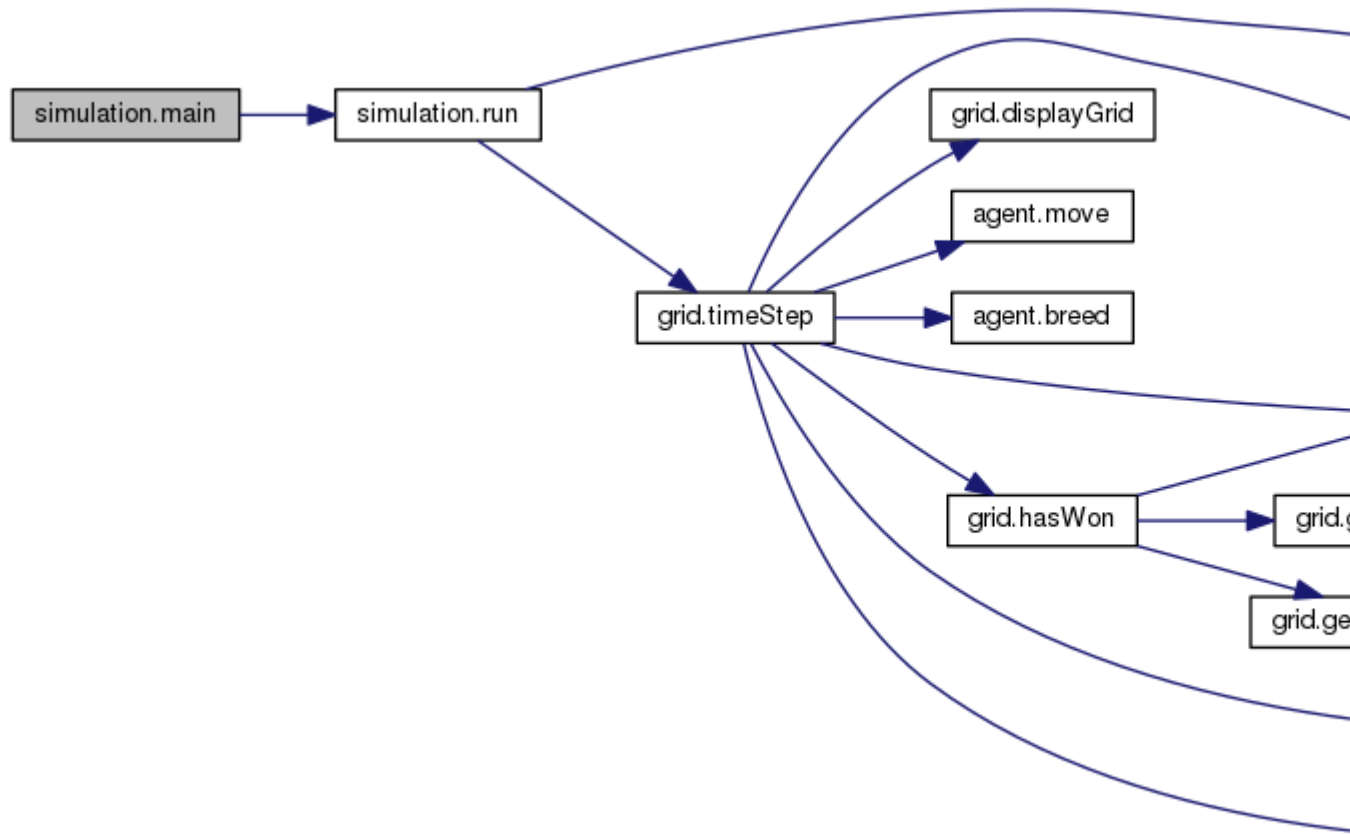
### Member Function Documentation

**static void simulation.main (String [] args)[static]**

Definition at line 15 of file simulation.java.

