CSE 241 – OBJECT ORIENTED PROGRAMMING

PROJECT

AZİZ CAN AKKAYA 1801042250

Contents

1 Intorduction

1.1 Briefing

Document Publised Date	: 11/07/2020
Project Verison	: 3.1v
Video Documentation :	https://youtu.be/VJ9uDZ7ElIs

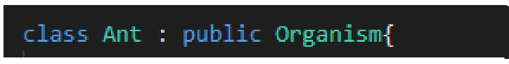
1.2 Objective

The objective of this project is to use every skill that learned and got a understanding of about object oriented programming. Esspecially on the topics of polymorphism and linked list concepts.

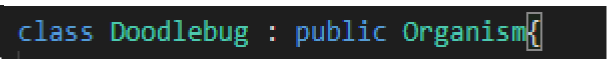
1.2.1 Project Tasks

The task is to build a 2D ASCII world and move random moves on organisms based on their aspects. It is a predator vs prey based of movement. The second is to build a header structure for this class of structure.

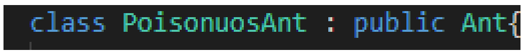
2 Technical Aspects



```
class Ant : public Organism{
```

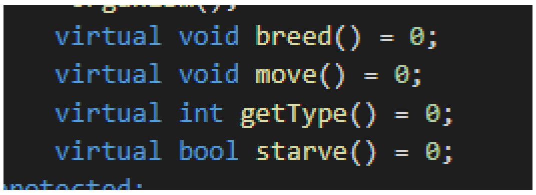


```
class Doodlebug : public Organism{
```



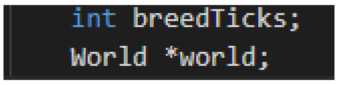
```
class PoisonuosAnt : public Ant{
```

I used a polymorphic class structure connect these 4 classes since they have similar characterisctics. It helped me to use the functionality of these functions easly and set them up efficently.

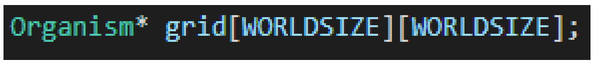


```
    virtual void breed() = 0;
    virtual void move() = 0;
    virtual int getType() = 0;
    virtual bool starve() = 0;
```

Making these functions pure virtual under the concept of polymorphism is allowed me to manipulate the data inside the organism according to data inside subclasses.



```
    int breedTicks;
    World *world;
```



```
Organism* grid[WORLDSIZE][WORLDSIZE];
```

Linking these two classes and friending them allowed me to reach eacher others functions and datas and make an outcome based on their datas and functions.

3 Conclusion

Although the program causes segmentation fault when it is forced to take a step from function, virtual functions is tested and working well individuallly. The main thing that causes segmentation fault is the instability of pointers.

On final notes this project helped us to use every skill that we learned and pushed us to our boundiries to see where we are. It is a good practice fort he most important subjects of object oriented programming.

4 Appendix

ORGANISM.H

```
#ifndef ORGANISM_H
#define ORGANISM_H

#include <iostream>
#include <string>
#include <vector>
#include <cstdlib>
#include <time.h>

class World;

class Organism{
    friend class World;
    public:
        Organism();
        Organism(World *world, int x, int y);
        ~Organism();
        virtual void breed() = 0;
        virtual void move() = 0;
        virtual int getType() = 0;
        virtual bool starve() = 0;
    protected:
        int x,y;
        bool moved;
        int breedTicks;
        World *world;
};

#endif
```

DOODLEBUG.H

```
#ifndef DOODLEBUG_H
#define DOODLEBUG_H

#include <iostream>
#include "world.h"
#include "organism.h"

class Doodlebug : public Organism{
    friend class World;
    public:
        Doodlebug();
        Doodlebug(World *world, int x, int y);
        void breed(); // Must define this since virtual
        void move(); // Must define this since virtual
        int getType(); // Must define this since virtual
        bool starve(); // Check if a doodlebug starves to death
        bool poisended();
    private:
        int starveTicks; // Number of moves before starving
        int poisonTimer;
};

#endif
```

P_ANT.H

```
#ifndef P_ANT_H
#define P_ANT_H

#include <iostream>
#include "world.h"
#include "organism.h"
#include "ant.h"

class PoisonuosAnt : public Ant{
    friend class World;
    friend class Organism;
    public:
        PoisonuosAnt();
        PoisonuosAnt(World *world, int x, int y);
        void breed();
        int getType();
};

#endif
```

