Final Lab Aphids Vs Insects

By

Jorge Jurado-Garcia

EE2510 Sec. 021, Spring 2021

Week 9&10 lab

Milwaukee School of Engineering

Submitted to:

Professor: Joshua D. Carl, Ph.D

EECS Department

Date Report Submitted: 05/15/21

**Objective**

The objective of this lab is to create a simulated game that is like predator vs. prey. Using at least 2 of the 3 advance OOP ideas, operator overloading, composition, and inheritance/polymorphism.

**Description**

In my program I created 4 classes: Game, Insect(abstract), Aphids, and Ladybugs. In which the bugs live in a 20 X 20 grid of cells. Only 1 bug can occupy a cell at a time, and the adges are closed so the bugs cannot move off the edges. Time is simulated in time steps. Each bug performs 1 or more actions each time step. And the user can also pick a specific number of steps in between printing results. The Game function has a grid of Insect pointers and has the functions. To populate the grid with Insect\* set to nullptr. And then inserting the aphids and ladybugs. It also has a start function and timestep function as well. In which where breed, movement, and death occurs.

**Conclusions**

The lab was successful and was able to execute all the functions for Ladybugs and Aphids. Specifically, for the movement function which is what I had the most trouble with. For some reason when executing the code lags and stop and returns 00000c5 instead of 0. I believe the reason for this must be because some of the logic in my is going in a loop. Instead of doing it once. The biggest challenge for me during this lab was working out how this would be implemented and how to make it as proficient as possible. Specifically, the 2-d array without using global variables. This is where I created my game class for. Uses a class game that inputs Insect\* I was able to implement the necessary movement functions and keep track of them in my class instead of doing it in my main code. This allows all the methods and functions be done under the hood. But what I love about this lab was forcing to use pointers and the challenge of thinking how to create the hierarchy for all the classes. I also created a check function that returns a value if an aphid/insect was present. This was very useful when trying to implement the movement and breed functions. Something I would have done differently would be how I did the logic for my movement function and breed. I believe the breed function for both aphid and ladybugs are identical so I could have just implemented that function as a virtual instead of a pure virtual and call it once. This would reduce the amount of copy and paste I had to do and make the code shorter and cleaner. I also wish that I had for time to implement the food class for my aphids. For the food class I was thinking of deriving from Insect class and only have a generic constructor, int getInsect function, and death function.

Preliminary Text:

Game Class: This class is to map my abstract class pointers onto the grid and be able to move items in me from my grid class and if the object in my grid class is picked it can then move it onto another place. I am thinking of adding a function in which you can insert an insect\* pointer to a selected index [m][n] in my grid path. For Insect abstract class I created three variables called name. X coordinate, and Y coordinate. Were the insect path having two variables with virtual functions and pure virtual functions all point for this class is just to make the code easier to digest and to understand.

For Aphid class is it my prey it holds the proper constructors and have some operator handlers namely for food. My Ladybug class has an operator handling that takes in aphids and delete that aphid and increases its food tank variable by one. It also has different stats for it but other than that it mostly the same to aphids.

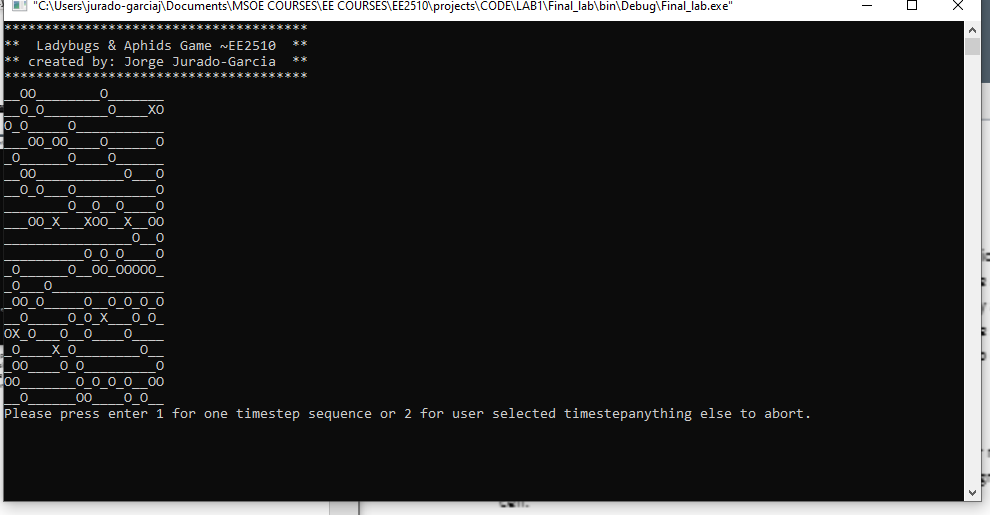
Aphids and Insects inherits form Insect abstract class. And Game class is base class for Insect abstract class.

