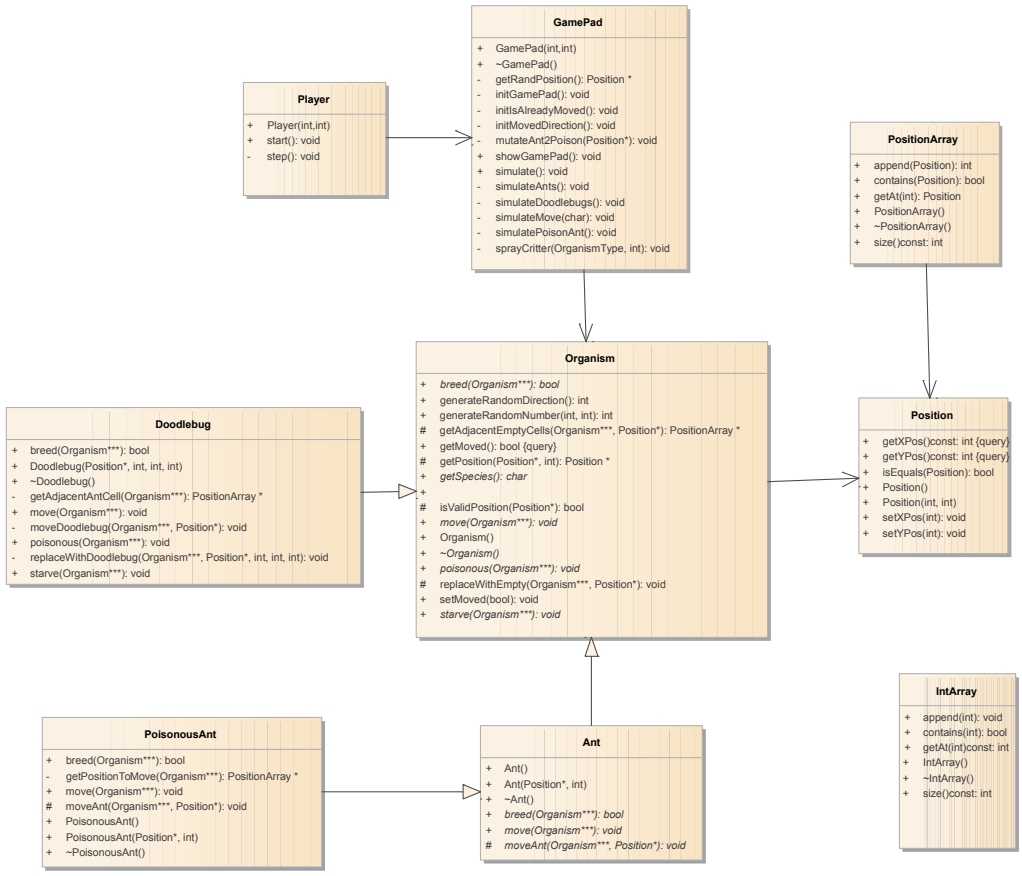
This project consist of 10 classes and main.cpp.

At the below, show the class diagram.



1. Player class

This is the class to start game by using GamePad class after getting the setting values.

-start() function

To show the game pad so that start the game.

And whenever clicked the enter key, call step() function so that go ahead the game.

-step() function

This function will be called whenever the enter key is clicked from start() function.

2. GamePad class

This is the class basic game pad class.

This class print game pad and call the functions of organisms so that simulate the game.

- getRandPosition () : Position * Private

@return : position.

This function generate random position according to the number of rows and columns and return it.

- initGamePad () : void Private

After to create game pad object, init that with parameters.

-sprayCritter (OrganismType type , int num) : void Private

@param type : the type of critter

@param num : the number of critter

Spray critters on the game pad.

-sprayCritter(OrganismType type, int num): void Private

@param type : the type of critter

@param num : the number of critter

The function to put critters to random positions on gamepad .

-showGamePad () : void Public

Show the game pad edge and the correct character for each critters on the game pad

-initIsAlreadyMoved () : void Private

Init the isMoved variable of critters on the game pad for the next move.

- mutateAnt2Poison (pos : Position*) : void Private

@param pos : position to be mutated

Mutate ant to poisonous ant.

- simulate () : void Public

Simulate the game with corresponding rules.

To simulate the game , this function call simulateAnts(), simulateDoodlebugs(), simulatePoisonAnt() functions.

- simulateAnts () : void Private

Simulate the behavior of ants

- simulateDoodlebugs () : void Private

Simulate the behavior of doodlebugs

-simulateMove (char species) : void Private

Move the critters according to species.

-simulatePoisonAnt () : void Private

Simulate the behavior of poisonous ants.

3. Organism class

This is abstract class for critters.

-breed (Organism world)***

@param world : game pad

@return bool: If success to breed ,return false.

This function is to breed for critter. This function will be overriding at child classes.

- generateRandomDirection ()

@return int

This function generates a random direction for indicating any direction

- generateRandomNumber (int start, int end)

@param start

@param end

@return int

Generate a random number from start to end.