WORLD.H

```cpp
#ifndef WORLD_H
#define WORLD_H

#include "organism.h"

const int WORLDSIZE = 20;
const int INITIALANTS = 100;
const int INITIALBUGS = 5;
const int DOODLEBUG = 1;
const int ANT = 2;
const int P_ANT = 3;
const int ANTBREED = 3;
const int DOODLEBREED = 8;
const int DOODLESTARVE = 3;
const int P_ANTBREED = 4;

class World{

    friend class Organism;
    friend class Doodlebug;
    friend class Ant;
    friend class PoisonuosAnt;
    public:
        World();
        ~World();
        Organism* getAt(int x, int y);
        void setAt(int x, int y, Organism *org);
        void Display();
        void Step();
        void Simulate();
    protected:
        Organism* grid[WORLDSIZE][WORLDSIZE];
};

#endif
```

ANT.H

```
#ifndef ANT_H
#define ANT_H

#include <iostream>
#include "world.h"
#include "organism.h"

class Ant : public Organism{
    friend class World;
    public:
        Ant();
        Ant(World *world, int x, int y);
        void breed(); // Must define this since virtual
        void move(); // Must define this since virtual
        int getType(); // Must define this since virtual
        bool starve(){ return false; }
};


#endif
```

MAIN.CPP

```
#include "organism.h"
#include "world.h"
#include "p_ant.h"
#include "ant.h"
#include "doodlebug.h"
#include <iostream>

using namespace std;

int main(){
    World w;
    srand(time(NULL)); // Seed random number generatorWorld w;
    int step = 0;

    int antcount = 0;
    while (antcount < INITIALANTS){
        int x = rand() % WORLDSIZE;
        int y = rand() % WORLDSIZE;
        if (w.getAt(x,y)==NULL){
            antcount++;
            Ant *a1 = new Ant(&w,x,y);
        }
    }
// Randomly create 5 doodlebugs
    int doodlecount = 0;
    while (doodlecount < INITIALBUGS){
        int x = rand() % WORLDSIZE;
        int y = rand() % WORLDSIZE;
        if (w.getAt(x,y)==NULL){
            doodlecount++;
            Doodlebug *d1 = new Doodlebug(&w,x,y);
        }
    }

    w.Display();

    //w.Simulate();

    while( true ){
        string in;
        step++;

        cout << "before simulate" << endl;
        //w.Step();
        cout << "simulate" << endl;
        w.Display();
        cout << "Steps Simulated : " << step << endl
        << "Press Any Key and Enter to contunie : ";
        cin >> in;
        system("clear");
    }

    return 0;
}
```