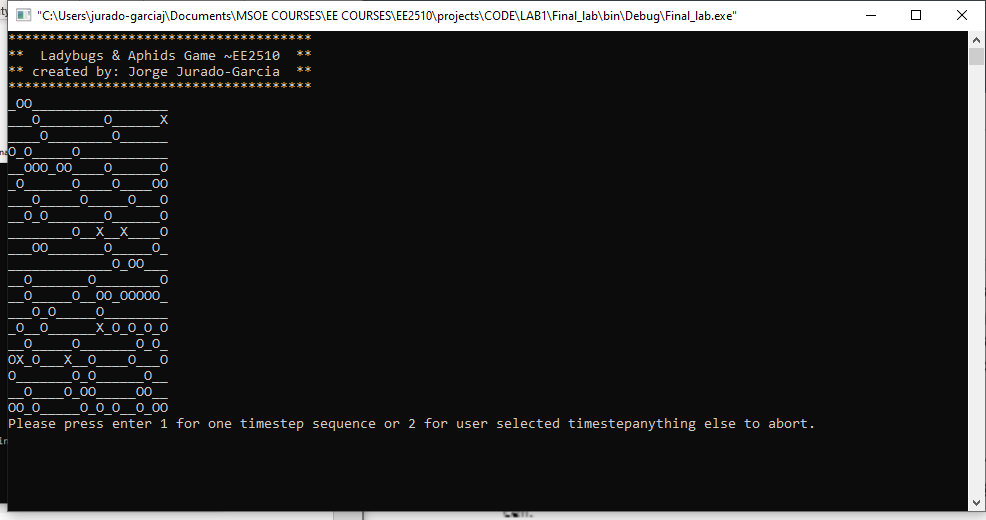
UML Diagram:

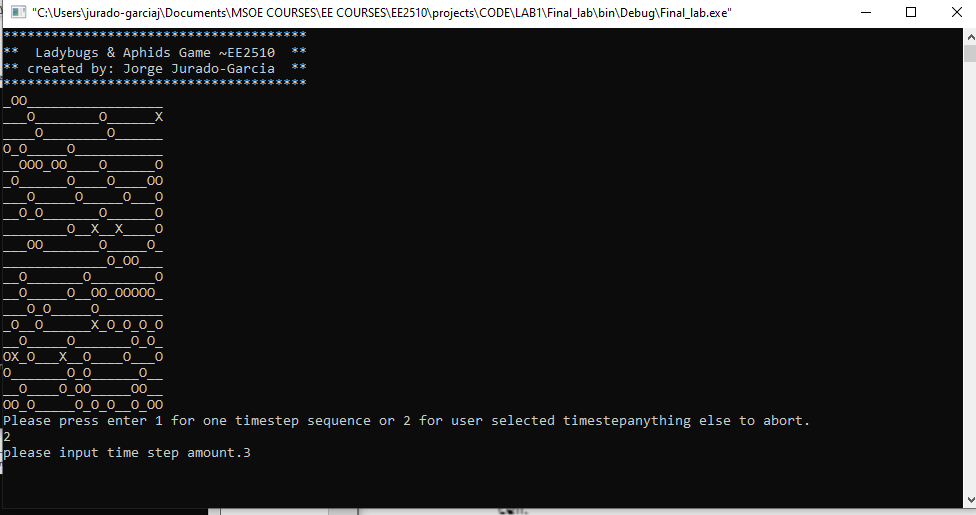
Diagram

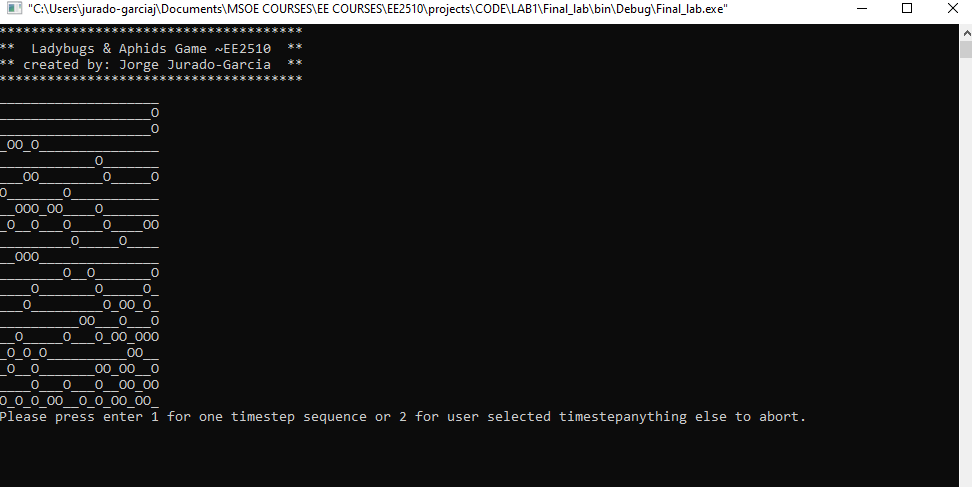
Description automatically generated

Console Result:

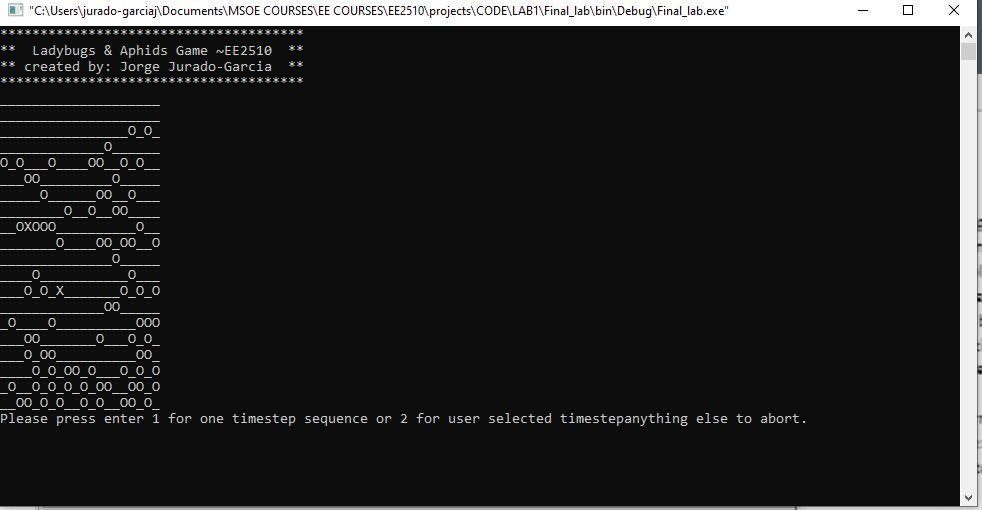


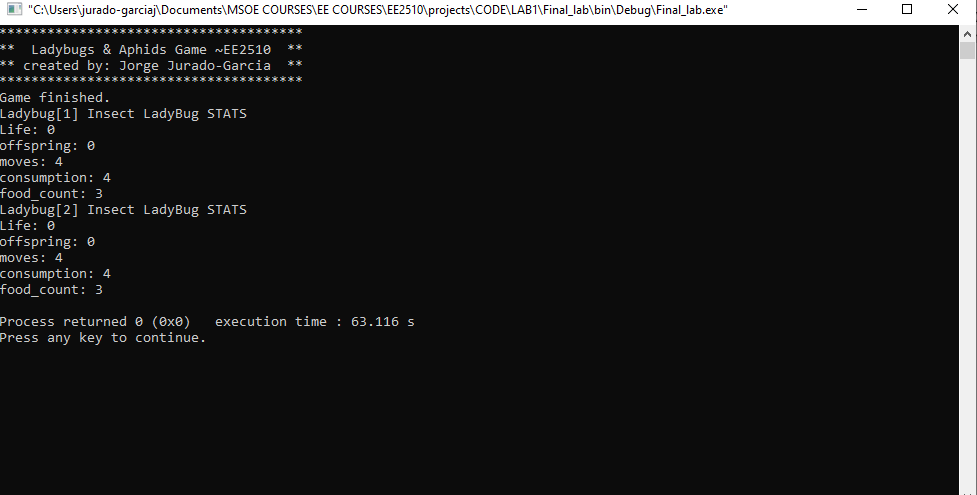






Abort function and stats of ladybug after four time steps.





Main File

#include <string>

#include <stdlib.h>

#include "Game.h"

#include "iostream"

**using** **namespace** std**;**

void intro**(){**

std**::**cout**<<**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**<<**std**::**endl**;**

std**::**cout**<<**"\*\* Ladybugs & Aphids Game ~EE2510 \*\*"**<<**std**::**endl**;**

std**::**cout**<<**"\*\* created by: Jorge Jurado-Garcia \*\*"**<<**std**::**endl**;**

std**::**cout**<<**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**<<**std**::**endl**;**

**}**

int main**()**

**{**

intro**();**

Game g1**;**

g1**.**startGame**();**

g1**.**printWorld**();**

**while(**1**){**

cout**<<**"Please press enter 1 for one timestep sequence or 2 for user selected timestep"**;**

cout**<<**"anything else to abort."**<<**endl**;**

int user\_input**;**

cin**>>**user\_input**;**

**if(**user\_input **==** 1**)**

**{**

g1**.**takeTimeStep**();**

system**(**"CLS"**);**

intro**();**

g1**.**printWorld**();**

**}**

**else** **if(**user\_input **==** 2**)**

**{**

int num**;**

cout**<<**"please input time step amount."**;**

cin**>>**num**;**

cout**<<**endl**;**

cout**<<**"timestep amount: "**<<**num**<<**endl**;**

**for(**int i**=**0**;** i**<**num**;** i**++)**

**{**

g1**.**takeTimeStep**();**

**}**

system**(**"CLS"**);**

intro**();**

g1**.**printWorld**();**

**}**

**else**

**{**

system**(**"CLS"**);**

intro**();**

cout**<<**"Game finished."**<<**endl**;**

g1**.**printstats**();**

**return** 0**;**

**}** //end elss

**}** //end while

**}** //end main

Game Header File

#ifndef GAME\_H\_INCLUDED

#define GAME\_H\_INCLUDED

#include "Insect.h"

class Insect;

class Game

{

private:

int generateRandomNumber(int startRange, int endRange) const; //random number generator

public:

Game(); //generic

Game(const Game& g1); //copy constructor

// ~Game(); //desctructor

int timeStepCount; //counts for my time step in my public function startgame

void startGame(); //starts and populates my grid

void takeTimeStep(); //next game

void printstats() const;

void printWorld() const; //prints the grid

//insector pointer from main that creates a grid of 20 x 20;

Insect\* grid[20][20];

};

#endif // GAME\_H\_INCLUDED

Game source file

#include <iostream>

#include <sstream>

#include <time.h> //time

#include <stdlib.h> //srand and rand

#include <string>

#include "Game.h"

#include "Insect.h"

#include "Ladybug.h"

#include "Aphids.h"

