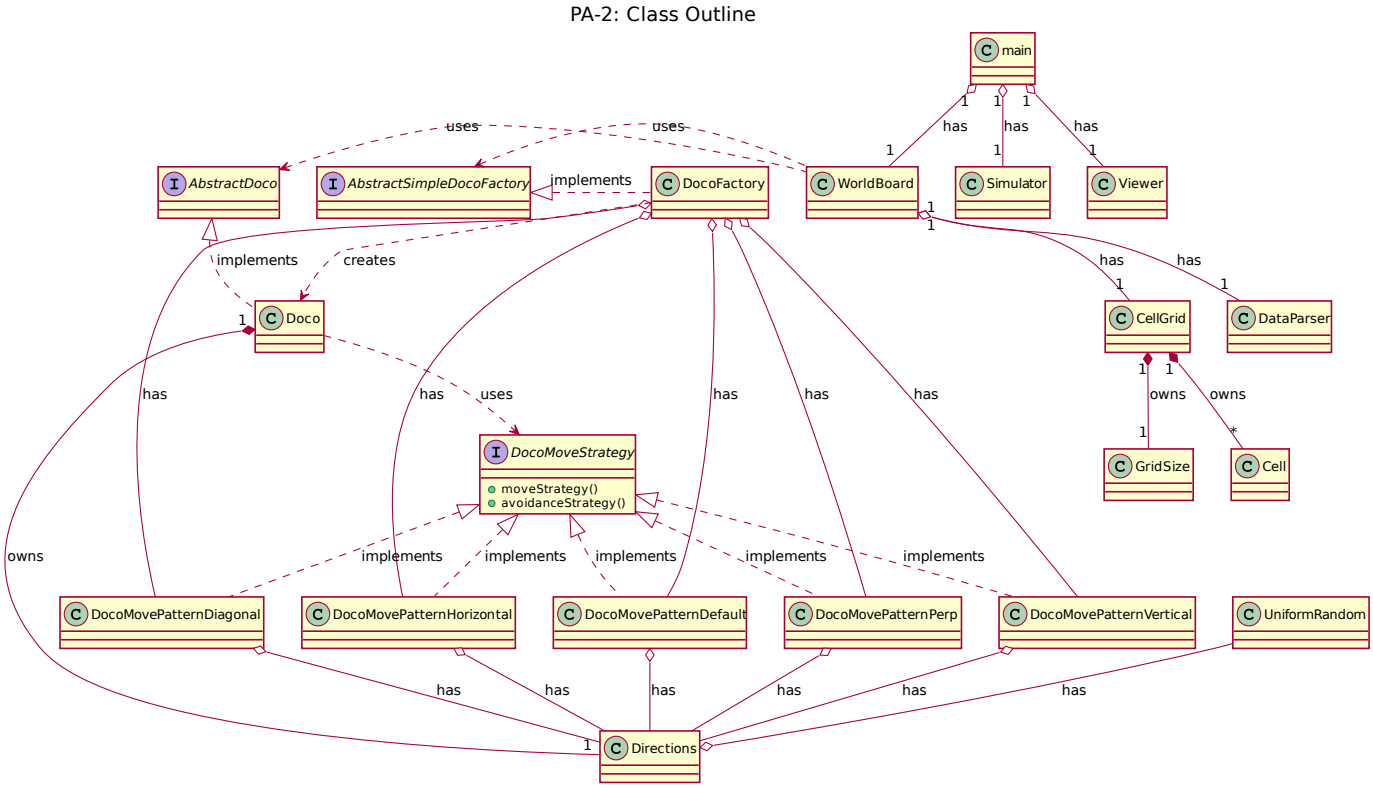
Digital One-Celled Organism (DOCO) Simulation