WORLD.CPP

```
#include "world.h"
#include "organism.h"
#include <iostream>
#include <string>
#include <vector>
#include <cstdlib>
#include <time.h>

using namespace std;

World::World(){
    int i,j;
    for (i=0; i<WORLDSIZE; i++){
        for (j=0; j<WORLDSIZE; j++){
            grid[i][j]=NULL;
        }
    }
```

```cpp
}
World::~World(){
    int i,j;
    for (i=0; i<WORLDSIZE; i++){
        for (j=0; j<WORLDSIZE; j++){
            if (grid[i][j]!=NULL)
                delete (grid[i][j]);
        }
    }
}

Organism* World::getAt(int x, int y){
    if ((x>=0) && (x<WORLDSIZE) && (y>=0) && (y<WORLDSIZE))
        return grid[x][y];
    return NULL;
}

void World::setAt(int x, int y, Organism *org){
    if ((x>=0) && (x<WORLDSIZE) && (y>=0) && (y<WORLDSIZE)){
        grid[x][y] = org;
    }
}

void World::Display(){
    int i,j;
    cout << endl << endl;
    for (j=0; j<WORLDSIZE; j++){
        for (i=0; i<WORLDSIZE; i++){
            if (grid[i][j]==NULL)
                cout << "_";
            else if (grid[i][j]->getType()==ANT)
                cout << "o";
            else if( grid[i][j]->getType() == P_ANT)
                cout << "c";
            else
                cout << "X";
        }
    cout << endl;
    }
}

void World::Step(){
    for (int x = 0; x < WORLDSIZE; x++){
    for (int y = 0; y < WORLDSIZE; y++){
      if (grid[x][y] == nullptr) continue;
      if (grid[x][y]->getType() == DOODLEBUG)
        grid[x][y]->move();
    }
  }
  for (int x = 0; x < WORLDSIZE; x++){
    for (int y = 0; y < WORLDSIZE; y++){
      if (grid[x][y] == nullptr) continue;
      if (grid[x][y]->getType() == ANT)
        grid[x][y]->move();
    }
  }

  for (int x = 0; x < WORLDSIZE; x++){
    for (int y = 0; y < WORLDSIZE; y++){
      if (grid[x][y] == nullptr) continue;
      if (grid[x][y]->getType() == P_ANT)
        grid[x][y]->move();
    }
  }

  for (int x = 0; x < WORLDSIZE; x++){
    for (int y = 0; y < WORLDSIZE; y++){
      if (grid[x][y] == nullptr) continue;
      grid[x][y]->breed();
    }
  }

  for (int x = 0; x < WORLDSIZE; x++){
    for (int y = 0; y < WORLDSIZE; y++){
      if (grid[x][y] == nullptr) continue;
      if (grid[x][y]->starve()){
        delete grid[x][y];
        grid[x][y] = nullptr;
      }
    }
  }
}

void World::Simulate(){

    int step = 0;
    int doodle = 0;
    int p_ant = 0;
    int ant = 0;

    while(1){
        string in;
        step++;

        for(int i = 0; i < WORLDSIZE; i++){
            for (int j = 0 ; j < WORLDSIZE; j++){
                if(grid[i][j]->getType() == DOODLEBUG)
```

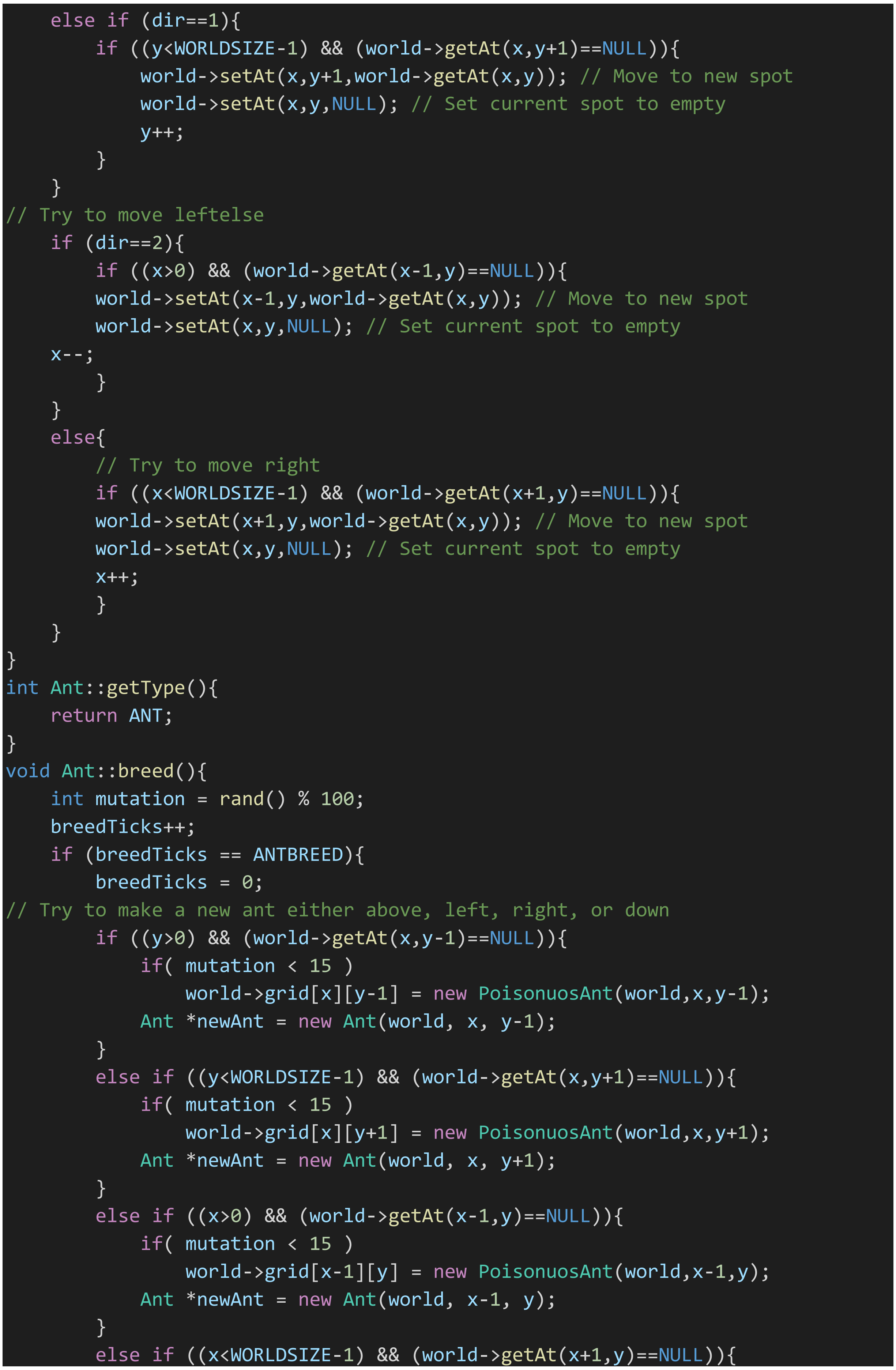
```
                    doodle++;
                else if(grid[i][j]->getType() == ANT)
                    ant++;
                else if(grid[i][j]->getType() == P_ANT)
                    p_ant;
            }
        }
        Step();
        Display();
        cout << "Steps Simulated : " << step << endl
        << "Doodlebugs : " << doodle << endl
        << "Ants : " << ant << endl
        << "Poisonus Ants : " << p_ant << endl << endl
        << "Press Enter to contunie : ";
        cin >> in;
        system("clear");
    }
}
```