Game**::**Game**(){**

srand**(**time**(NULL));** //random seed genrator

timeStepCount **=** 0**;** //game is started so rimestep is 0

//creates a 20 by 20 array of null pointers

**for** **(**int x **=** 0**;** x **<** 20**;** x**++)**

**for** **(**int y **=** 0**;** y **<** 20**;** y**++)**

grid**[**x**][**y**]** **=** **nullptr;**

**}**

Game**::**Game**(**const Game**&** g1**)**

**{**

timeStepCount **=** g1**.**timeStepCount**;**

**for** **(**int x **=** 0**;** x **<** 20**;** x**++){**

**for** **(**int y **=** 0**;** y **<** 20**;** y**++){**

grid**[**x**][**y**]** **=** g1**.**grid**[**x**][**y**];**

**}**

**}**

**}**

void Game**::**startGame**(){**

int x**,**y**;** //x and y coordiniates for grid

int aphidcount **=** 0**;**

int ladybugcount **=** 0**;**

//while function ladybug\_count is set to 20

**while** **(**ladybugcount **<** 10**){**

//uses game genrate function to create a random number

x **=** generateRandomNumber**(**0**,** 20 **-** 1**);**

y **=** generateRandomNumber**(**0**,** 20 **-** 1**);**

//example: grid[1][1] = ladybug or aphid then thats it

//if not populate it with a ladybug

**if** **(**grid**[**x**][**y**]** **==** **nullptr)**

**{**

grid**[**x**][**y**]** **=** **new** Ladybug**(this,** x**,** y**);**

ladybugcount**++;**

**}**

**}**

**while** **(**aphidcount **<** 100**){**

x **=** generateRandomNumber**(**0**,** 20 **-** 1**);**

y **=** generateRandomNumber**(**0**,** 20 **-** 1**);**

**if** **(**grid**[**x**][**y**]** **==** **nullptr){**

//example: grid[1][1] = ladybug or aphid then thats it

//if not populate it with a aphid

grid**[**x**][**y**]** **=** **new** Aphid**(this,** x**,** y**);**

aphidcount**++;**

**}**

**}**

**}**

void Game**::**takeTimeStep**(){**

timeStepCount**++;**

**for** **(**int x **=** 0**;** x **<** 20**;** x**++){**

**for** **(**int y **=** 0**;** y **<** 20**;** y**++){**

**if(** grid**[**x**][**y**]** **==** **nullptr)**

**{**

**}**//end if

**else** **if** **(**grid**[**x**][**y**]->**getInsect**()** **==** 2**)** //ladybug

**{**

grid**[**x**][**y**]->**movement**();**

std**::**cout**<<**"ladybug movement function done\n"**;**

**}**//end else if

**else** **if(**grid**[**x**][**y**]->**getInsect**()** **==** 1**)**

**{**

grid**[**x**][**y**]->**movement**();**

std**::**cout**<<**"Aphid movement function done\n"**;**

**}** //end else if

**else{**

std**::**cout**<<**"N"**;**

**}** //end else

**}**//end for

**}**

std**::**cout**<<**"lady bug done"**;**

**for** **(**int x **=** 0**;** x **<** 20**;** x**++){**

**for** **(**int y **=** 0**;** y **<** 20**;** y**++){**

**if(** grid**[**x**][**y**]** **==** **nullptr)**

**{**

**}**//end if

**else** **if** **(**grid**[**x**][**y**]->**getInsect**()** **==** 2**)** //ladybug

**{**

grid**[**x**][**y**]->**breed**();** //this function works

std**::**cout**<<**"ladybug breed done\n"**;**

grid**[**x**][**y**]->**death**();** //this function works

std**::**cout**<<**"ladybug death done\n"**;**

**}**//end else if

**else** **if(**grid**[**x**][**y**]->**getInsect**()** **==** 1**)**

**{**

grid**[**x**][**y**]->**breed**();** //this function worls

std**::**cout**<<**"aphid breed done\n"**;**

grid**[**x**][**y**]->**death**();** //this function works

std**::**cout**<<**"aphid death done\n"**;**

**}** //end else if

**else{**

std**::**cout**<<**"N"**;**

**}** //end else

**}**//end for

**}**

std**::**cout**<<**"aphid done"**;**

**}**//end program

void Game**::**printWorld**()** const

**{**

**for** **(**int x **=** 0**;** x **<** 20**;** x**++){**

**for** **(**int y **=** 0**;** y **<** 20**;** y**++){**

**if** **(**grid**[**x**][**y**]** **==** **nullptr)**

std**::**cout **<<** '\_'**;**

**else** **if** **(**grid**[**x**][**y**]->**getInsect**()** **==** 1**)**

std**::**cout **<<** 'O'**;**

**else** //world[x][y]->getType() == LADYBUG

std**::** cout **<<** 'X'**;**

**}**

std**::**cout **<<** std**::**endl**;**

**}**

**}**

void Game**::** printstats**()** const

**{**

int num**=**1**;**

**for** **(**int x **=** 0**;** x **<** 20**;** x**++){**

**for** **(**int y **=** 0**;** y **<** 20**;** y**++){**

//grid is a insect\* so in order to check what inside

// i used getInsect and if its object of ladybug then

//it will return a value of 2;