Programming Assignment 2 (PA-2)
Fall 2020, CS-307

Table of Contents

Preliminary Class Diagram



Last Updated: 10/23/2020

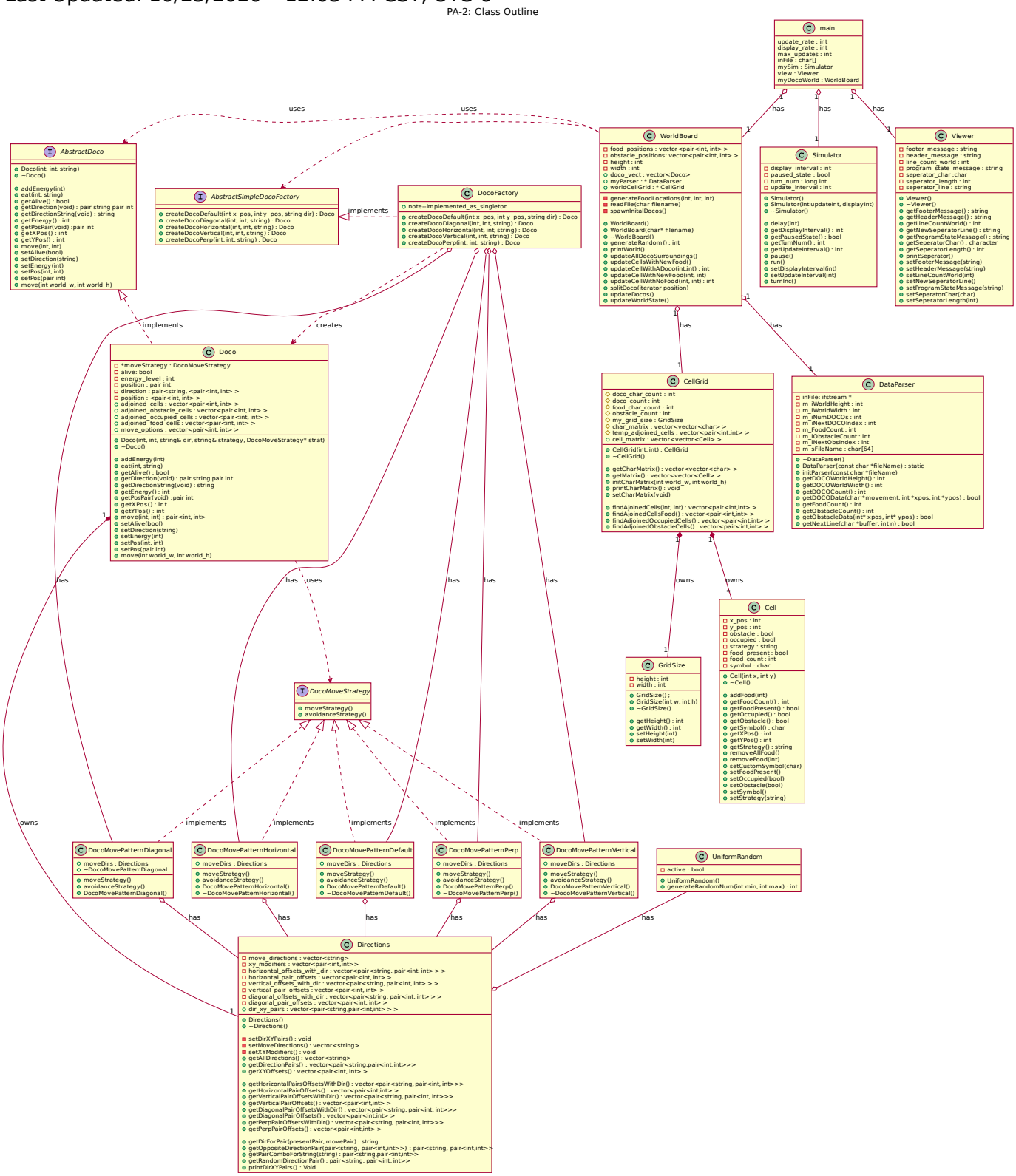
System Overview

Last Updated: 10/23/2020 – 11:58 AM CST, UTC-6

This is a console program that will run a simulation of a world where single celled organisms are spawned in. They will run around eating the food nearby and avoiding walking past the edge of the world. They will avoid walking into each other too. They now also avoid running into obstacles. Additionally strategy pattern and factory pattern have been implemented to allow for different types of DOCOs. The initial world specifications (height, width, DOCO’s and positions, and food locations) will be read in from a provided file in XML format. As part of version two, the DOCO strategy is also read in. There are a lot of details involved and they are specified below.

Class Outline UML Diagram

Last Updated: 10/23/2020 – 12:03 PM CST, UTC-6



Class Outline Written

Last Updated: 9/20/2020 - 3:40 PM CST, UTC-6

Important Note: *Return value specified in brackets. Arguments shown inside function. Actions performed explained.*

1 - Main

1.1 - Summary

1.1.1 - The main class is to instantiate the class objects and run the program.

1.2 - Properties

1.2.1 - private:
1.2.1.1 - [int] update_rate – rate of simuluation updating stored here.
1.2.1.2 - [int] display_rate – rate of simulation displaying stored here.
1.2.1.3 - [int] max_updates – maximum number of updates stored here.
1.2.1.4 - [string] inFile –initialization file for objects
1.2.2 - public:
1.2.2.1 - [Viewer] view
1.2.2.2 - [WoarldBoard] myDocoWorld

2 - Viewer

2.1 - Summary

2.1.1 - the viewer class is just to hold some data for what will be output to the console.

2.2 - Properties

2.2.1 - private:
2.2.1.1 - [string] header_message – holds the start message at the top of the console.
2.2.1.2 - [int] line_count_world – keeps the total line count, basically number of matrix rows.
2.2.1.3 - [string] footer_message – text at bottom of console.
2.2.1.4 - [string] program_state_message – information related to the simulation
2.2.1.5 - [char] seperator_char
2.2.1.6 - [int] seperator_length
2.2.1.7 - [string] seperator_line

2.3 - Methods

2.3.1 - public:
2.3.1.1 - [constructor] viewer() - create a viewer object
2.3.1.2 - [destructor] ~viewer – destroy the viewer object
2.3.1.3 - [string] getFooterMessage() - return end_message
2.3.1.4 - [string] getHeaderMessage() - return header_message
2.3.1.5 - [int] getLineCountWorld() - return line_count_world
2.3.1.6 - [string] getNewSeperatorLine() - return the separator line
2.3.1.7 - [string] getProgramStateMessage() - return program_state_message
2.3.1.8 - [char] getSeperatorChar() - gives you the character used in the separator.
2.3.1.9 - [int] getSeperatorLength() - reveals the length of the separator
2.3.1.10 - printSeperator() - prints the separator string
2.3.1.11 - setFooterMessage(string) – update bottom_text
2.3.1.12 - setHeaderMessage(string) – update header_message

2.3.1.13 - setLineCountWorld(int) – update line_count_world
2.3.1.14 - setNewSeperatorLine() - updates the line string with set attributes provided previously.
2.3.1.15 - setProgramStateMessage(string) – update program_state_message
2.3.1.16 - setSeperatorChar() - sets the character for the separator line.
2.3.1.17 - setSeperatorLength() - sets the separator length

3 - Simulator

3.1 - Summary

3.1.1 - Responsible for maintaining the changing game state and status.

3.2 - Properties

3.2.1 - private:
3.2.1.1 - [int] display_interval - the update interval for displaying the world
3.2.1.2 - [bool] paused_state – whether the world is supposed to be running or not
3.2.1.3 - [int] turn_num – the current update of the world
3.2.1.4 - [int] update_interval - the update interval for the world

3.3 - Methods

3.3.1 - public:
3.3.1.1 - [constructor] Simulator() - creates simulator object with default settings
3.3.1.2 - [constructor] Simulator(int updateInterval, int displayInterval) – creates the simulator with specific intervals for game updates and display updates.
3.3.1.3 - [destructor] ~Simulator() - de-allocates space for the simulator object
3.3.1.4 - delay(int) -
3.3.1.5 - [int] getGisplayInterval() - return display_update_interval
3.3.1.6 - bool] getPausedState() - return paused
3.3.1.7 - [int] getTurnNum() - return turn_num
3.3.1.8 - [int] getUpdateInterval() - return update_interval
3.3.1.9 - Pause() - set paused to True
3.3.1.10 - Run() - set paused to False
3.3.1.11 - setDisplayInterval(int) – change the rate at which the World is updating on the screen. Adjust display_interval to the value provided.
3.3.1.12 - setUpdateInterval(int) – change the rate at which the World is updating. Adjust update_interval to the value provided.
3.3.1.13 - Turninc() - increments the turn count.