- getAdjacentEmptyCells (Organism world, Position* pos)***

@param world : game pad

@param pos : position * : current position

@return PositionArray * : a list of valid position to adjacent empty cells

Return empty position .

- getMoved ()

Getter for isAlreadyMoved variable which check whether was moved or not.

@brief Organism::getMoved

@return bool

-setMoved (bool value)

@param value

Setter for isAlreadyMoved variable which check whether was moved or not.

- getPosition (Position* pos, int direction)

@brief Organism::getPosition

@param pos : current position

@param direction : direction to be moved.

@return Position * : new position

Get a new valid Position from grid according to the direction.

- getSpecies( )

@return char species

Getter for the species of current critter.

- isValidPosition (Position* pos)

@param pos : the position to be checked.

@return : true if checked is ok.

Check the position to be right. For example, whether is not edge.

- move (Organism world)***

@param world : game pad

Function for moving of Critter.

This will be overriding in the child classes.

- poisonous (Organism world)***

@param world : game pad.

The function for killing doodlebug after eating poisonous ant

- replaceWithEmpty (Organism world, Position* pos)***

@param world : game pad

@param pos : position to be replaced

Replace the critter with empty

- starve (Organism world)***

@param world : game pad

Only use for doodlebugs. If doodlebug not eat after three step, kill it.

4. Ant class

This is the child class derived from Organism class.

This is the class for ant behavior

-breed (Organism world)***

@param world : game board

@return bool : if success to breed, false. this is only used for mutation.

The function for the breed of ant. This is overriding breed() function of Organism class.

- move (Organism world)***

@param world : game board

The function for the moving of ant. This is overriding.

- moveAnt (Organism world, Position* pos)***

@param world : game pad.

@param pos: new postion to be moved

Replace the critter on the old position with new ant so that simulate the moving of ant.

5. PoisonousAnt class

This is the class derived from Organism class.

This will be used to simulate the behavior of poisonous ant.

-breed (Organism world)***

@param world : game board

@return bool : if success to breed, false. this is only used for mutation.

The function for the breed of PoisonousAnt. This is overriding.

- move (Organism world)***

@param world : game board

The function for the moving of poisonous ant. This is overriding.

- getPositionToMove (Organism world)***

@brief PoisonousAnt::getPositionToMove

@param world

@return position array to be possible.

Get position array to be moved.

-moveAnt (Organism world, Position* pos)***

@param world : game pad.

@param pos : new position to be moved.

Replace the critter on the old position with new ant so that simulate the moving of ant.

6. Doodlebug class

This is the class derived from Organism class.

This will be used to simulate the behavior of doodlebug.

-breed (Organism world)***

@param world : game pad

The function for breeding.

- move (Organism world)***

@param world : game pad

The function to make doodlebugs move with checking there are ants or not at adjacent cells.

- moveDoodlebug (Organism world, Position* pos)***

@param world : game pad

@param pos : position

Move a doodlebug to the valid position. If there is ant or poisonous ant, the doodlebug will eat that.

-starve (Organism world)***

@param world : game pad

The function for starve status. This checks if doodlebug has not eaten in 3 steps .

- getAdjacentAntCell (Organism* world)**

@brief Doodlebug::getAdjacentAntCell

@param world : game pad

@return PositionArray : the cells to be ant.

Get cells to be filled with ant.

- poisonous (Organism world)***

@brief Doodlebug::poisonous

@param world : game pad

The function for killing within 2 steps after eating poisonous ant.

- replaceWithDoodlebug (Organism world, Position* pos, int survived, int starve, int poisonAnt)***

@param world : game pad

@param pos : new position

@param survived : survived step

@param starve : starving step

@param poisonAnt : step after eat poisonous ant

Replace critter with doodlebug for moving.

7. Position class

This is the class to store position of critters.

- getXPos () const

The getter of current x position.

- setXPos (int value)

The setter of current x position.

- getYPos () const

The getter of current y position.

-setYPos(int value)

Setter of current y position.

- isEquals (Position pos)

@param pos : position to be checked

@return bool : if equals, true.

Check whether equals with this and pos.

8. PositionArray class

This is the class to store Position type variables as list.

This will be used to store the position history of critters.

- append (Position pos)

@brief PositionArray::append

@param pos : appended position

@return int : length

Append new value into array tail.

- contains (Position pos)

@brief PositionArray::contains

@param pos : input pos

@return bool : if contains, return true.

Check whether contains pos

- getAt (int index)

@brief PositionArray::getAt

@param index : from 0.

@return Position.

Get the value to be placed in index.

- size () const

@brief PositionArray::size

@return length

Return length of this.

9. IntArray class

This is the class to store int array as list.

This will be used to store the history of critter numbers.

- append (int value)

@param value

Append the value to the end of this.

- contains (int value)

@brief IntArray::contains

@param value

@return if contains, true.

Check whether contains the value.

- getAt (int position) const

@param position : the index of array

@return int.

Get the value of array with index.

- size () const

@return the size of array.

Get size of this.