**if(** **(**grid**[**x**][**y**]** **==** **nullptr)** **)**

**{**

**}**

**else**

**{**

**if(** **(**grid**[**x**][**y**]->**getInsect**())** **==** 2**)**

**{**

std**::**string result**;**

result **=** grid**[**x**][**y**]->**stats**();**

std**::**cout**<<**"Ladybug["**<<**num**<<**"]"**<<**result**;**

num**++;**

**}**

**}**

**}**

**}**

**}**

int Game**::**generateRandomNumber**(**int startRange**,** int endRange**)** const

**{**

**return** rand**()** **%** **(**endRange **-** startRange **+** 1**)** **+** startRange**;**

**}**

Insect Header File

/\*

File created by: Jorge Jurado-Garcia

Name: insect.h file

Abstract class

Date: 4/30/21

Modifications:

5/9 changing pure virtual functions to virtual

and deleting get/set function and creating data memembers

as protected instead of private

All rights reserved

Educational purposes

\*/

#ifndef INSECT\_H\_INCLUDED

#define INSECT\_H\_INCLUDED

#include "Game.h"

class Game;

class Insect

{

public:

//generic

Insect();

//parameter constructor

Insect(Game\* curgame, int X\_coord, int Y\_coord);

//THIS IS WILL BE NAME, X-LOCATION, AND Y-LOCATION

//pure virtual functions makes insects abstract base class

virtual int check(int x, int y)=0; //pure virtual

virtual int getInsect() = 0 ; //pure virtual

virtual std::string stats(); //pure virtual;

virtual void breed()=0;

virtual void movement();

virtual void death()=0;

Game\* current\_game;

//game\* for current\_game address of the address of gameptr

int X\_coord;

int Y\_coord;

protected:

int timestepcount;

int breed\_count;

int food\_count;

int life; //life expedency

int offspring; //production

int moves; //moves

int consumption; //ladybugs eaten

//end of class

};

#endif // INSECT\_H\_INCLUDED

Insect Source File

/\*

File created by: Jorge Jurado-Garcia

Name: insect.ccp file

Abstract class

Date: 4/30/21

Modifications:

All rights reserved

Educational purposes

\*/

#include <sstream>

#include <iostream>

#include "Insect.h"

//generic constructor

Insect**::** Insect**()**

**{**

current\_game **=** **nullptr;**

X\_coord **=** 0**;**

Y\_coord **=** 0**;**

timestepcount **=** 0**;**

breed\_count **=** 0**;**

food\_count **=** 0**;**

life **=** 0**;** //life expedency

offspring **=** 0**;** //production

moves **=** 0**;** //moves

consumption **=** 0**;** //ladybugs eaten

**}**

Insect**::** Insect**(**Game**\*** curgame**,** int X\_coord**,** int Y\_coord**)**

**{**

**this->**current\_game **=** curgame**;**

**this->**X\_coord **=** X\_coord**;**

**this->**Y\_coord **=** Y\_coord**;**

//timestepcount copies the timestepcount from class game

//since current\_game is a ptr to game game\*

breed\_count **=** 0**;**

food\_count **=** 0**;**

life **=** 0**;** //life expedency

offspring **=** 0**;** //production

moves **=** 0**;** //moves

consumption **=** 0**;** //ladybugs eaten

timestepcount **=** curgame**->**timeStepCount**;**

**}**

std**::**string Insect**::** stats**()**

**{**

std**::**string ret**;**

ret **=** " Insect "**;**

**return** ret**;**

**}**

void Insect**::** movement**()**

**{**

**if(**timestepcount**==**current\_game**->**timeStepCount**)**

**{**

**return;**

**}**

timestepcount**++;**

**}**

Aphid Header file

/\*

File created by: Jorge Jurado-Garcia

Name: Aphids.h file

Abstract class

Date: 4/30/21

Modifications:

All rights reserved

Educational purposes

\*/

#ifndef APHIDS\_H\_INCLUDED

#define APHIDS\_H\_INCLUDED

#include "Insect.h"

class Aphid**:**

public Insect

**{**

public**:**

//generic constructor

Aphid**();**

//parameter constructor

Aphid**(**Game**\*** curgame**,** int X\_coord**,** int Y\_coord**);**

int get\_x**();**

int get\_y**();**

//virtual functions to override

virtual void death**();**

virtual int check**(**int x**,** int y**);**

virtual void movement**();**

virtual void breed**();** //breeds and returns a insect

virtual int getInsect**();** //return an aphid

virtual std**::**string stats**();**

**};**

#endif // APHIDS\_H\_INCLUDED

Aphid Source File

/\*

File created by: Jorge Jurado-Garcia

Name: Aphids.ccp file

Abstract class

Date: 4/30/21

Modifications:

All rights reserved

Educational purposes

\*/

#include <iostream>

#include <sstream>

#include <time.h>

#include <stdlib.h>

#include <string>

#include "Aphids.h"