4 - WorldBoard

4.1 - Summary

4.1.1 - The world board is responsible for holding all the entities of the DOCO simulation. Creating it will create the other objects.

4.2 - Properties

4.2.1 - private:
4.2.1.1 - [vector] food_positions – store food positions
4.2.1.2 - [vector] obstacle_positions – store obstacle positions
4.2.1.3 - [int] height - store board height
4.2.1.4 - [int] width – store board width
4.2.2 - public:

4.2.2.1 - [vector<DOCO>] doco_vect – will hold a vector of all the current DOCO's on the board. These will be able to be iterated over and removed as part of the vector class functionality.
4.2.2.2 - [DataParser] myParser – the DataParser obejct for the class. mytParser = DataParser(char *filename). This filename will be DOCOData02.xml
4.2.2.3 - [Simulator] mySim – containers the DOCOSim Object
4.2.2.4 - [CellGrid] worldCellGrid – will hold the CellGrid Object which contains all the Cells and GridSize

4.3 - Methods

4.3.1 - private:
4.3.1.1 - generateFoodLocations(int height, int width, int count) – creates the food spawn locations and adds to a vector to be placed on the board.
4.3.1.2 - readFile() - reads the file in
4.3.1.3 - spawninitialDocos()
4.3.2 - public:
4.3.2.1 - [contructor] WorldBoard() - build WorldBoard object
4.3.2.2 - [constructor] WorldBoard(filename) – build WolrdBoard object given filename
4.3.2.3 - [deconstructor] ~WorldBoard()
4.3.2.3.1 - PrintWorld() - this print the Cell Board and use the CellGrid built in function to do it.
4.3.2.3.2 - UpdateAllDocoSurroundings() - updates surroundings of the doco
4.3.2.3.3 - updateCellsWithNewFood() - updates cell to have no food inside it
4.3.2.3.4 - updateCellWithADoco() - adds occupied = true to the cell
4.3.2.3.5 - updateCellWithNewFood() - adds food to that cell
4.3.2.3.6 - updateCellWithNoFood() - remove food from a cell
4.3.2.3.7 - [Doco] splitDoco() - modify a doco to half energy, then copy, set copy to opposite direction
4.3.2.3.8 - UpdateDocos() - update all the doco's next move and stats one at a time.
4.3.2.3.9 - UpdateWorldState() - updates the entire worldBoard

5 - DOCO

5.1 - Summary

5.1.1 - A DOCO is an organism object that moves around on the CellGrid based off it's own desires and what is immediately around it.

5.2 - Properties

5.2.1 - private:
5.2.1.1 - [docoMoveStragey] moveStrategy
5.2.1.2 - [bool] alive – whether or not the DOCO is alive or dead, if it's dead it should be removed or become invisible on the screen.
5.2.1.3 - [int] energy_level – the amount of energy the DOCO has. It will be initialized to 500 by default.
5.2.1.4 - [pair int] position
5.2.1.5 - [string] direction – A direction that the DOCO is currently heading. It will be be one of the following strings "N", "NE", "E", "SE", "S", "SW", "W", "NW".
5.2.1.6 - [pair int] position
5.2.2 - public:
5.2.2.1 - [matrix] adjoined_cells – this will contain the matrix of adjoining cells to a DOCO. Adjoining means only the cells are touching, diagonal included.

5.2.2.2 - [matrix] adjoined_obstacle_cells – this will contain the matrix of adjoining cells that are obstacles.
5.2.2.3 - [matrix] adjoined_occupied_cells – this will contain the matrix of adjoining cells that are occupied.
5.2.2.4 - [matrix] adjoined_food_cells – this will contain the matrix of adjoining cells that contain food. This is why the Cell object has a food_present boolean property.
5.2.2.5 - [matrix] move_options – this will contain the matrix of movement options that are available to the DOCO based on it's movement preferences and requirements.

5.3 - Methods

5.3.1 - public:
5.3.1.1 - [constructors] Doco(int x, int y, std::string& start_dir, std::string& strategy, DocoMoveStrategy* strat); // starting position, x, y, direction
5.3.1.2 - [destructor] ~DOCO() - deletes the DOCO
5.3.1.3 - addEnergy(int) – add the specified amount of energy to the DOCO's energy_level
5.3.1.4 - eat(x_pos, y_pos) – the DOCO regenerates 50 energy for each pellet eaten, and it eats all the pellets at this location. This call the CellGrid.Matrix.SpecificCell.setFoodPresent(bool) and setSymbol(char), removeAllFood() commands for the cell being eaten off of.
5.3.1.5 - [bool] getAlive() - returns whether the DOCO is alive or dead.
5.3.1.6 - [string pair int] getDirection() - returns the current direction of the DOCO
5.3.1.7 - [string] getDirectionString() - returns the current direction of the DOCO
5.3.1.8 - [int] getEnergy() - returns the energy_level of the DOCO
5.3.1.9 - [pair int] getPosPair
5.3.1.10 - [int] getXPos()
5.3.1.11 - [int] getYPos()
5.3.1.12 - [pair<int, int>] move(int, int) – doco chooses new move position and goes to it.
5.3.1.13 - setAlive(bool) – updates the alive status of the DOCO.
5.3.1.14 - setDirection(string) – sets the direction of the DOCO, this will be one of the following strings "N", "NE", "E", "SE", "S", "SW", "W", "NW". Upon initialization this will be random or taken from the read in file.
5.3.1.15 - setEnergy(int) – set the energy_level of the DOCO to a specified amount.
5.3.1.16 - setPos(x_pos, y_pos) – updates the x and y position of the DOCO.
5.3.1.17 - setPos(pair int) – sets Doco position

6 - Cell

6.1 - Summary

6.1.1 - Cells are rigid objects on the board, they don't move or change positions. There will be many of these mapped onto a Cell Grid.

