|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Model.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | |  |
| [**PREV CLASS**](http://docs.google.com/fishPond/IllegalPondSizeException.html)   [**NEXT CLASS**](http://docs.google.com/fishPond/Plant.html) | [**FRAMES**](http://docs.google.com/index.html?fishPond/Model.html)    [**NO FRAMES**](http://docs.google.com/Model.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

## **fishPond**

Class Model

java.lang.Object  
  
 **fishPond.Model**

public class **Model**extends java.lang.Object

Model for the Fish Pond Simulation. The model consists of a List of Fish, a List of Plants, and a two dimensional array of boolean values representing the pond (each element in the array is either ROCK, or WATER.)

Each time the simulation is re-started a new Model object is created.

STUDENTS MAY NOT ADD ANY FIELDS. ALSO, STUDENTS MAY NOT ADD ANY PUBLIC METHODS. (PRIVATE METHODS OF YOUR OWN ARE OKAY.)

**Author:** Fawzi Emad, Put Your Name Here

|  |  |
| --- | --- |
| **Field Summary** | |
| static boolean | [**ROCK**](http://docs.google.com/fishPond/Model.html#ROCK)            Value stored in landscape array to represent rock |
| static boolean | [**WATER**](http://docs.google.com/fishPond/Model.html#WATER)            Value stored in landscape array to represent water |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Model**](http://docs.google.com/fishPond/Model.html#Model(int,%20int,%20int,%20int,%20int))(int numRows, int numCols, int numRocks, int numFish, int numPlants)            THIS METHOD HAS BEEN WRITTEN FOR YOU! |
| [**Model**](http://docs.google.com/fishPond/Model.html#Model(fishPond.Model))([Model](http://docs.google.com/fishPond/Model.html) other)            Copy Constructor. |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**addFish**](http://docs.google.com/fishPond/Model.html#addFish(fishPond.Fish))([Fish](http://docs.google.com/fishPond/Fish.html) f)            Adds the fish f to the fish list, if possible. |
| void | [**addPlant**](http://docs.google.com/fishPond/Model.html#addPlant(fishPond.Plant))([Plant](http://docs.google.com/fishPond/Plant.html) p)            Attempts to add the plant p to plant list, if possible. |
| static void | [**fishEatPlant**](http://docs.google.com/fishPond/Model.html#fishEatPlant(fishPond.Fish,%20fishPond.Plant))([Fish](http://docs.google.com/fishPond/Fish.html) f, [Plant](http://docs.google.com/fishPond/Plant.html) p)            Fish f eats a portion of plant p. |
| void | [**fishExplosions**](http://docs.google.com/fishPond/Model.html#fishExplosions())()            THIS METHOD HAS BEEN WRITTEN FOR YOU! |
| int | [**getCols**](http://docs.google.com/fishPond/Model.html#getCols())()            returns number of columns in landscape array |
| java.util.ArrayList<[Fish](http://docs.google.com/fishPond/Fish.html)> | [**getFish**](http://docs.google.com/fishPond/Model.html#getFish())()            Returns a COPY of the fish list. |
| java.util.ArrayList<[Plant](http://docs.google.com/fishPond/Plant.html)> | [**getPlants**](http://docs.google.com/fishPond/Model.html#getPlants())()            Returns a COPY of the plants list. |
| int | [**getRows**](http://docs.google.com/fishPond/Model.html#getRows())()            returns number of rows in landscape array |
| boolean | [**getShape**](http://docs.google.com/fishPond/Model.html#getShape(int,%20int))(int row, int col)            Returns the specified entry of the landscape array (either WATER or ROCK). |
| void | [**growPlants**](http://docs.google.com/fishPond/Model.html#growPlants())()            Iterates through the plants list, growing each plant by invoking it's "grow" method. |
| boolean | [**isSpaceAvailable**](http://docs.google.com/fishPond/Model.html#isSpaceAvailable(int,%20int))(int r, int c) |
| void | [**moveFish**](http://docs.google.com/fishPond/Model.html#moveFish())()            Note: This method assumes that each live fish that is not surrounded by rocks is already facing a direction where there is no rock! (Typically the call to this method should immediately follow a call to "turnFish", which ensures that these conditions are satisfied.) |
| void | [**plantExplosions**](http://docs.google.com/fishPond/Model.html#plantExplosions())()            THIS METHOD HAS BEEN WRITTEN FOR YOU. |
| void | [**removeDeadFish**](http://docs.google.com/fishPond/Model.html#removeDeadFish())()            Iterates through the list of Fish. |
| void | [**removeDeadPlants**](http://docs.google.com/fishPond/Model.html#removeDeadPlants())()            Iterates through the list of Plants. |
| void | [**shrinkFish**](http://docs.google.com/fishPond/Model.html#shrinkFish())()            Iterates through fish list. |
| void | [**turnFish**](http://docs.google.com/fishPond/Model.html#turnFish())()            Iterate through list of Fish. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Field Detail** |