//generic constructor

Aphid**::** Aphid**()**

**{**

food\_count **=** 3**;**

breed\_count **=** 3**;**

**}**

//parameter constructor

Aphid**::**Aphid**(**Game**\*** curgame**,** int X\_coord**,** int Y\_coord**)**

**:**Insect**(**curgame**,** X\_coord**,** Y\_coord**)**

**{**

breed\_count **=** 3**;**

food\_count **=** 3**;**

**}**

//death of aphid

void Aphid**::** death**()**

**{**

**if(**food\_count**==**0**){**

current\_game**->**grid**[**X\_coord**][**Y\_coord**]** **=** **nullptr;**

**}**

**return;**

**}**

//breed function

void Aphid**::** breed**()**

**{**

std**::**cout**<<**"in breed function\n"**;**

**if(**breed\_count **!=** 0**){**

std**::**cout**<<**"breed count is not zero\n"**;**

**return;**

**}**

std**::**cout**<<**"breed count is zero"**;**

int ar**[**4**];**

ar**[**0**]** **=** check**(**X\_coord**+**1**,**Y\_coord**);** //down

ar**[**1**]** **=** check**(**X\_coord**,**Y\_coord**+**1**);** //right

ar**[**2**]** **=** check**(**X\_coord**-**1**,**Y\_coord**);** //up

ar**[**3**]** **=** check**(**X\_coord**,**Y\_coord**-**1**);** //left

int New\_X\_coord**;**

int New\_Y\_coord**;**

**for(**int j **=** 0**;** j**<**4**;**j**++)**

**{**

int sum **=** ar**[**j**]+** sum**;**

std**::**cout**<<**ar**[**j**];**

**if(**sum**==**0**){**

**return;**

**}**

**}**

**for(**int i**=**0**;**i**<**4**;**i**++){**

**if(**i**==**0**){**

New\_X\_coord **=** X\_coord**+**1**;**

New\_Y\_coord **=** Y\_coord**;**

**}**

**else** **if(**i**==**1**){**

New\_X\_coord **=** X\_coord**;**

New\_Y\_coord **=** Y\_coord**+**1**;**

**}**

**else** **if(**i**==**2**){**

New\_X\_coord **=** X\_coord**-**1**;**

New\_Y\_coord **=** Y\_coord**;**

**}**

**else{**

New\_X\_coord **=** X\_coord**;**

New\_Y\_coord **=** Y\_coord**-**1**;**

**}**

**if(** ar**[**i**]** **==** 1 **){**

//make the current position to a null ptr

offspring**++;**

current\_game**->**grid**[**X\_coord**][**Y\_coord**]** **=** **this;**

current\_game**->**grid**[**New\_X\_coord**][**New\_Y\_coord**]** **=** **nullptr;**

std**::**cout**<<**"passed this\n"**;**

current\_game**->**grid**[**New\_X\_coord**][**New\_Y\_coord**]** **=** **new** Aphid**(**current\_game**,**New\_X\_coord**,**New\_Y\_coord**);**

std**::**cout**<<**"beed count accomplushed"**;**

i **=** 5**;**

**}**

**}**//end for

**}** //end function

//getInsect function

int Aphid**::** getInsect**()**

**{**

**return** 1**;**

**}**

int Aphid**::** get\_x**()**

**{**

**return** X\_coord**;**

**}**

int Aphid**::** get\_y**()**

**{**

**return** Y\_coord**;**

**}**

//stats function

std**::**string Aphid**::** stats**()**

**{**

std**::**string num**;**

num **=** " "**;**

**return** num**;**

**}**

//movement function

void Aphid**::** movement**()**

**{**

**if(**timestepcount**==**current\_game**->**timeStepCount**)**

**{**

**return;**

**}**

timestepcount**++;**

std**::**cout**<<**X\_coord**<<**" "**<<**Y\_coord**<<**"\n"**;**

int ar**[**4**];**

ar**[**0**]** **=** check**(**X\_coord**+**1**,**Y\_coord**);** //down

ar**[**1**]** **=** check**(**X\_coord**,**Y\_coord**+**1**);** //right

ar**[**2**]** **=** check**(**X\_coord**-**1**,**Y\_coord**);** //up

ar**[**3**]** **=** check**(**X\_coord**,**Y\_coord**-**1**);** //left

int New\_X\_coord**;**

int New\_Y\_coord**;**

int sum**=**0**;**

**for(**int i **=** 0**;** i**<**4**;**i**++)**

**{**

**for(**int j**=**0**;**j**<**4**;**j**++){**

sum **=** ar**[**j**]+** sum**;**

std**::**cout**<<**"ar["**<<**j**<<**"]"**<<**ar**[**j**]<<**"\n"**;**

**}**

std**::**cout**<<**"sum: "**<<**sum**<<**"\n"**;**

**if(**sum**==**0**){**

**return;**

**}**

**if(**i**==**0**){**

New\_X\_coord **=** X\_coord**+**1**;**

New\_Y\_coord **=** Y\_coord**;**

std**::**cout**<<**New\_X\_coord**<<**" "**<<**New\_Y\_coord**<<**"\n"**;**

**}**

**else** **if(**i**==**1**){**

New\_X\_coord **=** X\_coord**;**

New\_Y\_coord **=** Y\_coord**+**1**;**

std**::**cout**<<**New\_X\_coord**<<**" "**<<**New\_Y\_coord**<<**"\n"**;**

**}**

**else** **if(**i**==**2**){**

New\_X\_coord **=** X\_coord**-**1**;**

New\_Y\_coord **=** Y\_coord**;**

std**::**cout**<<**New\_X\_coord**<<**" "**<<**New\_Y\_coord**<<**"\n"**;**

**}**

**else{**

New\_X\_coord **=** X\_coord**;**

New\_Y\_coord **=** Y\_coord**-**1**;**

std**::**cout**<<**New\_X\_coord**<<**" "**<<**New\_Y\_coord**<<**"\n"**;**

**}**

**if(** ar**[**i**]** **==** 1 **){**

std**::**cout**<<**New\_X\_coord**<<**" " **<<**New\_Y\_coord**<<**"\n"**;**

current\_game**->**grid**[**New\_X\_coord**][**New\_Y\_coord**]** **=** **nullptr;**