6.2 - Properties

6.2.1 - private:
6.2.1.1 - [int] x_pos – this will store the x position of a cell
6.2.1.2 - [int] y_pos – this will store the y position of a cell
6.2.1.3 - [bool] obstacle – this will store whether a cell is currently occupied by an obstacle

6.2.1.4 - [bool] occupied – this will store whether a cell is currently occupied by a DOCO or other organism.
6.2.1.5 - [string] strategy -show the DOCO movement strategy
6.2.1.6 - [bool] food_present – this will store whether there is food present in a cell so that the DOCO can smell it and go to it when nearby
6.2.1.7 - [int] food_count - this will store the food count in the cell
6.2.1.8 - [char] symbol – will store the symbol to print to the board for the location

6.3 - Methods

6.3.1 - public:
6.3.1.1 - [constructor] Cell(x_pos, y_pos) – create the cell object, it can not be created without a position on the board.
6.3.1.2 - [destructor] ~Cell – de-allocate memory for the Cell when program ends.
6.3.1.3 - addFood(int) – add food to the Cell with the amount specified
6.3.1.4 - [bool] getFoodCount() - returns the number of food pellets in the cell
6.3.1.5 - [bool] getFoodPresent() - returns whether food is present or not in the cell
6.3.1.6 - [bool] getOccupied() - returns whether or not the cell is occupied by a DOCO
6.3.1.7 - [bool] getObstacle() - returns whether or not the cell is occupied by an Obstacle
6.3.1.8 - [char] getSymbol() - gets the character symbol for this cell
6.3.1.9 - [int] getXPos() - return the x_pos of the Cell
6.3.1.10 - [int] getYPos() - returns the y_pos of the Cell
6.3.1.11 - [string] getStrategy() - return a movement Stategy for a doco in the cell
6.3.1.12 - removeAllFood() - set the food count to zero
6.3.1.13 - removeFood(int) – remove food pellets from cell with amount specified
6.3.1.14 - setCusstomSymbol(char) – set character symbol for the cell to the char provided.
6.3.1.15 - setFoodPresent(bool) – set whether there is any food in the Cell
6.3.1.16 - setOccupied(bool) – set whether the cell is occupied or not by a DOCO
6.3.1.17 - setObstacle(bool) – set whether the cell is occupied or not by a Obstacle
6.3.1.18 - setSymbol(char) – set character symbol for the cell to the char provided.
6.3.1.19 - setStragey(string) – set DOCO movement strategy If DOCO present

7 - GridSize

7.1 - Summary

7.1.1 - GridSize contains your grid shape info.

7.2 - Properties

7.2.1 - private:
7.2.2 - [int] height – this is for the height of the grid
7.2.3 - [int] width – this for the width of the grid

7.3 - Methods

7.3.1 - public:
7.3.1.1 - [constructor] GridSize() -
7.3.1.2 - [constructor] GridSize(int width, int height) – in order to make a grid object it should be required that the width and height are there.
7.3.1.3 - [destructor] ~GridSize() - want to de-allocate memory when this is destroyed
7.3.1.4 - [int] getHeight() - returns height
7.3.1.5 - [int] getWidth() - returns width
7.3.1.6 - setHeight(int) – sets [private] height
7.3.1.7 - setWidth(int) – sets [private] width

8 - CellGrid

8.1 - Summary

8.1.1 - Is a grid of cell object in the form of a matrix

8.2 - Properties

8.2.1 - private
8.2.1.1 - [int[doco_char_count - # of doco's on the board
8.2.1.2 - [int] doco_count - # of docos from vector
8.2.1.3 - [int] food_char_count - # number of unique food positions on the board.
8.2.1.4 - [int] obstacle_count - # of obstacles
8.2.1.5 - [GridSize] my_grid_size – size of the grid
8.2.1.6 - [matrix] char_matrix – holds the matrix of cells in character format.
8.2.1.7 - [matrix] temp_adjoined_cells – holds temporary adjoined cells for each time findAdjoinedCells(x_pos, y_pos) is called.
8.2.2 - Public
8.2.2.1 - [Cell] cell_matrix – holds the matrix of cell objects

8.3 - Methods

8.3.1 - public:
8.3.1.1 - [constructor] CellGrid(height, width) – creates the gridShape to the specified height and width, then populates the cell_matrix with cell_objects initialized to each position.
8.3.1.2 - [destructor] ~CellGrid() - de-allocates memory for CellGrid object
8.3.1.3 - [matrix] getCharMatrix() - returns the char_matrix private variable
8.3.1.4 - [matrix] getMatrix() - returns the cell_matrix private variable
8.3.1.5 - [matrix] initCharMatrix() -
8.3.1.6 - [void] printCharMatrix() - returns the character matrix in printed form based off of the char_matrix class property.
8.3.1.7 - [matrix] setCharMatrix() - set the char_matrix to their appropriate characters based on the status of the cells.
8.3.1.8 - [matrix] findAdjoinedCells(x_pos, y_pos) – using the x and y position provided in conjunction with its data on the cell matrix, finds the cells within one space of it (N, E, S, W, NE, SE, SW, NW).
8.3.1.9 - [matrix] findAdjoinedccupiedCells() – using the temporary adjoining cell matrix part of CellGrid, it returns a new matrix of just the cells that are occupied around it. Using this temp variable allows removing some error checking here.
8.3.1.10 - [matrix] findAdjoinedCellsFood() – checks each of the temp_adjoining_cells and returns the matrix of cells that contain food. Using the temp variable eliminates some error checking.
8.3.1.11 - [matrix] findAdjoinedObstacleCells() – using the temporary adjoining cell matrix part of CellGrid, it returns a new matrix of just the cells that are occupied around it. Using this temp variable allows removing some error checking here.

