

OOP Hand-in Project Assignment



Student: Marc Giger

Lecturer: John Walsh / John Brosnan

Date: 1 December 2019

# Contents

[Contents II](#_Toc26105066)

[OOP Project: Requirements Specification 3](#_Toc26105067)

[UML / VOPC Extract 1 4](#_Toc26105068)

[UML Extract 2 5](#_Toc26105069)

[GitHub link 6](#_Toc26105070)

[GitHub logs / Commit history 6](#_Toc26105071)

[Part extract of the JavaDoc output 7](#_Toc26105072)

[JavaDoc extract: Class Shark 7](#_Toc26105073)

[JavaDoc extract: Class Model 12](#_Toc26105083)

[Declaration of Originality 17](#_Toc26105090)

# OOP Project: Requirements Specification

**Visualisation of Fish- and Shark-Population (on a very basic view)**

The Visualisation Program should help for academic research. The user gains insight of the reciprocity of shark and fish stock in the Pacific Ocean.

* Sharks feed from fish, hence without fish there are no sharks.
* Without a predator the fish stock would increase quickly.

These are simple understanding statements, which can be easily visualised. The aim of this project is to show how the visualisation can be done with the JAVA-IDE: IntelliJ IDEA.

A 9 x 16 grid is set up for the visualisation, which is the main part of the GUI. Furthermore, three instantiable classes are written (Water, Fish, Shark) to be placed within the Grid. Add some realistic attributes and methods and your program is almost done. To fulfil the project specification the following features are implemented as well: Persistence (Serializable), Inheritance, Nested Objects (Composition/Aggregation), GUI and background algorithms (See UML / VOPC Diagram and Program Code for a better understanding).

Thinking further, the program right now is on a very basic stage. There are more factors which have a critical impact on the animal stock such as mankind, sea temperature or animal habits in general. We could add all those factors till we have a perfect simulation of the real world within this program. So, what we have laying in front of us, is an infinite expandable simulation, if our imagination and our knowledge is capable to.

**Features implemented so far:**

* Fish, shark and the sea displayed on a grid
* User can reset grid if he is not happy with it
* User can set the animal moving by pressing button Simulation
  + User can decide up to 2000 Simulation in a loop
  + User is forced to enter a value of 1 to 2000
* User can Save and Load current stats through the implemented menu

