Current checklist:

* + Implement Organism.Act() function
  + Implement Reproduce() so it creates a new child object that will be implemented with a future Age() function.
  + Implement the mutate method to target a specific(Random) gene, or sequence of genes in the “chromosome” object (Chromosome is going to be an ArrayList<Genes> )
  + Implement a mini-display when the mouse goes over a “Organism object” *(Probably going to be inside the GUI class)*

|  |
| --- |
| Below is the area for our code, and any updates to the code. Please indicate any changes by adding a comment to the area(s) in question. |

*this is the most recent constructor for an Allele object. all imports have been taken out of the signature for neatness*

public class Allele implements Gene

{

int num, id;

double doub;

String name,allelename ;

public Allele()

{

this.name = "Default";

this.id = 0;

this.num = 0;

this.doub = 0.0;

this.allelename = "DefaultAllele";

}

}

|  |
| --- |
| Updated Code - moving Organism - No flowers/attempting to check adjacent organisms |

private Location getNextLocation() {

Location loc = getLocation();

Location next = loc.getAdjacentLocation(getDirection());

return next;

}

private boolean getID(){

return this.id;

}

public ArrayList<Organism> getNeighbours(Location lctn){

ArrayList<Organism> list = new ArrayList<>();

for (int i = 0; i < direction.length; i++){

Location adj = getLocation().getAdjacentLocation(this.direction[i]);

if(getGrid().isValid(adj) == true){

continue;

}

else{

list.add(); //Add the object at the location adj ← incomplete

System.out.print("Organism at " + this.direction[i]);

}

}

return list;

}

/\*public ArrayList<Organism> getNeighbours(Location lctn){

ArrayList<Organism> list = new ArrayList<>();

for (int i = 0; i < 8; i++){

Location adj = getLocation().getAdjacentLocation(this.direction[i]);

Organism tempOrg = (Organism)getGrid().get(adj);

if (tempOrg.getID() == false){

System.out.print("Object is not organism");

}

else if (tempOrg.getID() == true){

System.out.print("Object is an organism");

list.add(tempOrg);

}

}

return list;

}\*/

public String getGender() {

return this.gender;

}

@Override

public void move() {

Location next = getNextLocation();

moveTo(next);

}

@Override

public void act() {

ArrayList<Organism> list = this.getNeighbours(this.getLocation());

for (int i = 0; i < list.size(); i++) {

if (getGender().equalsIgnoreCase(list.get(i).getGender())) {

System.out.println(list.get(i).name);

} else {

System.out.print("Different Gender");

}

}

if (canMove()) {

move();

} else {

turn();

}

}

}

|  |
| --- |
| Updated code for checking to see if any neighbouring Objects are Organisms using the ***instanceof*** call |

public static ArrayList<Organism> getNeighbourOrganismsFromArrayList(ArrayList<Object> arrList)

{

ArrayList<Organism> orgList = new ArrayList<>();

for(int n = 0; n < arrList.size(); n++)

{

Object testOrg = arrList.get(n);

try{

if(testOrg instanceof Organism)

{

System.out.println("Organism at index "+n+"\nNow adding to list");

orgList.add((Organism) testOrg);

}

else{

System.out.println("Not an organism at index "+n );

}

}

catch(Exception E)

{

System.out.println("Not an organism");

}

}

return orgList;

}

public ArrayList<Object> getNeighbours(Location loc) {

ArrayList<Object> list = new ArrayList<>();

for (int i = 0; i < direction.length; i++) {

Location adj = getLocation().getAdjacentLocation(this.direction[i]);

if (getGrid().isValid(adj) == true) {

System.out.println("IsValid");

list.add(getGrid().get(adj)); //Add the object at the location adj

System.out.println("Organism " + this.name + " direction checked " + this.direction[i]);

} else {

System.out.println("Location " + this.direction[i] + " is not Valid");

}

}

return list;

}

|  |
| --- |
| Updated organism Object Constructor (May 23rd 2013) |

public Organism() {

int ranName = (int) (Math.random() \* names.length);

int ranGender = (int) (Math.random() \* genders.length);

int ranDiet = (int) (Math.random() \* diets.length);

int ranDirec = (int) (Math.random() \* direction.length);

this.name = names[ranName];

this.genotype = "";

this.diet = diets[ranDiet];

this.dietLocked = false;

this.gender = genders[ranGender];

this.genderLocked = false;

this.genes = null;

this.setDirection(ranDirec);

}

|  |
| --- |
| Block of code included at the end of **act** method. (requires **info.gridworld.grid.Grid** import)  This removes Organism object after a certain # of steps and places a rock in its position.  Also include in the specified locations:  - **public int age;** ← Instance variable  - **this.age = (int) (Math.random() \* 300);**  ← In Organism constructor (300 can be changed)  - **sb.append(("Steps Before Death: " + this.age + "\n"));** ← In the getInfo method |

this.age--;

if (this.age == 0){

Location loc = this.getLocation();

Grid<Actor> grid = this.getGrid();

Rock grave = new Rock();

this.removeSelfFromGrid();

grave.putSelfInGrid(grid, loc);

}

Here is the gif file , just drop it into simulation after creating a Grave class, which extends rock, and only has one constructor: public Grave(){}

|  |
| --- |
| Reproduction Code **[Outdated]** |

public static Color color[] = {Color.BLACK, Color.BLUE, Color.CYAN, Color.DARK\_GRAY, Color.GRAY, Color.GREEN, Color.LIGHT\_GRAY, Color.MAGENTA, Color.ORANGE, Color.PINK, Color.RED, Color.WHITE, Color.YELLOW};

private void reproduce(Organism parent, Organism parent\_2) {

Organism offspring = new Organism();

