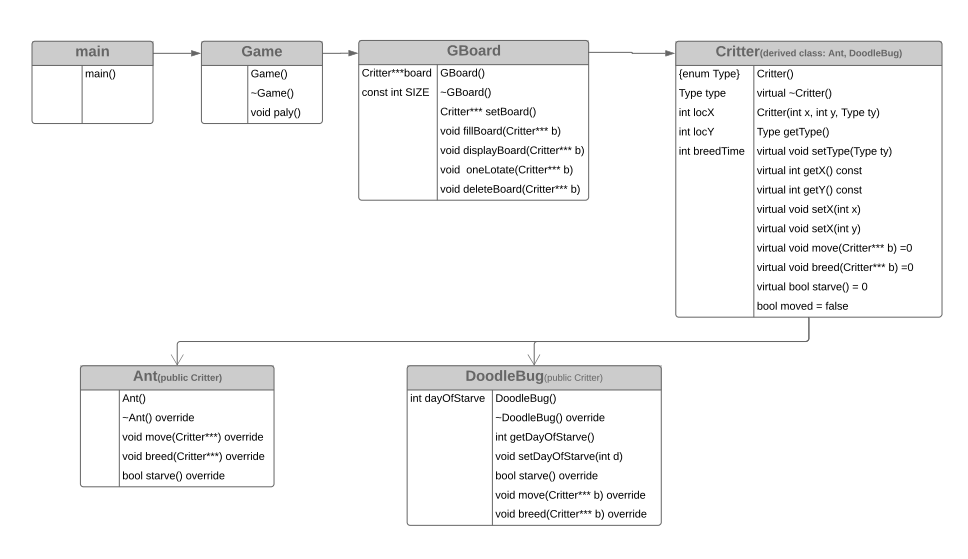
**Design, Reflection, Test Plan for Group Project**

Taekyoung Kim, Zuhair Ahmed

CS 162 2019 Winter

**DESIGN**

Below is the diagram of our group program.



Main Requirements:

* The board should be 2D array, and the cell of board points to a critter. That means we need to use “\*\*\*” Ant the board size is 20\*20.
* The doodle bugs should first move. Then ant move. After moving, they can breed.
* Critter class should be parent class, and ant and doodlebug classes are derived classes.
* Finding a cell for move and breed should be randomly.
* Doodle bugs are starved to death if they have not eaten for more than three days.
* Ants can breed when they have 3 turns (days), and doodle bugs can breed when 8 days.
* If there are no cells near the critters, they could not move or breed and wait for next turn.

For Critter:

* To detect each critter, the Critter class, parent class has getters and setters of coordinate of row and columns of the board. And also, it has another getter and setter for Type, enum.
* Other functions are all pure abstract so are not implemented in the Critter class.

<The bool moved() function is added later>

For Ant:

* It has two big parts: move and breed. As for the starve(), it does not have that one but still is needed to write on the header function.
* Ants needs to check if the cell is empty or not before moving. Also, it should be randomly. However, if we first pick a random number and that cell is occupied by another critter, it has to pick another random number again and again. So, our design should check the empty cell first then choose randomly one of those empty cells.
* The board cell and row between 0 and 19. So, we should be cautious not to cross the border.
* As for breed, the first work of this is also check if the cells near the ants is empty or not.

For Doodle Bug:

* Other parts are the same, but it must eat ant so the first thing that doodle bugs do is finding a cell that ants stay.
* Also, it has actual starve() function, and it check if the doodle bugs have not eaten ants for 3 days or more and delete the doodle bug.

<This starve() function first was void, but changed to bool later>

For Game Board class:

* We need to create board with 2d array of pointers using “\*\*\*” because each cell of this board are critters.
* Then, we need to fill this board with ants and doodle bugs using random number generator.
* Also, we need to create a function that prints the board with critters using ASCII characters.
* Inside the oneLotate function, the most important thing is if all events are in the right order. First doodle bugs should move then ant. And after all moving are done, the critters can breed.

For Game class:

* This class should be simple. It only asks the number of days and call the functions from the GBoard class and show them to users then has the feature of each continue and exit.

**REFLECTION**

1. When designing this program, bool moved function was not there. But when write the oneLotate function, we need something to check if critters moved or not. So, we set it in the critter class.
2. As for the starve() function, we first wrote it with void. Then, it was a little troublesome to handle it inside Ant class. So, later we changed it bool and made the return value of Ant’s starve() false.
3. When the program first showed the board, it was so exciting. However, it kept creating error. We tried many things but none of them was not working. Then, finally, we found the most important thing here is if every event is in the right order. We should first check if critters cross the border and then if the cell is empty or not using “nltptr.”
4. To be cautious of Typo!

The silliest mistakes that we made about the program is we input “X” in the place where should have been input “Y”. When we fixed those typos, the program just worked perfectly. Then, when we ran it on the school server, it showed 0 error and no memory leaks. Also need to be more careful confusing “=” which is not the same as “==” when testing for conditions in if/while statements

1. Starting Early. Taekyoung did a great job in organizing group together, booking group on Canvas, setting up Github and started on program well in advance of deadline. This allowed us to have adequate time to debug and write reflection document, as well as submit group project a few days early.

**Test Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input Values** | **Expected Outcomes** | **Actual Outcomes** |
| Positive / correct user data | 10 Days | Print out of various critters (both Ants + Doodlebugs) at random locations, all functions behaved correctly at each round (including breeding and dying), and menu prompted restart correctly | Print out of various critters (both Ants + Doodlebugs) at random locations, all functions behaved correctly at each round (including breeding and dying), and menu prompted restart correctly |
| Input extreme high data | 10000 Days | Print out of various critters (both Ants + Doodlebugs) at random locations, all functions behaved correctly at each round (including breeding and dying), and menu prompted restart correctly | Print out of various critters (both Ants + Doodlebugs) at random locations, all functions behaved correctly at each round (including breeding and dying), and menu prompted restart correctly |
| Input extreme low | 1 Day | Print out of various critters (both Ants + Doodlebugs) at random locations, all functions behaved correctly at each round (including breeding and dying), and menu prompted restart correctly | Print out of various critters (both Ants + Doodlebugs) at random locations, all functions behaved correctly at each round (including breeding and dying), and menu prompted restart correctly |
| Input invalid data | -5.5 Days | Kick out statements preventing action from input validation loops | Kick out statements preventing action from input validation loops |
| Input invalid data | Tiger Days | Kick out statements preventing action from input validation loops | Kick out statements preventing action from input validation loops |
| Input invalid data | 2k00l4skoolz Days | Kick out statements preventing action from input validation loops | Kick out statements preventing action from input validation loops |

**Work Distribution**

Taekyoung did the vast majority of the work for group project. She took the lead in writing and working with TA/Piazza on follow up questions she had on code / code requirements. She also did a great job on coaching Zuhair on complex concepts like pointers to 2D arrays while he continued to get up to speed on Linked Lists, Inheritance, and Polymorphism. Zuhair primarily focused on debugging and editing Reflection doc. The Work distribution was easily over 80% Taekyoung and 20% Zuhair.