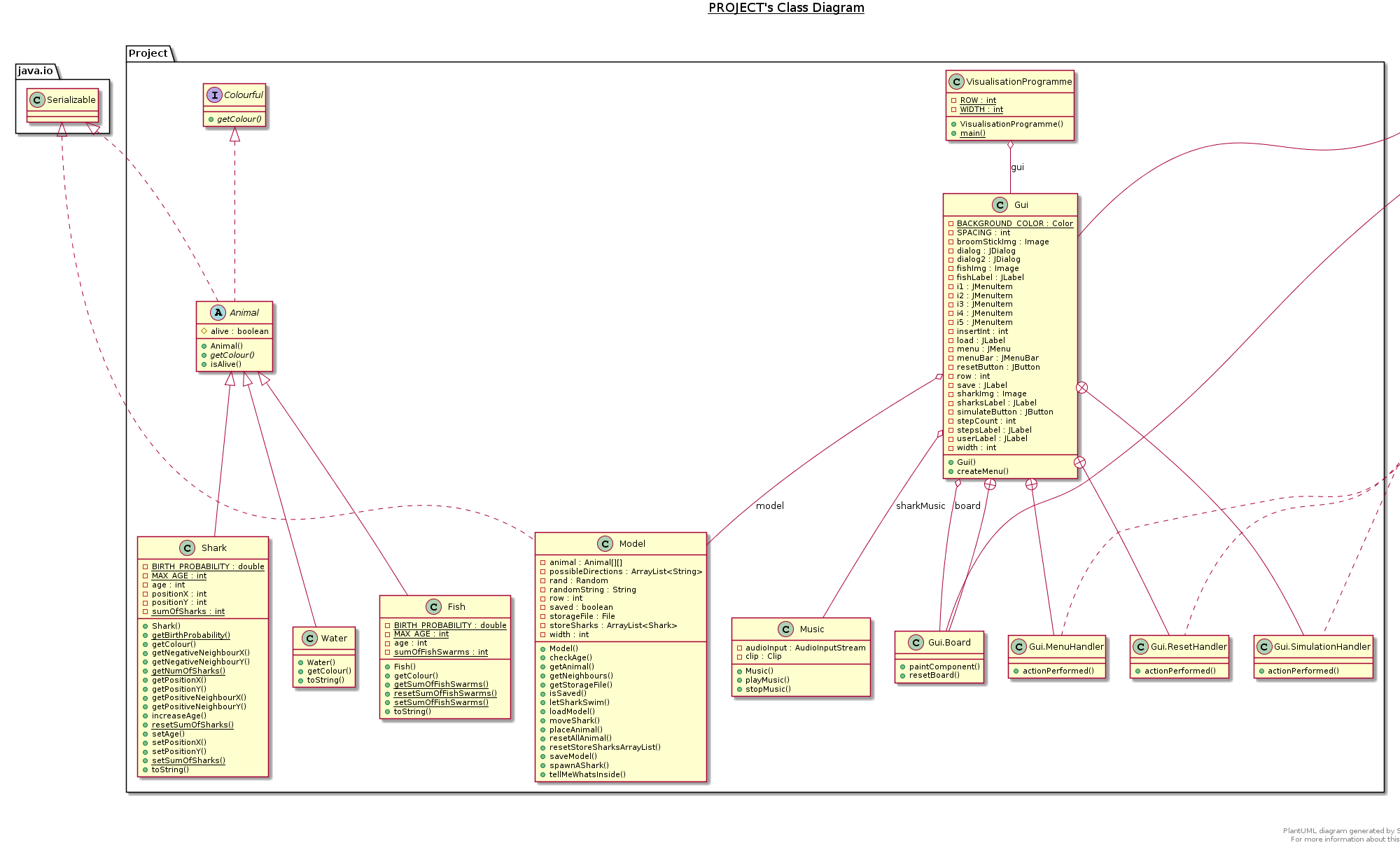
**Upgrades:**

* Fish and shark are displayed by an icon (favoured feature by user)
* To make the simulation more dramatic: White Shark music is playing (comments of testers: “Authentic!”)