9 - DataParser

9.1 - Summary

9.1.1 - The data parser will read in the provided file and it will specify the number of DOCOs to spawn and the number of food pellets to spawn. Sample file to

read in is in an XML format. This class is fully written and supplied already, so no code / pseudo code is necessary or required. It is in here for simple reference.

9.2 - Properties

9.2.1 - private:
9.2.1.1 - [ifstream] *inFile – DOCO world definition file
9.2.1.2 - [int] m_iWorldWidth – number of cells wide for DOCO grid read in
9.2.1.3 - [int] m_iWorldHeight – number of cells high for the DOCO grid read in
9.2.1.4 - [int] m_iNumDOCOs – Number of DOCOs in the world
9.2.1.5 - [int] m_iNextDOCOIndex – Index of next DOCO to read
9.2.1.6 - [int] m_FoodCount – Number of initial food pellets
9.2.1.7 - [int] m_ObstacleCount – Number of initial obstacles
9.2.1.8 - [int] m_iNextObsIndex – Index of next Obstacle to read
9.2.1.9 - [char[64]] m_sFileName – Data File name string

9.3 - Methods

9.3.1 - public:
9.3.1.1 - [constructor] DataParser(char *fileName) – creates the object an initializes from a file provided
9.3.1.2 - [destructor] ~DataParser() - destroys the object with delete
9.3.1.3 - initParser(char *filename) -
9.3.1.4 - [int] getDOCOWorldWidth() - returns the width of the world
9.3.1.5 - [int] getDOCOWorldHeight() - returns the height of the world
9.3.1.6 - [int] getDOCOCount() - returns how many DOCOs are to be spawned in
9.3.1.7 - [bool] getDOCOData(char *movement, int *xpos, int *ypos) – Reads to the current DOCO count. Returns true or false based on whether data for another DOCO is present.
9.3.1.8 - [int] getFoodCount() - returns amount of food to spawn in.
9.3.1.9 - [int] getObstacleCount() -
9.3.1.10 - [bool]getObstacleData(int x, int y) -
9.3.1.11 - [bool] getNextLine(char *buffer, int n) – Reads lines from a file and places them in buffer, removing any leading white space. Skips blank lines. Ignores comments starting with <!-- and ending with →. Returns true for a successful read, false if the end of file was encountered.

10 - Directions

10.1 - Summary

10.1.1 - Responsible for creating direction pairs associated with cardinal directions. Maps to a grid.

10.2 - Properties

10.2.1 - private:
10.2.1.1 - [vector string] move_directions
10.2.1.2 - [vector pair int] xy_modifiers
10.2.1.3 - [vector pair int] horizontal_offsets_with_dir
10.2.1.4 - [vector pair str int] horizontal_pair_offsets
10.2.1.5 - [vector pair int] vertical_offsets_with_dir
10.2.1.6 - [vector pair str int] vertical_pair_offsets
10.2.1.7 - [vector pair int] diagonal_offsets_with_dir
10.2.1.8 - [vector pair str int] diagonal_pair_offsets
10.2.2 - public:

10.2.2.1 - [vector pair str int] dir_xy_pairs

10.3 - Methods

10.3.1 - private:
10.3.1.1 - setDirXYPairs() - initializes the direction “N” and offset pair for each direction.
10.3.1.2 - SetMoveDirections() - sets the cardinal move options
10.3.1.3 - setXYModifiers() - sets the XY offset modifiers
10.3.2 - public:
10.3.2.1 - [constructor] Directions()
10.3.2.2 - [deconstructor] ~Directions()
10.3.2.3 - getAllDirections() - returns all possible directions
10.3.2.4 - getDirectionPairs() - returns all direction and pair combos
10.3.2.5 - getXYOffsets() - returns the offsets for x and y
10.3.2.6 - getHorizontalPairsOffsetsWithDir()
10.3.2.7 - getHorizontalPairsOffsets()
10.3.2.8 - getVerticalPairsOffsetsWithDir()
10.3.2.9 - getVerticalPairsOffsets()
10.3.2.10 - getDiagonalPairsOffsetsWithDir()
10.3.2.11 - getDiagonalPairsOffsets()
10.3.2.12 - getPerpPairsOffsetsWithDir()
10.3.2.13 - getPerpPairsOffsets()
10.3.2.14 - getDirForPair(string) - returns the direction given a pair offset.
10.3.2.15 - getOppositeDirectionPair(string) - returns the opposite direction of what’s passed in.
10.3.2.16 - getPairComboForString()
10.3.2.17 - getRandomDirectionPair()
10.3.2.18 - printDirXYPairs() - prints all the pair offsets with the associated direction to cout.

11 - DocoFactory

11.1 - Summary

11.1.1 - Responsible for creating docos of varying types

11.2 - Properties

11.2.1 - private:
11.2.1.1 - [int] instance number

11.3 - Methods

11.3.1 - public:
11.3.1.1 - [Doco] createDocoDefault(int x_pos, int y_pos, std::string direction)
11.3.1.2 - [Doco] createDocoDiagonal(int x_pos, int y_pos, std::string direction)
11.3.1.3 - [Doco] createaeDocoHorizontal(int x_pos, int y_pos, std::string direction)
11.3.1.4 - [Doco] createDocoVertical(int x_pos, int y_pos, std::string direction)
11.3.1.5 - [Doco] createDocoPerp(int x_pos, int y_pos, std::string direction)

12 - DocoMovePatternDiagonal

12.1 - Summary

12.1.1 - Responsible for the diagonal move pattern preference and the move avoidance preference. Goal is to return pairs that it can go to given it’s pattern.