### WATER

public static final boolean **WATER**

Value stored in landscape array to represent water

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#fishPond.Model.WATER)

### ROCK

public static final boolean **ROCK**

Value stored in landscape array to represent rock

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#fishPond.Model.ROCK)

|  |
| --- |
| **Constructor Detail** |

### Model

public **Model**(int numRows,  
  
 int numCols,  
  
 int numRocks,  
  
 int numFish,  
  
 int numPlants)

THIS METHOD HAS BEEN WRITTEN FOR YOU!

If numRows is smaller than MIN\_POND\_ROWS, or if numCols is smaller than MIN\_POND\_COLS, then this method will throw an IllegalPondSizeException.

The fields "rows" and "cols" are initilized with the values of parameters numRows and numCols.

The field "landscape" is initialized as a 2-dimensional array of booleans. The size is determined by rows and cols. Every entry in the landscape array is filled with WATER. The border around the perimeter of the landscape array (top, bottom, left, right) is then overwritten with ROCK.

Random rocks are placed in the pond until the number of rocks (in addition to those in the border) reaches numRocks.

The "plants" ArrayList is instantiated. Randomly placed Plant objects are put into the List. Their positions are chosen so that they are never above rocks or in the same position as another plant. Plants are generated in this way until the list reaches size numPlants.

The "fish" ArrayList is instantiated. Now randomly placed Fish objects are put into the List. Their directions are also randomly selected. The positions are chosen so that they are never above rocks, plants, or other fish. Fish are generated in this way until the list reaches size numFish.

**Parameters:**numRows - number of rows for pondnumCols - number of columns for pondnumRocks - number of rocks to be drawn in addition to rocks around border of pondnumFish - number of fish to start withnumPlants - number of plants to start with

### Model

public **Model**([Model](http://docs.google.com/fishPond/Model.html) other)

Copy Constructor.

Since landscape is immutable, it is be copied with just a reference copy. Fish and Plants are mutable, so they must be copied with a DEEP copy! (WARNING: Each fish and each plant must be copied.)

|  |
| --- |
| **Method Detail** |

### plantExplosions

public void **plantExplosions**()

THIS METHOD HAS BEEN WRITTEN FOR YOU.

When a plant gets bigger than Plant.MAX\_PLANT\_SIZE, it will explode into 2 to 9 smaller plants, whose sizes add up to the size of the original plant. The smaller plants will be placed in the 9 regions of the landscape array that surround the original plant. If there are rocks, fish, or other plants already occupying these adjacent regions, then fewer than 9 plants are created. If there are no available regions nearby, the plant will not explode.

### fishExplosions

public void **fishExplosions**()

THIS METHOD HAS BEEN WRITTEN FOR YOU!

When a fish gets bigger than Fish.MAX\_FISH\_SIZE, it will explode into 4 to 8 smaller fish, whose sizes add up to the size of the original fish. The smaller fish will be placed in the eight regions of the landscape array surrounding the original fish. The little fish will be begin moving in directions that point away from the original location. (Note that no little fish is placed into the original location of the landscape array where the exploding fish was -- just in the surrounding squares.) If there are rocks, fish, or plants already occupying these adjacent squares, then fewer than eight little fish are created. If there are not at least four available surrounding squares, then the fish will not explode.

### isSpaceAvailable

public boolean **isSpaceAvailable**(int r,  
  
 int c)

### fishEatPlant

public static void **fishEatPlant**([Fish](http://docs.google.com/fishPond/Fish.html) f,  
  
 [Plant](http://docs.google.com/fishPond/Plant.html) p)

Fish f eats a portion of plant p. The amount eaten is either Plant.PLANT\_BITE\_SIZE or the current size of the plant, whichever is smaller. The fish's size is increased by this amount and the plant's size is decreased by this amount.

### getRows

public int **getRows**()

returns number of rows in landscape array

### getCols

public int **getCols**()

returns number of columns in landscape array

### shrinkFish

public void **shrinkFish**()

Iterates through fish list. For each fish that isAlive, shrinks the fish by invoking it's "shrink" method.