ANT.CPP

```
#include <iostream>
#include "world.h"
#include "organism.h"
#include "ant.h"
#include "p_ant.h"

using namespace std;

Ant::Ant() : Organism(){
    //empty
}
Ant::Ant(World *world, int x, int y) : Organism(world,x,y){
    //empty
}
void Ant::move(){
    // Pick random direction to move
    int dir = rand() % 4;
    // Try to move up, if not at an edge and empty spot
    if (dir==0){
        if ((y>0) && (world->getAt(x,y-1)==NULL)){
            world->setAt(x,y-1,world->getAt(x,y)); // Move to new spot
            world->setAt(x,y,NULL);
            y--;
        }
    }
// Try to move down
```



```
    else if (dir==1){
        if ((y<WORLDSIZE-1) && (world->getAt(x,y+1)==NULL)){
            world->setAt(x,y+1,world->getAt(x,y)); // Move to new spot
            world->setAt(x,y,NULL); // Set current spot to empty
            y++;
        }
    }
// Try to move leftelse
    if (dir==2){
        if ((x>0) && (world->getAt(x-1,y)==NULL)){
        world->setAt(x-1,y,world->getAt(x,y)); // Move to new spot
        world->setAt(x,y,NULL); // Set current spot to empty
    x--;
        }
    }
    else{
        // Try to move right
        if ((x<WORLDSIZE-1) && (world->getAt(x+1,y)==NULL)){
        world->setAt(x+1,y,world->getAt(x,y)); // Move to new spot
        world->setAt(x,y,NULL); // Set current spot to empty
        x++;
        }
    }
}
int Ant::getType(){
    return ANT;
}
void Ant::breed(){
    int mutation = rand() % 100;
    breedTicks++;
    if (breedTicks == ANTBREED){
        breedTicks = 0;
// Try to make a new ant either above, left, right, or down
        if ((y>0) && (world->getAt(x,y-1)==NULL)){
            if( mutation < 15 )
                world->grid[x][y-1] = new PoisonuosAnt(world,x,y-1);
            Ant *newAnt = new Ant(world, x, y-1);
        }
        else if ((y<WORLDSIZE-1) && (world->getAt(x,y+1)==NULL)){
            if( mutation < 15 )
                world->grid[x][y+1] = new PoisonuosAnt(world,x,y+1);
            Ant *newAnt = new Ant(world, x, y+1);
        }
        else if ((x>0) && (world->getAt(x-1,y)==NULL)){
            if( mutation < 15 )
                world->grid[x-1][y] = new PoisonuosAnt(world,x-1,y);
            Ant *newAnt = new Ant(world, x-1, y);
        }
        else if ((x<WORLDSIZE-1) && (world->getAt(x+1,y)==NULL)){
            if( mutation < 15 )
                world->grid[x+1][y] = new PoisonuosAnt(world,x+1,y);
            Ant *newAnt = new Ant(world, x+1, y);
        }
    }
}
```