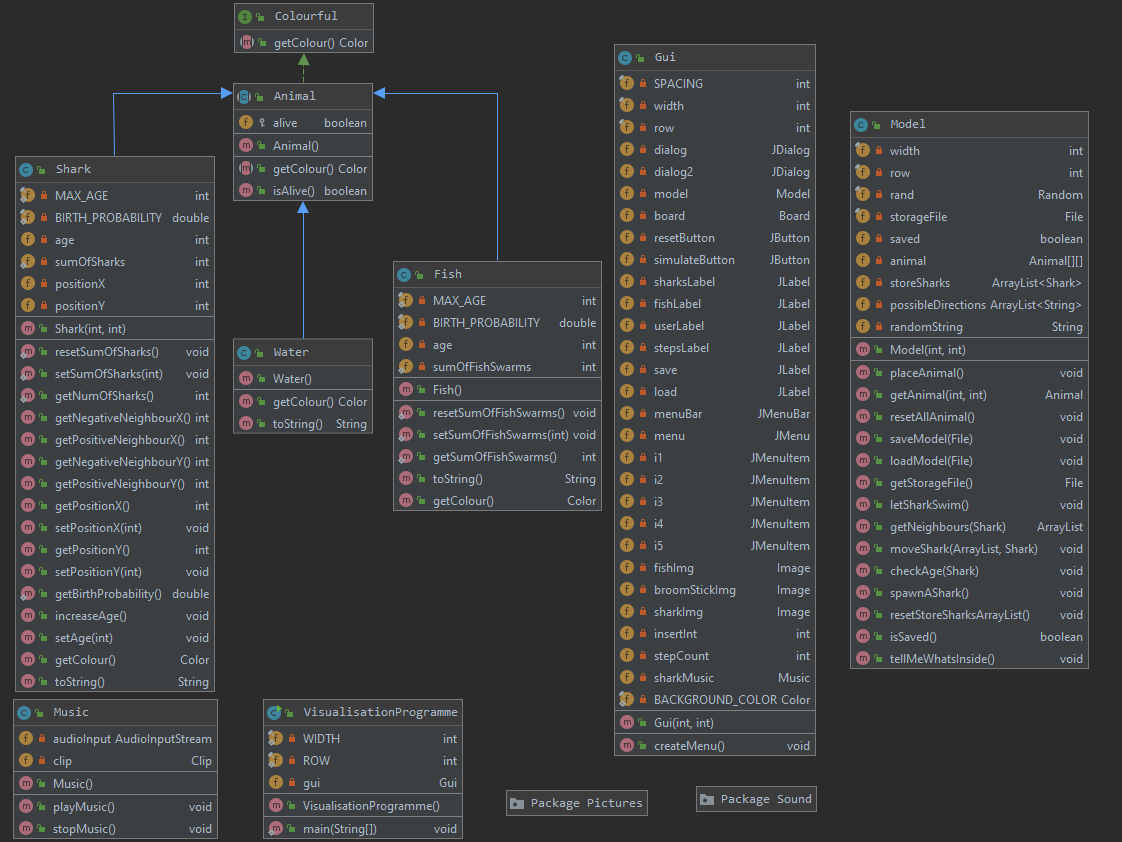
**Not implemented Features:**

* (Fish swim and spawn) Could have been made easy by reusing code, but how the program is set up so far it is not efficient). If l would start from the beginning, l would add a Position class. On the other hand, I was able to invest more time in discovering other features that I added to the visualisation program.
* (Sharks eat fish). Similar reason from above why not implemented yet.

# UML / VOPC Extract 1

The UML / VOPC Extract 1 is done with sketch it (reverse engineering plugin) and <https://www.planttext.com/> to visualise. 

# UML Extract 2

The UML Extract 2 is done with the imbedded function of IntelliJ.

# GitHub link

<https://github.com/MarcGiger/com.marcsoftware.project.git>

# GitHub logs / Commit history

In total 97 commits from the 4th of November till 29th of November. Accessible via

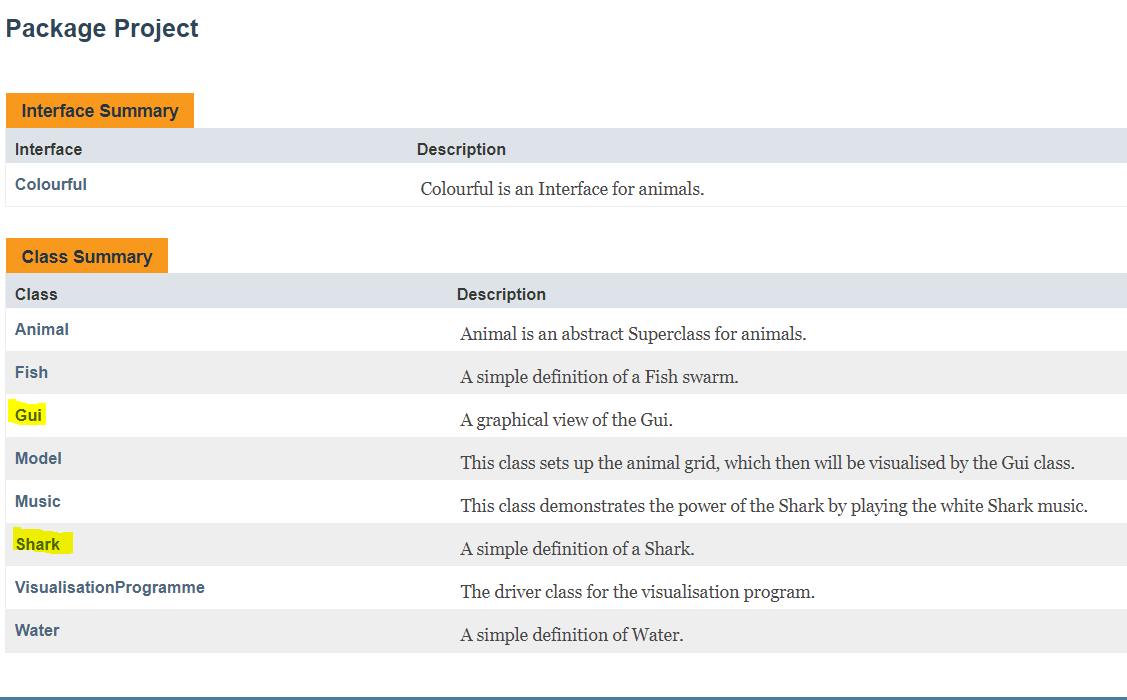
<https://github.com/MarcGiger/com.marcsoftware.project/commits/master>

or via



# Part extract of the JavaDoc output

All classes are commented. Below your will find the extract of the Shark and the Model Class.