12.2 - Properties

12.2.1 - public:
12.2.1.1 - [Directions] moveDirs

12.3 - Methods
12.3.1 - public:
12.3.1.1 - moveStrategy() - returns the pairs diagonal can move to
12.3.1.2 - avoidanceStrategy() - returns the pairs diagonal moves to when a wall is hit
12.3.1.3 - [constructor]DocoMovePatternDiagonal()
12.3.1.4 - [destructor]~docoMovePatternDiagonal

13 - DocoMovePatternHorizontal
13.1 - Summary
13.1.1 - Responsible for the Horizontal move pattern preference and the move avoidance preference. Goal is to return pairs that it can go to given it's pattern.
13.2 - Properties
13.2.1 - public:
13.2.1.1 - [directions] moveDirs
13.3 - Methods
13.3.1 - public:
13.3.1.1 - moveStrategy() - returns the pairs horizontal move pattern can move to.
13.3.1.2 - avoidanceStrategy() - returns the pairs horizontal moves to when a wall is hit.
13.3.1.3 - [constructor]DocoMovePatternDiagonal()
13.3.1.4 - [deconstructor] ~DocoMovePatternDiagonal()

14 - DocoMovePatternDefault
14.1 - Summary
14.1.1 - Responsible for the default move pattern preference and the move avoidance preference. Goal is to return pairs that it can go to given it's pattern.
14.2 - Properties
14.2.1 - public:
14.2.1.1 - [directions] moveDirs
14.3 - Methods
14.3.1 - public:
14.3.1.1 - moveStrategy() - returns the pairs a default pattern moves to
14.3.1.2 - avoidanceStrategy() - returns the pairs a default pattern moves to when a wall is hit
14.3.1.3 - [constructor]DocoMovePatternDefaultl()
14.3.1.4 - [deconstructor] ~DocoMovePatternDefault()

15 - DocoMovePatternPerp
15.1 - Summary
15.1.1 - Responsible for the perpendicular move pattern preference and the move avoidance preference. Goal is to return pairs that it can go to given it's pattern.
15.2 - Properties
15.2.1 - public:
15.2.1.1 - [directions] moveDirs
15.3 - Methods
15.3.1 - public:
15.3.1.1 - moveStrategy() - returns the pairs a perpendicular pattern can move to

15.3.1.2 - avoidanceStrategy() - returns the pairs a perpendicular pattern can move to when a wall is hit.
15.3.1.3 - [constructor]DocoMovePatternPerp()
15.3.1.4 - [deconstructor] ~DocoMovePatternPerp()

16 - DocoMovePatternVertical

16.1 - Summary

16.1.1 - Responsible for the vertical move pattern preference and the move avoidance preference. Goal is to return pairs that it can go to given it's pattern.

16.2 - Properties

16.2.1 - public:
16.2.1.1 - [directions] moveDirs

16.3 - Methods

16.3.1 - public:
16.3.1.1 - moveStrategy()
16.3.1.2 - avoidanceStrategy()
16.3.1.3 - [constructor]DocoMovePatternVertical()
16.3.1.4 - [deconstructor] ~DocoMovePatternVertical()

17 - UniformRandom

17.1 - Summary

17.1.1 - Responsible for generating random numbers in a uniform distribution.

17.2 - Properties

17.2.1 - private:
17.2.1.1 - [bool] active

17.3 - Methods

17.3.1 - public:
17.3.1.1 - [contructor]UniformRandom()
17.3.1.2 - [int] generateRandomNum(int, int) – generates a random number between a lower and upper bound inclusive.

18 - AbstractDoco

18.1 - Summary

18.1.1 - Responsible for creating the contract for what Doco needs to implement.

18.2 - Properties

18.3 - Methods

18.3.1 - public:
18.3.1.1 - Doco()
18.3.1.2 - ~Doco
18.3.1.3 - addEnergy()
18.3.1.4 - eat()
18.3.1.5 - getAlive()
18.3.1.6 - getDirection()
18.3.1.7 - getDirectionString()
18.3.1.8 - getEnergy()
18.3.1.9 - getPosPair()
18.3.1.10 - getXPos()
18.3.1.11 - getYPos()

18.3.1.12 - move()
18.3.1.13 - setAlive()
18.3.1.14 - setDirection()
18.3.1.15 - setEnergy()
18.3.1.16 - setPos()
18.3.1.17 - setPos()
18.3.1.18 - move()

19 - AbstractSimpleDocoFactory

19.1 - Summary

19.1.1 - Responsible for creating the contract for a Doco factory and what it has to implement.

19.2 - Properties

19.3 - Methods

19.3.1 - public:
19.3.1.1 - createDocoDefault()
19.3.1.2 - createDocoDiagonal()
19.3.1.3 - createDocoHorizontal()
19.3.1.4 - createDocoVertical()
19.3.1.5 - createDocoPerp()

20 - DocoMoveStrategy

20.1 - Summary

20.1.1 - Responsible for creating the move strategy of the doco, this is the interface Doco is programmed to.

20.2 - Properties

20.3 - Methods

20.3.1 - public:
20.3.1.1 - moveStrategy() - subclass must implement this
20.3.1.2 - avoidanceStrategy() - subclass must implement this