DOODLEBUG.CPP

```
#include <iostream>
#include "world.h"
#include "organism.h"
#include "doodlebug.h"

using namespace std;

Doodlebug::Doodlebug() : Organism(){
    starveTicks = 0;
    poisonTimer = 0;
}
Doodlebug::Doodlebug(World *world, int x, int y) : Organism(world,x,y){
    starveTicks = 0;
    poisonTimer = 0;
}

bool Doodlebug::poisended(){
    if( poisonTimer == 2 )
        return true;
    return false;
}

void Doodlebug::move(){
// Look in each direction and if a bug is found move there
// and eat it.
    if ((y>0) && (world->getAt(x,y-1)!=NULL) && (world->getAt(x,y-1)-
>getType() == ANT || world->getAt(x,y-1)->getType() == P_ANT ) ){
        if( world->getAt(x,y-1)->getType() == P_ANT)
            poisonTimer++;
        delete (world->grid[x][y-1]); // Kill ant
        world->grid[x][y-1] = this; // Move to spot
        world->setAt(x,y,NULL);
        starveTicks =0 ; // Reset hunger
        y--;
        return;
    }
    else if ((y<WORLDSIZE-1) && (world->getAt(x,y+1)!=NULL) && ( world-
>getAt(x,y+1)->getType() == ANT || world->getAt(x,y+1)-
>getType() == P_ANT )  ){
        if( world->getAt(x,y+1)->getType() == P_ANT)
            poisonTimer++;
        delete (world->grid[x][y+1]); // Kill ant
        world->grid[x][y+1] = this; // Move to spot
        world->setAt(x,y,NULL);
        starveTicks =0 ; // Reset hunger
        y++;
        return;
    }
    else if ((x>0) && (world->getAt(x-1,y)!=NULL) && (world->getAt(x-1,y)-
>getType() == ANT) || world->getAt(x-1,y)->getType() == P_ANT){
        if( world->getAt(x-1,y)->getType() == P_ANT)
            poisonTimer++;
        delete (world->grid[x-1][y]); // Kill ant
        world->grid[x-1][y] = this; // Move to spot
        world->setAt(x,y,NULL);
        starveTicks =0 ; // Reset hunger
        x--;
        return;
    }
    else if ((x<WORLDSIZE-1) && (world->getAt(x+1,y)!=NULL) && (world-
>getAt(x+1,y)->getType() == ANT || world->getAt(x+1,y)->getType() == P_ANT)){
        if( world->getAt(x+1,y)->getType() == P_ANT)
            poisonTimer++;
        delete (world->grid[x+1][y]); // Kill ant
        world->grid[x+1][y] = this; // Move to spot
        world->setAt(x,y,NULL);
        starveTicks =0 ; // Reset hunger
        x++;
        return;
    }
        // If we got here, then we didn't find food. Move
        // to a random spot if we can find one.
    int dir = rand() % 4;
    // Try to move up, if not at an edge and empty spot
    if (dir==0){
        if ((y>0) && (world->getAt(x,y-1)==NULL)){
            world->setAt(x,y-1,world->getAt(x,y)); // Move to new spot
            world->setAt(x,y,NULL);
            y--;
        }
    }
// Try to move down
    else if (dir==1){
        if ((y<WORLDSIZE-1) && (world->getAt(x,y+1)==NULL)){
            world->setAt(x,y+1,world->getAt(x,y)); // Move to new spotworld-
>setAt(x,y,NULL); // Set current spot to empty
            y++;
        }
    }
// Try to move left
    else if (dir==2){
        if ((x>0) && (world->getAt(x-1,y)==NULL)){
            world->setAt(x-1,y,world->getAt(x,y)); // Move to new spot
        world->setAt(x,y,NULL); // Set current spot to empty
        x--;
        }
    }
// Try to move right
    else{
        if ((x<WORLDSIZE-1) && (world->getAt(x+1,y)==NULL)){
            world->setAt(x+1,y,world->getAt(x,y)); // Move to new spot
            world->setAt(x,y,NULL); // Set current spot to empty
            x++;
        }
    }
    starveTicks++; // Increment starve tick since we didn't
// eat on this turn
}
int Doodlebug::getType(){
    return DOODLEBUG;
}
void Doodlebug::breed(){
    breedTicks++;
    if(breedTicks == DOODLEBREED){
        breedTicks = 0;
// Try to make a new ant either above, left, right, or down
        if ((y>0) && (world->getAt(x,y-1)==NULL)){
            Doodlebug *newDoodle = new Doodlebug(world, x, y-1);
        }
        else if ((y<WORLDSIZE-1) && (world->getAt(x,y+1)==NULL)){
            Doodlebug *newDoodle = new Doodlebug(world, x, y+1);
        }
        else if ((x>0) && (world->getAt(x-1,y)==NULL)){
            Doodlebug *newDoodle = new Doodlebug(world, x-1, y);
        }
        else if ((x<WORLDSIZE-1) && (world->getAt(x+1,y)==NULL)){
            Doodlebug *newDoodle = new Doodlebug(world, x+1, y);
        }
    }
}
bool Doodlebug::starve(){
// Starve if no food eaten in last DOODLESTARVE time ticks
    if (starveTicks > DOODLESTARVE){
        return true;
    }
    else{
        return poisended();
    }
}
```