# JavaDoc extract: Class Shark

**Package** [Project](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\package-summary.html)

## Class Shark

* java.lang.Object
  + [Project.Animal](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html)
    - Project.Shark

**All Implemented Interfaces:**

java.io.Serializable, [Colourful](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Colourful.html)

public class **Shark**

extends [Animal](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html)

A simple definition of a Shark. A Shark can die, give birth.

**Version:**

0.1

**Author:**

Marc Giger

**See Also:**

[Serialized Form](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\serialized-form.html#Project.Shark)

### *Field Summary*

### Fields inherited from class Project.[Animal](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html)

[alive](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html#alive)

### *Constructor Summary*

|  |  |
| --- | --- |
| **Constructors** | |
| **Constructor** | **Description** |
| [**Shark**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#%3Cinit%3E(int,int))​(int positionX, int positionY) | Create a Shark. |

### *Method Summary*

All MethodsStatic MethodsInstance MethodsConcrete Methods

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Description** |
| static double | [**getBirthProbability**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#getBirthProbability())() | Get the birth probability. |
| java.awt.Color | [**getColour**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#getColour())() | Get the colour shown on the grid for Animal-Type Shark. |
| int | [**getNegativeNeighbourX**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#getNegativeNeighbourX())() | Get the value of the field on the left if it exists. |
| int | [**getNegativeNeighbourY**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#getNegativeNeighbourY())() | Get the value of the field above if it exists. |
| static int | [**getNumOfSharks**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#getNumOfSharks())() | Get the amount of created Sharks. |
| int | [**getPositionX**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#getPositionX())() | Get the position on which the shark is placed on the x-axis. |
| int | [**getPositionY**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#getPositionY())() | Get the position on which the shark is placed on the y-axis. |
| int | [**getPositiveNeighbourX**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#getPositiveNeighbourX())() | Get the value of the field on the right if it exists. |
| int | [**getPositiveNeighbourY**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#getPositiveNeighbourY())() | Get the value of the field below if it exists. |
| void | [**increaseAge**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#increaseAge())() | Increase the age of the Shark by one. |
| static void | [**resetSumOfSharks**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#resetSumOfSharks())() | Set the amount of Sharks to zero. |
| void | [**setAge**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#setAge(int))​(int age) | Set the age of the Shark. |
| void | [**setPositionX**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#setPositionX(int))​(int positionX) | Set the position on the x-axis. |
| void | [**setPositionY**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#setPositionY(int))​(int positionY) | Set the position on the y-axis. |
| static void | [**setSumOfSharks**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#setSumOfSharks(int))​(int sumOfSharks) | Set the amount of Sharks to inserted num. |
| java.lang.String | [**toString**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html#toString())() |  |

### Methods inherited from class Project.[Animal](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html)

[isAlive](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html#isAlive())

### Methods inherited from class java.lang.Object

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

### *Constructor Detail*

#### Shark

public Shark​(int positionX,

int positionY)

Create a Shark.

**Parameters:**

positionX - giving the position on the x-axis.

positionY - giving the position on the y-axis.

### *Method Detail*

#### resetSumOfSharks

public static void resetSumOfSharks()

Set the amount of Sharks to zero.

#### setSumOfSharks

public static void setSumOfSharks​(int sumOfSharks)

Set the amount of Sharks to inserted num.

**Parameters:**

sumOfSharks - this int is the new value of sumOfSharks.

#### getNumOfSharks

public static int getNumOfSharks()

Get the amount of created Sharks. Increases with every new Shark-Object.

**Returns:**

Amount of Sharks

#### getNegativeNeighbourX

public int getNegativeNeighbourX()

Get the value of the field on the left if it exists. Otherwise, it returns -1000.

**Returns:**

a value of 0 - 14 or -1000

#### getPositiveNeighbourX

public int getPositiveNeighbourX()

Get the value of the field on the right if it exists. Otherwise, it returns -1000.

**Returns:**

a value of 1 - 15 or -1000

#### getNegativeNeighbourY

public int getNegativeNeighbourY()

Get the value of the field above if it exists. Otherwise, it returns -1000.

**Returns:**

a value of 0 - 7

#### getPositiveNeighbourY

public int getPositiveNeighbourY()

Get the value of the field below if it exists. Otherwise, it returns -1000.