current\_game**->**grid**[**X\_coord**][**Y\_coord**]** **=** **nullptr;**

std**::**cout**<<**"passed all info\n"**;**

X\_coord **=** New\_X\_coord**;**

Y\_coord **=** New\_Y\_coord**;**

current\_game**->**grid**[**New\_X\_coord**][**New\_Y\_coord**]** **=** **this;**

std**::**cout**<<**"passed all info\n"**;**

i **=** 4**;** //stop the for loop

std**::**cout**<<**"j is:"**<<**i**<<**"\n"**;**

**}**//end if statement

**}**//end for

std**::**cout**<<**"done\n"**;**

**}** //end movement

//check function

int Aphid**::** check**(**int x**,** int y**)**

**{**

//this is to check if the user who imported x value between 0-20

//bool out is a way to check what will happen if if its outofscope

//then the check function is null

int out**;**

**if(** **(**x**<**0 **||** **(**y**<**0**))** **)**

**{**

out **=** 0**;**

**return** out**;**

**}**

**if(** **(**x**>**19**)** **||** **(**y**>**19**)** **)**

**{**

out **=** 0**;**

**return** out**;**

**}**

**if(**current\_game**->**grid**[**x**][**y**]** **==** **nullptr)**

**{**

out **=** 1**;**

**}**

**else** **if(**current\_game**->**grid**[**x**][**y**]->**getInsect**()** **==** 2**)** //ladybug

**{**

out **=** 0**;**

**}**

**else** //aphid

**{**

out **=** 0**;**

**}**

**return** out**;**

**}**

Ladybug header file

/\*

File created by: Jorge Jurado-Garcia

Name: LadyBugs.h file

Abstract class

Date: 5/02/21

Modifications:

All rights reserved

Educational purposes

\*/

#ifndef LADYBUG\_H\_INCLUDED

#define LADYBUG\_H\_INCLUDED

#include "Insect.h"

#include "Aphids.h"

class Ladybug**:**

public Insect

**{**

public**:**

//generic constructor

Ladybug**();**

//parameter constructor

Ladybug**(**Game**\*** current\_game**,** int X\_coord**,** int Y\_coord**);**

//stats function

virtual std**::** string stats**();**

//virtual bool check(int x, int y);

//virtual functions to override

virtual void movement**();**

virtual int check**(**int x**,** int y**);**

virtual void death**();**

virtual void breed**();** //breeds and returns a insect

virtual int getInsect**();** //return an insect

**};**

#endif // LADYBUG\_H\_INCLUDED

Ladybug Source File

/\*

File created by: Jorge Jurado-Garcia

Name: ladybug.ccp file

Date: 5/02/21

Modifications:

All rights reserved

Educational purposes

\*/

#include <iostream>

#include <sstream>

#include <string>

#include "Ladybug.h"

//generic constructor

Ladybug:: Ladybug()

{

food\_count = 3;

life = 0;

offspring = 0;

moves = 0;

consumption = 0;

breed\_count = 8;

}

//parameter constructor

Ladybug::Ladybug(Game\* current\_game, int X\_coord, int Y\_coord)

:Insect(current\_game,X\_coord,Y\_coord)

{

breed\_count = 8;

food\_count = 3;

life = 0;

offspring = 0;

moves = 0;

consumption = 0;

}

//death of aphid

void Ladybug:: death()

{

if(food\_count<=0){

current\_game->grid[X\_coord][Y\_coord] = nullptr;

}

return;

}

//breed function

void Ladybug:: breed()

{

std::cout<<"in breed function\n";

if(breed\_count != 0){

std::cout<<"breed count is not zero\n";

return;

}

if(breed\_count != 0){

return;

}

int ar[4];

ar[0] = check(X\_coord+1,Y\_coord); //down

ar[1] = check(X\_coord,Y\_coord+1); //right

ar[2] = check(X\_coord-1,Y\_coord); //up

ar[3] = check(X\_coord,Y\_coord-1); //left

int New\_X\_coord;

int New\_Y\_coord;

for(int j = 0; j<4;j++)

{

int sum = ar[j]+ sum;

if(sum==0){

return;

}

}

for(int i=0;i<4;i++){

if(i==0){

New\_X\_coord = X\_coord+1;

New\_Y\_coord = Y\_coord;

}

else if(i==1){

New\_X\_coord = X\_coord;

New\_Y\_coord = Y\_coord+1;

}

else if(i==2){

New\_X\_coord = X\_coord-1;

New\_Y\_coord = Y\_coord;

}

else{

New\_X\_coord = X\_coord;

New\_Y\_coord = Y\_coord-1;

}

if( ar[i] == 1 ){

//make the current position to a null ptr

offspring++;

current\_game->grid[X\_coord][Y\_coord] = this;

current\_game->grid[New\_X\_coord][New\_Y\_coord] = nullptr;

std::cout<<"passed this\n";

current\_game->grid[New\_X\_coord][New\_Y\_coord] = new Ladybug(current\_game,New\_X\_coord,New\_Y\_coord);

std::cout<<"beed count accomplushed";

i = 5;