Class Outline PlantUML Text

Edit and Recreate Diagram with: https://www.planttext.com

Last Updated: 10/23/2020 – 12:04 PM CST, UTC-6

```
@startuml

title PA-2: Class Outline

interface DocoMoveStrategy
{
    +moveStrategy()
    +avoidanceStrategy()
}

class Doco
{
    -*moveStrategy : DocoMoveStrategy
    -alive: bool
    -energy_level : int
    -position : pair int
    -direction : pair<string, <pair<int, int> >
    -position : <pair<int, int> >
    +adjoined_cells : vector<pair<int, int> >
    +adjoined_obstacle_cells : vector<pair<int, int> >
    +adjoined_occupied_cells : vector<pair<int, int> >
    +adjoined_food_cells : vector<pair<int, int> >
    +move_options : vector<pair<int, int> >
    +Doco(int, int, string& dir, string& strategy, DocoMoveStrategy* strat)
    +~Doco()

    +addEnergy(int)
    +eat(int, string)
    +getAlive() : bool
    +getDirection(void) : pair string pair int
    +getDirectionString(void) : string
    +getEnergy() : int
    +getPosPair(void) :pair int
    +getXPos()  :  int
    +getYPos()  : int
    +move(int, int) : pair<int, int>
    +setAlive(bool)
    +setDirection(string)
    +setEnergy(int)
    +setPos(int, int)
    +setPos(pair int)
    +move(int world_w, int world_h)
}

interface AbstractSimpleDocoFactory
{
    +createDocoDefault(int x_pos, int y_pos, string dir) : Doco
    +createDocoDiagonal(int, int, string) : Doco
    +createDocoHorizontal(int, int, string) : Doco
    +createDocoVertical(int, int, string) : Doco
    +createDocoPerp(int, int, string) : Doco
}

class DocoFactory {
```

```
    +note--implemented_as_singleton
    +createDocoDefault(int x_pos, int y_pos, string dir) : Doco
    +createDocoDiagonal(int, int, string) : Doco
    +createDocoHorizontal(int, int, string) : Doco
    +createDocoVertical(int, int, string) : Doco
    +createDocoPerp(int, int, string) : Doco
}

class CellGrid
{
    #doco_char_count : int
    #doco_count : int
    #food_char_count : int
    #obstacle_count : int
    #my_grid_size : GridSize
    #char_matrix :  vector<vector<char> >
    #temp_adjoined_cells :  vector<pair<int,int> >
    +cell_matrix : vector<vector<Cell> >

    +CellGrid(int, int) : CellGrid
    +~CellGrid()

    +getCharMatrix() :  vector<vector<char> >
    +getMatrix() : vector<vector<Cell> >
    +initCharMatrix(int world_w, int world_h)
    +printCharMatrix() : void
    +setCharMatrix(void)

    +findAjoinedCells(int, int) : vector<pair<int,int> >
    +findAjoinedCellsFood() :  vector<pair<int,int> >
    +findAdjoinedOccupiedCells() : vector<pair<int,int> >
    +findAdjoinedObstacleCells() : vector<pair<int,int> >

}

class WorldBoard
{
    -food_positions : vector<pair<int, int> >
    -obstacle_positions: vector<pair<int, int> >
    -height : int
    -width : int
    +doco_vect : vector<Doco>
    +myParser : * DataParser
    +worldCellGrid : * CellGrid

    -generateFoodLocations(int, int, int)
    -readFile(char filename)
    -spawnInitalDocos()

    +WorldBoard()
    +WorldBoard(char* filename)
    +~WorldBoard()
    +generateRandom() : int
    +printWorld()
    +updateAllDocoSurroundings()
    +updateCellsWithNewFood()
    +updateCellWithADoco(int,int) : int
    +updateCellWithNewFood(int, int)
    +updateCellWithNoFood(int, int) : int
    +splitDoco(iterator position)
    +updateDocos()
```

```
    +updateWorldState()
}

class main {
    update_rate : int
    display_rate : int
    max_updates : int
    inFile : char[]
    mySim : Simulator
    view : Viewer
    myDocoWorld : WorldBoard
}

class DataParser {
    -inFile: ifstream *
    -m_iWorldHeight : int
    -m_iWorldWidth : int
    -m_iNumDOCOs : int
    -m_iNextDOCOIndex : int
    -m_FoodCount : int
    -m_iObstacleCount : int
    -m_iNextObsIndex : int
    -m_sFileName : char[64]

    +~DataParser()
    +DataParser(const char *fileName) : static
    +initParser(const char *fileName)
    +getDOCOWorldHeight() : int
    +getDOCOWorldWidth() : int
    +getDOCOCount() : int
    +getDOCOData(char *movement, int *xpos, int *ypos) : bool
    +getFoodCount() : int
       +getObstacleCount() : int
       +getObstacleData(int* xpos, int* ypos) : bool
    +getNextLine(char *buffer, int n) : bool
}

class Directions {
    -move_directions : vector<string>
    -xy_modifiers : vector<pair<int,int>>
    -horizontal_offsets_with_dir : vector<pair<string, pair<int, int> > >
       -horizontal_pair_offsets : vector<pair<int, int> >
       -vertical_offsets_with_dir : vector<pair<string, pair<int, int> > >
       -vertical_pair_offsets : vector<pair<int, int> >
       -diagonal_offsets_with_dir : vector<pair<string, pair<int, int> > >
       -diagonal_pair_offsets : vector<pair<int, int> >
    +dir_xy_pairs : vector<pair<string,pair<int,int> > >

    +Directions()
    +~Directions()

    -setDirXYPairs() : void
    -setMoveDirections() : vector<string>
    -setXYModifiers() : void
    +getAllDirections() : vector<string>
    +getDirectionPairs() : vector<pair<string,pair<int,int>>>
    +getXYOffsets() : vector<pair<int, int> >

    +getHorizontalPairsOffsetsWithDir() : vector<pair<string, pair<int, int>>>
    +getHorizontalPairOffsets() : vector<pair<int,int> >
    +getVerticalPairOffsetsWithDir() : vector<pair<string, pair<int, int>>>
```

```
    +getVerticalPairOffsets() : vector<pair<int,int> >
    +getDiagonalPairOffsetsWithDir() : vector<pair<string, pair<int, int>>>
    +getDiagonalPairOffsets() : vector<pair<int,int> >
    +getPerpPairOffsetsWithDir() : vector<pair<string, pair<int, int>>>