**Returns:**

a value of 1 - 8

#### getPositionX

public int getPositionX()

Get the position on which the shark is placed on the x-axis.

**Returns:**

the position on the x-axis (width)

#### setPositionX

public void setPositionX​(int positionX)

Set the position on the x-axis.

**Parameters:**

positionX - the number of the position within the grid. (width)

#### getPositionY

public int getPositionY()

Get the position on which the shark is placed on the y-axis.

**Returns:**

the position on the y-axis (row)

#### setPositionY

public void setPositionY​(int positionY)

Set the position on the x-axis.

**Parameters:**

positionY - the number of the position within the grid. (row)

#### getBirthProbability

public static double getBirthProbability()

Get the birth probability.

**Returns:**

the constant defined birth probability

#### increaseAge

public void increaseAge()

Increase the age of the Shark by one.

#### setAge

public void setAge​(int age)

Set the age of the Shark.

**Parameters:**

age - an int 0<

#### getColour

public java.awt.Color getColour()

Get the colour shown on the grid for Animal-Type Shark.

**Specified by:**

[getColour](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Colourful.html#getColour()) in interface [Colourful](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Colourful.html)

**Specified by:**

[getColour](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html#getColour()) in class [Animal](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html)

**Returns:**

colour of Shark

#### toString

public java.lang.String toString()

**Overrides:**

toString in class java.lang.Object

# JavaDoc extract: Class Model

**Package** [Project](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\package-summary.html)

## Class Model

* java.lang.Object
  + Project.Model

**All Implemented Interfaces:**

java.io.Serializable

public class **Model**

extends java.lang.Object

implements java.io.Serializable

This class sets up the animal grid, which then will be visualised by the Gui class.

**See Also:**

[Serialized Form](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\serialized-form.html#Project.Model)

### *Constructor Summary*

|  |  |
| --- | --- |
| **Constructors** | |
| **Constructor** | **Description** |
| [**Model**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#%3Cinit%3E(int,int))​(int width, int row) | Create a Model to handle storing, moving, replacing animals. |

### *Method Summary*

All MethodsInstance MethodsConcrete Methods

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Description** |
| void | [**checkAge**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#checkAge(Project.Shark))​([**Shark**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html) shark) | deletes Sharks that reached age of 60 |
| [**Animal**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html) | [**getAnimal**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#getAnimal(int,int))​(int positionX, int positionY) | Returns the Animal which is placed in the 2d Animal Array |
| java.util. ArrayList | [**getNeighbours**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#getNeighbours(Project.Shark))​([**Shark**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html) shark) | This method stores possible directions the shark can swim. |
| java.io.File | [**getStorageFile**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#getStorageFile())() | Get the save file. |
| boolean | [**isSaved**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#isSaved())() | This method is to check if the storageFile was safed, otherwise it will not load. |
| void | [**letSharkSwim**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#letSharkSwim())() | This method iterates through the storeSharks ArrayList and will call the moveShark method when the shark is alive. |
| void | [**loadModel**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#loadModel(java.io.File))​(java.io.File file) | This method loads the Animal Array. |
| void | [**moveShark**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#moveShark(java.util.ArrayList,Project.Shark))​(java.util.ArrayList whereToSwim, [**Shark**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html) shark) | This method will place the shark at a neighbour position, if there are any possible directions passed. |
| void | [**placeAnimal**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#placeAnimal())() | Places the Animal in the 2d grid. |
| void | [**resetAllAnimal**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#resetAllAnimal())() | Clears the 2d Array. |
| void | [**resetStoreSharksArrayList**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#resetStoreSharksArrayList())() | This method clears the ArrayList storeSharks. |
| void | [**saveModel**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#saveModel(java.io.File))​(java.io.File file) | This method saves the Animal Array. |
| void | [**spawnAShark**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#spawnAShark())() | This method will increase the amount of sharks (create a new object) if the probability strikes. |
| void | [**tellMeWhatsInside**](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Model.html#tellMeWhatsInside())() | This method is for testing purposes. |

### Methods inherited from class java.lang.Object

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

### *Constructor Detail*

#### Model

public Model​(int width,

int row)

Create a Model to handle storing, moving, replacing animals.