ORGANISM.CPP

```
#include <iostream>
#include "organism.h"
#include "world.h"
#include <string>
#include <vector>
#include <cstdlib>
#include <time.h>

using namespace std;


Organism::Organism(){
    world = NULL;
    moved = false;
    breedTicks = 0;
    x=0;
    y=0;
}
Organism::Organism(World *wrld, int x, int y){
    this->world = wrld;
    moved = false;
    breedTicks = 0;
    this->x=x;
    this->y=y;
    wrld->setAt(x,y,this);
}

Organism::~Organism(){
    //empty
}
```

P_ANT.CPP

```
#include <iostream>
#include "world.h"
#include "organism.h"
#include "ant.h"
#include "p_ant.h"

using namespace std;

PoisonuosAnt::PoisonuosAnt() : Ant(){
    //empty
```

```cpp
}

PoisonuosAnt::PoisonuosAnt(World *world, int x, int y) : Ant(world,x,y){
    //empty
}

int PoisonuosAnt::getType(){
    return P_ANT;
}

void PoisonuosAnt::breed(){
    bool bred = false;
    breedTicks++;
    if (breedTicks == ANTBREED){
        breedTicks = 0;
// Try to make a new ant either above, left, right, or down
        if ((y>0) && (world->getAt(x,y-1)==NULL)){
            PoisonuosAnt *newAnt = new PoisonuosAnt(world,x,y-1);
            bred = true;
        }
        else if ((y<WORLDSIZE-1) && (world->getAt(x,y+1)==NULL)){
            PoisonuosAnt *newAnt = new PoisonuosAnt(world,x,y+1);
            bred = true;
        }
        else if ((x>0) && (world->getAt(x-1,y)==NULL)){
            PoisonuosAnt *newAnt = new PoisonuosAnt(world,x-1,y);
            bred = true;
        }
        else if ((x<WORLDSIZE-1) && (world->getAt(x+1,y)==NULL)){
            PoisonuosAnt *newAnt = new PoisonuosAnt(world,x+1,y);
            bred = true;
        }

        if(bred == false){
            if ((y>0) && world->grid[x][y-1]->getType() == ANT ){
                PoisonuosAnt *newAnt = new PoisonuosAnt(world,x,y-1);
                bred = true;
            }
            else if ((y<WORLDSIZE-1) && world->grid[x][y+1]-
>getType() == ANT ){
                PoisonuosAnt *newAnt = new PoisonuosAnt(world,x,y+1);
                bred = true;
            }
            else if ((x>0) && world->grid[x-1][y]->getType() == ANT ){
                PoisonuosAnt *newAnt = new PoisonuosAnt(world,x-1,y);
                bred = true;
            }
            else if ((x<WORLDSIZE-1) && world->grid[x+1][y]-
>getType() == ANT ){
```

```
                PoisonuosAnt *newAnt = new PoisonuosAnt(world,x+1,y);
                bred = true;
            }
        }
    }
}
```