Here is the call graph for this function:

IMAGE

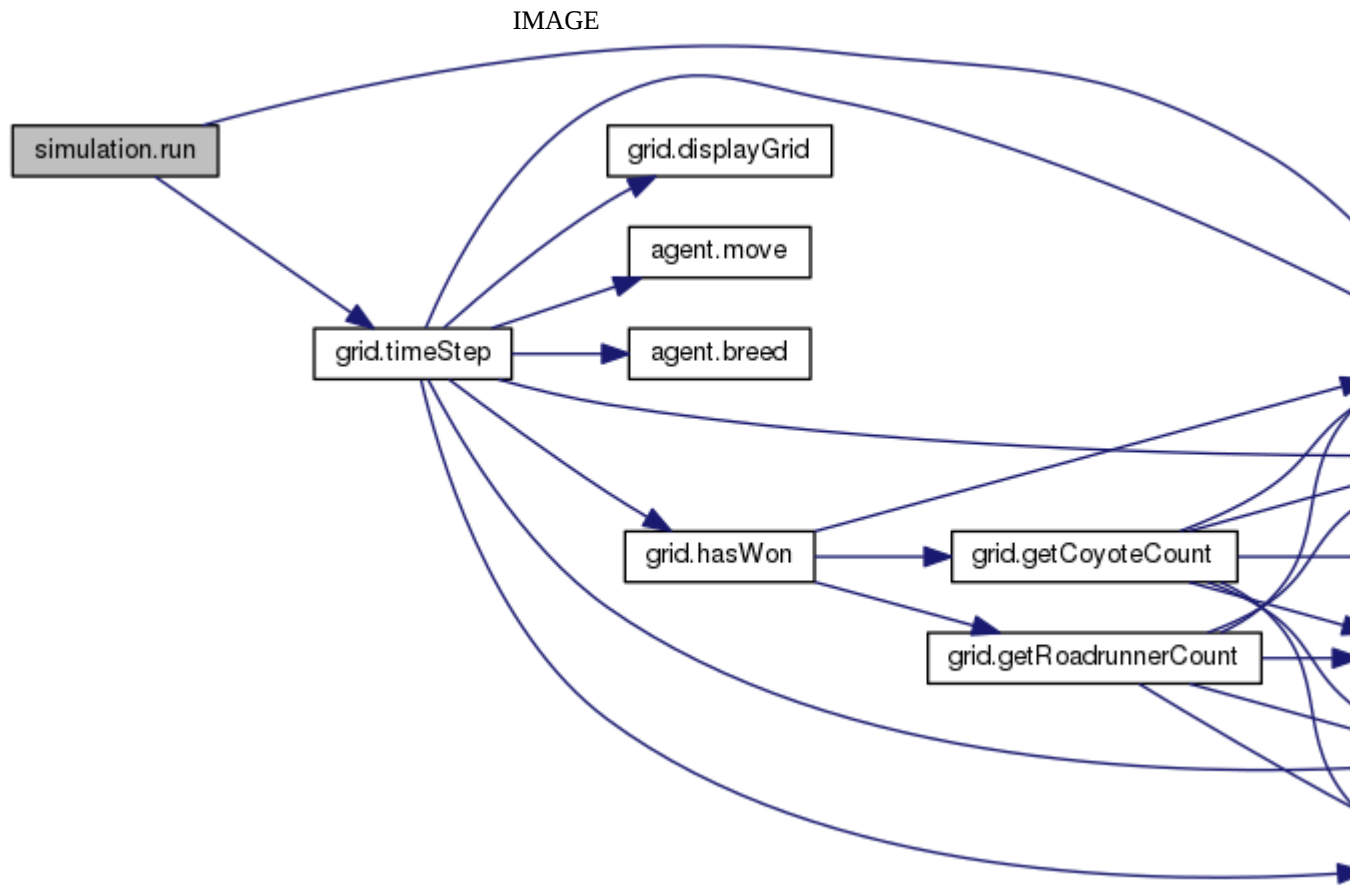


**`static void simulation.run ()[static]`**

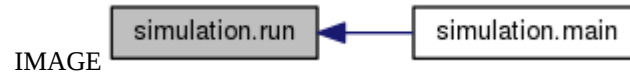
Instantiates a grid, taking the dimensions from the configuration class and runs a timeStep at certain speed.

Definition at line 10 of file simulation.java.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

5 simulation.java

# File Documentation

## agent.java File Reference

### Classes

class **agent**

*Agent class.*

## config.java File Reference

### Classes

class **config**

*Configuration class.*

## coyote.java File Reference

### Classes

class **coyote**

*Coyote class.*



## grid.java File Reference

### Classes

class **grid**

*Grid class of type singleton.*

## roadrunner.java File Reference

### Classes

class **roadrunner**

*Road-runner class. A template for a road-runner.*

## **simulation.java File Reference**

### **Classes**

class **simulation**

*Simulation class. A template for the simulation.* **Index**

INDE