**Parameters:**

width - this int gets passed from the driver class and sets the with within grid

row - this int gets passed from the driver class and sets the amount of columns within the grid

### *Method Detail*

#### placeAnimal

public void placeAnimal()

Places the Animal in the 2d grid.

#### getAnimal

public [Animal](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Animal.html) getAnimal​(int positionX, int positionY)

Returns the Animal which is placed in the 2d Animal Array

**Parameters:**

positionX - the number of the position within the grid. (width)

positionY - the number of the position within the grid. (row)

**Returns:**

the animal object which was placed within the 2d array

#### resetAllAnimal

public void resetAllAnimal()

Clears the 2d Array. All positions set to null.

#### saveModel

public void saveModel​(java.io.File file)

This method saves the Animal Array.

Created with help from Persistence notes (Powerpoint) and other sources: https://stackoverflow.com/questions/1467193/java-serialization-of-multidimensional-array

**Parameters:**

file - is created when constuctor is envoked and gets stored within the running program

#### loadModel

public void loadModel​(java.io.File file)

This method loads the Animal Array.

The loading was challenging, because the 2dArray had sometimes null assigned to it. Therefore, I added the class Water to never have null assigned within the Array.

Created with help from Persistence notes (Powerpoint) and other source: https://stackoverflow.com/questions/1467193/java-serialization-of-multidimensional-array

**Parameters:**

file - is created when constuctor is envoked and gets stored within the running program

#### getStorageFile

public java.io.File getStorageFile()

Get the save file.

**Returns:**

the file which was saved

#### letSharkSwim

public void letSharkSwim()

This method iterates through the storeSharks ArrayList and will call the moveShark method when the shark is alive.

#### getNeighbours

public java.util.ArrayList getNeighbours​([Shark](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html) shark)

This method stores possible directions the shark can swim.

**Parameters:**

shark - this object shall be moved

**Returns:**

an ArrayList filled with free neighbour fields.

#### moveShark

public void moveShark​(java.util.ArrayList whereToSwim, [Shark](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html) shark)

This method will place the shark at a neighbour position, if there are any possible directions passed. Furthermore, will it call the aging process.

**Parameters:**

whereToSwim - the list contains possible directions or can be null

shark - the object that shall be moved

#### checkAge

public void checkAge​([Shark](file:///C:\Users\marcg\OneDrive\Desktop\JavaDoc\Project\Shark.html) shark)

deletes Sharks that reached age of 60

**Parameters:**

shark - the object which age needs to be checked

#### spawnAShark

public void spawnAShark()

This method will increase the amount of sharks (create a new object) if the probability strikes.

#### resetStoreSharksArrayList

public void resetStoreSharksArrayList()

This method clears the ArrayList storeSharks.

#### isSaved

public boolean isSaved()

This method is to check if the storageFile was safed, otherwise it will not load.

**Returns:**

a value true or false

#### tellMeWhatsInside

public void tellMeWhatsInside()

This method is for testing purposes. Is the populating of the grid working?



# **ITT 4 COL2**Declaration of Originality



|  |  |
| --- | --- |
| **‘Declaration of Originality Form’** - Institute of Technology, Tralee. | |
| This form **must** be completed and signed and submitted with all assignments. | |
| Please complete the information below (using BLOCK CAPITALS). | |
| Name: Marc Giger  T Number: T00210159  Class Group: Erasmus  Assignment Title: OOP Hand-in Project Assignment | |
| **Students are advised to inform themselves of the Institute Anti-Plagiarism Policy.** | |
| **I confirm that this assignment is my own work and that I have:** | |
| Familiarised myself with the Institute’s Anti-Plagiarism Policy | 🖾 |
| Used the Institute’s approved referencing style throughout | 🖾 |
| Clearly referenced, in both the text and the bibliography or references, all sources used in the work  Not made use of the work of any other student(s) past or present without acknowledgement. This includes any of my own work, that has been previously, or concurrently, submitted for assessment, either at this or any other educational institution | 🖾  🖾 |
| Not sought or used the services of any professional agencies to produce this work | 🖾 |
| In addition, I understand that any false claim in respect of this work will result in disciplinary action in accordance with Institute regulations | 🖾 |
|  |  |
|  |  |
|  |  |
| **DECLARATION:**  I am aware of and understand the Institutes’ policy on plagiarism and I certify that this assignment is my own work**,** except where indicated by referencing, and that I have followed the good academic practices noted above  Signed | |