}

}//end for

}

int Ladybug:: getInsect()

{

return 2;

}

//stats function

std:: string Ladybug:: stats()

{

std::string ret;

std::ostringstream out1;

std::ostringstream out2;

std::ostringstream out3;

std::ostringstream out4;

std::ostringstream out5;

out1<<life;

out2<<offspring;

out3<<moves;

out4<<consumption;

out5<<food\_count;

ret = Insect::stats();

ret = ret + "LadyBug STATS\n";

ret = ret + "Life: " + out1.str() + "\n";

ret = ret + "offspring: " + out2.str() + "\n";

ret = ret + "moves: " + out3.str() + "\n";

ret = ret + "consumption: " + out4.str() + "\n";

ret = ret + "food\_count: " + out5.str() + "\n";

return ret;

}

void Ladybug:: movement()

{

if(timestepcount == current\_game->timeStepCount)

{

return;

}

timestepcount++;

std::cout<<X\_coord<<" "<<Y\_coord<<std::endl;

int ar[4];

ar[0] = check(X\_coord+1,Y\_coord); //down

ar[1] = check(X\_coord,Y\_coord+1); //right

ar[2] = check(X\_coord-1,Y\_coord); //up

ar[3] = check(X\_coord,Y\_coord-1); //left

int New\_X\_coord;

int New\_Y\_coord;

int sum=0;

int pp=0; //best choice of action

for(int j=0;j<4;j++){

sum = ar[j]+ sum;

if(ar[j] == 2){

if( j == 0 ){

j = 4;

}

pp = j;

}

std::cout<<"ar["<<j<<"]"<<ar[j]<<"\n";

} //end for

std::cout<<"sum: "<<sum<<"\n";

if(sum==0){

return;

}

if(pp == 0)

{

for(int i = 0; i<4;i++)

{

if(i==0){

New\_X\_coord = X\_coord+1;

New\_Y\_coord = Y\_coord;

std::cout<<New\_X\_coord<<" "<<New\_Y\_coord<<"\n";

}

else if(i==1){

New\_X\_coord = X\_coord;

New\_Y\_coord = Y\_coord+1;

std::cout<<New\_X\_coord<<" "<<New\_Y\_coord<<"\n";

}

else if(i==2){

New\_X\_coord = X\_coord-1;

New\_Y\_coord = Y\_coord;

std::cout<<New\_X\_coord<<" "<<New\_Y\_coord<<"\n";

}

else{

New\_X\_coord = X\_coord;

New\_Y\_coord = Y\_coord-1;

std::cout<<New\_X\_coord<<" "<<New\_Y\_coord<<"\n";

}

if( ar[i] == 1 ){ //aphid in its place

//make the current position to a null ptr

std::cout<<X\_coord<<" "<<Y\_coord<<std::endl;

current\_game->grid[New\_X\_coord][New\_Y\_coord] = nullptr;

current\_game->grid[X\_coord][Y\_coord] = nullptr;

current\_game->grid[New\_X\_coord][New\_Y\_coord] = this;

X\_coord = New\_X\_coord;

Y\_coord = New\_Y\_coord;

moves++;

food\_count--;

i == 4; //stop for loop

}//end if

}// end for

}//end if

else

{

if(pp==4){

New\_X\_coord = X\_coord+1;

New\_Y\_coord = Y\_coord;

std::cout<<New\_X\_coord<<" "<<New\_Y\_coord<<"\n";

}

else if(pp==1){

New\_X\_coord = X\_coord;

New\_Y\_coord = Y\_coord+1;

std::cout<<New\_X\_coord<<" "<<New\_Y\_coord<<"\n";

}

else if(pp==2){

New\_X\_coord = X\_coord-1;

New\_Y\_coord = Y\_coord;

std::cout<<New\_X\_coord<<" "<<New\_Y\_coord<<"\n";

}

else{

New\_X\_coord = X\_coord;

New\_Y\_coord = Y\_coord-1;

std::cout<<New\_X\_coord<<" "<<New\_Y\_coord<<"\n";

}//end of else statement

//make the current position to a null ptr

std::cout<<X\_coord<<" "<<Y\_coord<<std::endl;

current\_game->grid[New\_X\_coord][New\_Y\_coord] = nullptr;

current\_game->grid[X\_coord][Y\_coord] = nullptr;

current\_game->grid[New\_X\_coord][New\_Y\_coord] = this;

X\_coord = New\_X\_coord;

Y\_coord = New\_Y\_coord;

moves++;

consumption++;

}//end else

}//end function

//check function

int Ladybug:: check(int x, int y)

{

int out;

//this is to check if the user who imported x value between 0-20

//bool out is a way to check what will happen if if its outofscope

//then the check function is null

if( (x<0 || (y<0)) )

{

out = 0;

return out;

}

if( (x>19) || (y>19) )

{

out = 0;

return out;

}

if(current\_game->grid[x][y] == nullptr)

{

std::cout<<"null pointer\n";

out = 1;

}

else if(current\_game->grid[x][y]->getInsect() == 2)

{

std::cout<<"ladbug found\n";

out = 0;

}

else{

std::cout<<"aphid found pointer\n";

out = 2;

}

return out;

}