### growPlants

public void **growPlants**()

Iterates through the plants list, growing each plant by invoking it's "grow" method.

### removeDeadFish

public void **removeDeadFish**()

Iterates through the list of Fish. Any fish that is no longer alive is removed from the list.

### removeDeadPlants

public void **removeDeadPlants**()

Iterates through the list of Plants. Any plant that is no longer alive is removed from the list.

### turnFish

public void **turnFish**()

Iterate through list of Fish. For each fish that isAlive, do the following:

1. If this fishIsSurroundedByRocks, DO NOTHING, and move on to the next fish. (This fish will not turn.)

2. If this fish's direction is not equal to one of the codes UP, DOWN, LEFT, or RIGHT, then throw an IllegalFishDirectionException, passing this fish's direction to the constructor.

3. Check whether or not this fish is about to hit a rock if it moves in it's current direction. If it is about to hit a rock, call the fish's setRandomDirection method. Repeat this step until the fish is no longer about to hit a rock. Do not make any EXTRA calls to setRandomDirection or you will fail our tests!

### moveFish

public void **moveFish**()

Note: This method assumes that each live fish that is not surrounded by rocks is already facing a direction where there is no rock! (Typically the call to this method should immediately follow a call to "turnFish", which ensures that these conditions are satisfied.)

This method iterates through the list of fish. For each fish that isAlive do the following:

1. Check to see if this fishIsSurroundedByRocks. If so, DO NOTHING and move along to the next fish in the list. (This fish does not move, does not eat, does not fight.)

2. Move this fish by calling it's "move" method.

3. Check if there is a plant that isAlive and is located in the same position as this fish. If so, have the fish eat part of the plant by calling fishEatPlant.

4. Check if there is another fish (distinct from this fish) that is in the same location as this fish. If so, have the two fish fight each other by calling the fight method. IMPORTANT -- the fight method is not symmetrical. You must use THIS fish as the current object, and pass the OTHER fish as the parameter (otherwise you will not pass our tests.)

### addPlant

public void **addPlant**([Plant](http://docs.google.com/fishPond/Plant.html) p)

Attempts to add the plant p to plant list, if possible.

First checks if the landscape in the plant's location is equal to ROCK. If it is, then does not add the plant to the list. Instead throws an IllegalPlantPositionException, passing IllegalPlantPositionException.PLANT\_OVER\_ROCK to the constructor.

Now checks for another plant (distinct from the parameter) that is in the same location as the parameter. If one is found, then does not add the plant to the list. Instead throws an IllegalPlantPositionException, passing IllegalPlantPositionException.TWO\_PLANTS\_IN\_ONE\_PLACE to the constructor.

Otherwise, adds the plant to the list "plants".

### addFish

public void **addFish**([Fish](http://docs.google.com/fishPond/Fish.html) f)

Adds the fish f to the fish list, if possible.

First checks if the landscape in the fish's location is equal to ROCK. If it is, then the fish is not added to the list. Instead, throws an IllegalFishPositionException, passing IllegalFishPositionException.FISH\_OVER\_ROCK to the constructor.

Next checks for another fish (distinct from the parameter) that is in the same location as the parameter. If one is found, then the fish is not added to the list. Instead throws an IllegalFishPositionException, passing IllegalFishPositionException.TWO\_FISH\_IN\_ONE\_PLACE to the constructor.

Otherwise, adds the parameter to the fish list.

### getFish

public java.util.ArrayList<[Fish](http://docs.google.com/fishPond/Fish.html)> **getFish**()

Returns a COPY of the fish list. Hint: Use the ArrayList copy constructor, or you will fail our tests!

### getPlants

public java.util.ArrayList<[Plant](http://docs.google.com/fishPond/Plant.html)> **getPlants**()

Returns a COPY of the plants list. Hint: Use the ArrayList copy constructor, or you will fail our tests!

### getShape

public boolean **getShape**(int row,  
  
 int col)

Returns the specified entry of the landscape array (either WATER or ROCK).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Model.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | |  |
| [**PREV CLASS**](http://docs.google.com/fishPond/IllegalPondSizeException.html)   [**NEXT CLASS**](http://docs.google.com/fishPond/Plant.html) | [**FRAMES**](http://docs.google.com/index.html?fishPond/Model.html)    [**NO FRAMES**](http://docs.google.com/Model.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

[Web Accessibility](https://www.umd.edu/web-accessibility)