    +getPerpPairOffsets() : vector<pair<int,int> >

    +getDirForPair(presentPair, movePair) : string
    +getOppositeDirectionPair(pair<string, pair<int,int>>) : pair<string, pair<int,int>>
    +getPairComboForString(string) : pair<string,pair<int,int>>
    +getRandomDirectionPair() : pair<string, pair<int, int>>
    +printDirXYPairs() : Void
}

class Simulator {
    -display_interval : int
    -paused_state : bool
    -turn_num : long int
    -update_interval : int

    +Simulator()
    +Simulator(int updateInt, displayInt)
    +~Simulator()

    +delay(int)
    +getDisplayInterval() : int
    +getPausedState() : bool
    +getTurnNum() : int
    +getUpdateInterval() : int
    +pause()
    +run()
    +setDisplayInterval(int)
    +setUpdateInterval(int)
    +turnInc()
}

class GridSize {
    -height : int
    -width : int

    +GridSize() ;
    +GridSize(int w, int h)
    +~GridSize()

    +getHeight() : int
    +getWidth() : int
    +setHeight(int)
    +setWidth(int)
}

class Cell {
    -x_pos : int
    -y_pos : int
    -obstacle : bool
    -occupied : bool
    -strategy : string
    -food_present : bool
    -food_count : int
    -symbol : char

    +Cell(int x, int y)
    +~Cell()
```

```
    +addFood(int)
    +getFoodCount() : int
    +getFoodPresent() : bool
    +getOccupied() : bool
    +getObstacle() : bool
    +getSymbol() : char
    +getXPos() : int
    +getYPos() : int
    +getStrategy() : string
    +removeAllFood()
    +removeFood(int)
    +setCustomSymbol(char)
    +setFoodPresent()
    +setOccupied(bool)
    +setObstacle(bool)
    +setSymbol()
    +setStrategy(string)
}

class Viewer {
    -footer_message : string
    -header_message : string
    -line_count_world : int
    -program_state_message : string
    -seperator_char :char
    -seperator_length : int
    -seperator_line : string

    +Viewer()
    +~Viewer()
    +getFooterMessage() : string
    +getHeaderMessage() : string
    +getLineCountWorld() : int
    +getNewSeperatorLine()  : string
    +getProgramStateMessage() : string
    +getSeperatorChar() : character
    +getSeperatorLength() : int
    +printSeperator()
    +setFooterMessage(string)
    +setHeaderMessage(string)
    +setLineCountWorld(int)
    +setNewSeperatorLine()
    +setProgramStateMessage(string)
    +setSeperatorChar(char)
    +setSeperatorLength(int)
}


interface AbstractDoco {
    +Doco(int, int, string)
    +~Doco()

    +addEnergy(int)
    +eat(int, string)
    +getAlive() : bool
    +getDirection(void) : pair string pair int
    +getDirectionString(void) : string
    +getEnergy() : int
    +getPosPair(void) :pair int
    +getXPos()  :  int
```

```
    +getYPos()  : int
    +move(int, int)
    +setAlive(bool)
    +setDirection(string)
    +setEnergy(int)
    +setPos(int, int)
    +setPos(pair int)
    +move(int world_w, int world_h)
}

class DocoMovePatternDiagonal {
    +moveDirs : Directions
    +moveStrategy()
    +avoidanceStrategy()
    +DocoMovePatternDiagonal()
    +~DocoMovePatternDiagonal
}

class DocoMovePatternHorizontal {
    +moveDirs : Directions
    +moveStrategy()
    +avoidanceStrategy()
    +DocoMovePatternHorizontal()
    +~DocoMovePatternHorizontal()
}

class DocoMovePatternVertical {
    +moveDirs : Directions
    +moveStrategy()
    +avoidanceStrategy()
    +DocoMovePatternVertical()
    +~DocoMovePatternVertical()
}

class DocoMovePatternDefault {
    +moveDirs : Directions
    +moveStrategy()
    +avoidanceStrategy()
    +DocoMovePatternDefault()
    +~DocoMovePatternDefault()
}

class DocoMovePatternPerp {
    +moveDirs : Directions
    +moveStrategy()
    +avoidanceStrategy()
    +DocoMovePatternPerp()
    +~DocoMovePatternPerp()
}

class UniformRandom
{
    -active : bool
    +UniformRandom()
    +generateRandomNum(int min, int max) : int
}

CellGrid "1" *-down- "*" Cell : owns
CellGrid "1" *-down- "1" GridSize : owns
WorldBoard "1" o-down- "1" CellGrid : has
WorldBoard "1" o-down- "1" DataParser : has
```

```
DocoMovePatternDefault .up.|> DocoMoveStrategy : implements
DocoMovePatternDiagonal .up.|> DocoMoveStrategy : implements
DocoMovePatternHorizontal .up.|> DocoMoveStrategy : implements
DocoMovePatternVertical .up.|> DocoMoveStrategy : implements
DocoMovePatternPerp .up.|> DocoMoveStrategy : implements

Doco .down.> DocoMoveStrategy : uses

AbstractSimpleDocoFactory <|.right. DocoFactory : implements
DocoFactory ..> Doco : creates
Doco "1" *-left- "1" Directions : owns
Doco .left.|> AbstractDoco : implements
WorldBoard .left.> AbstractDoco : uses
WorldBoard .left.> AbstractSimpleDocoFactory : uses

DocoMovePatternDiagonal

 --o DocoFactory : has
DocoMovePatternHorizontal --o DocoFactory : has
DocoMovePatternVertical --o DocoFactory : has
DocoMovePatternPerp --o DocoFactory : has
DocoMovePatternDefault --o DocoFactory : has


DocoMovePatternDiagonal o-down- Directions : has
DocoMovePatternHorizontal o-down- Directions : has
DocoMovePatternVertical o-down- Directions : has
DocoMovePatternPerp o-down- Directions : has
DocoMovePatternDefault o-down- Directions : has

UniformRandom --o Directions : has


main "1" o-down- "1" Simulator : has
main "1" o-down- "1" Viewer : has
main "1" o-down- "1" WorldBoard : has

@enduml
```

Appendix

Naming:

- Class Specific Variables: member variables, properties, attributes
- Class Specific Functions: member functions, methods, behaviors

Tools:

- https://www.planttext.com

Guides:

- https://plantuml.com/class-diagram
- http://ogom.github.io/draw_uml/plantuml