Location loc;

int rndmColor = (int) (Math.random() \* this.color.length);

if (parent.getGender() == "female"){

loc = parent.getNextLocation();

} else{

loc = parent\_2.getNextLocation();

}

Grid<Actor> grid = parent.getGrid();

offspring.setColor(color[rndmColor]);

offspring.putSelfInGrid(grid, loc);

}

--------------------------------------------------------------------------------------------------------

**ACT METHOD**

ArrayList<Organism> list = this.getNeighbourOrganismsFromArrayList(getNeighbours(this.getLocation()));

for (int i = 0; i < list.size(); i++) {

if (getGender().equalsIgnoreCase(list.get(i).getGender())) {

//System.out.println(this.name + " has same gender as " + list.get(i).name);

list.remove(i);

} else {

//System.out.println(this.name + " has different gender than " + list.get(i).name);

continue;

}

}

if (list.size() >= 1) {

int rndmIndx = (int) (Math.random() \* list.size());

reproduce(this, list.get(rndmIndx));

this.reproducing = true;

}

// Random move

double rndmMove = Math.random();

if (rndmMove < 0.75 && !reproducing) {

if (canMove()) {

move();

} else {

turn();

}

} else {

randomTurn();

}

this.reproducing = false;

|  |
| --- |
| Updated Code of Organism class |

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package biology.Organisms;

import biology.simulation.BiologySimulation;

import biology.simulation.GUI;

import biology.simulation.Grave;

import info.gridworld.actor.Actor;

import info.gridworld.actor.Bug;

import info.gridworld.grid.Grid;

import info.gridworld.grid.Location;

import java.awt.Color;

import java.util.ArrayList;

import java.util.Arrays;

/\*\*

\*

\* @author Will

\*/

public class Organism extends Bug{

public String genotype, gender, name, diet, testGene;

public int cipher, death, key, generation;

private boolean id, reproducing, oneChild;

public int x, y, age, lagPeriod, maxSteps;

public boolean isCarnivorous, isVegetarian, isOmnivorous, genderLocked, dietLocked;

public static String [] testAlleles = {"Aa", "Ii", "Nn"};

private ArrayList<Organism> genes = new ArrayList<>();

public static String[] diets = {"Omnivorous", "Vegetarian", "Carnivorous"}, names = {"Alouette", "Beta", "Gamma", "Rho"}, genders = {"Female", "Male"};

public static int[] direction = {Location.NORTH, Location.EAST,Location.SOUTH,Location.WEST,Location.NORTHEAST, Location.NORTHWEST, Location.SOUTHEAST, Location.SOUTHWEST};

public static Color color[] = {Color.BLACK, Color.BLUE, Color.CYAN, Color.DARK\_GRAY, Color.GRAY, Color.GREEN, Color.LIGHT\_GRAY, Color.MAGENTA, Color.ORANGE, Color.PINK, Color.WHITE, Color.YELLOW, Color.RED};

public Organism() {

int ranName = (int) (Math.random() \* names.length);

int ranGender = (int) (Math.random() \* genders.length);

int ranDiet = (int) (Math.random() \* diets.length);

int ranDirec = (int) (Math.random() \* direction.length);

this.age = 1;

this.lagPeriod = 0;

this.name = names[ranName];

this.genotype = "";

this.diet = diets[ranDiet];

this.dietLocked = false;

this.gender = genders[ranGender];

this.genderLocked = false;

this.genes = null;

this.reproducing = false;

this.generation = 1;

this.oneChild = false;

this.maxSteps = 300;

this.death = this.maxSteps;

this.setDirection(ranDirec);

this.testGene = "";

BiologySimulation.orgCount++;

}

public static ArrayList newMetric\_TonOfOrganisms() {

int ran = (int) (Math.random() \* 50);

ArrayList<Organism> metric = new ArrayList<>();

for (int n = 0; n < ran; n++) {

Organism org = new Organism();

int ranDiet = (int) (Math.random() \* diets.length);

int ranName = (int) (Math.random() \* names.length);

int ranGender = (int) (Math.random() \* genders.length);

int ranDirec = (int) (Math.random() \* direction.length);

org.setName(names[ranName] + " " + n);

org.setGender(genders[ranGender]);

org.setDiet(diets[ranDiet]);

org.setDirection(direction[ranDirec]);

metric.add(org);

org.id = true;

}

return metric;

}

public static ArrayList newMetricTonOfOrganisms(int num) {

ArrayList<Organism> metric = new ArrayList<>();

Arrays.sort(names);

for (int n = 0; n < num; n++) {

Organism org = new Organism();

int ranDiet = (int) (Math.random() \* diets.length);

int ranName = (int) (Math.random() \* names.length);

int ranGender = (int) (Math.round(Math.random()));

int ranDirec = (int) (Math.random() \* direction.length);

org.setRandomGenoType();

org.setName(names[ranName] + " " + n);

org.setGender(genders[ranGender]);

org.setDiet(diets[ranDiet]);

org.setDirection(direction[ranDirec]);

metric.add(org);

}

return metric;

}

public void setName(String name) {

this.name = name;

}

public void mutate()//implement later

{

int ran = (int) (Math.random());

}

public void setGender(String gender) {

if (this.genderLocked != false) {

System.out.println("Gender is Locked");

} else {

String temp = this.gender;

temp.toLowerCase();

if (temp.equalsIgnoreCase("female") || temp.equalsIgnoreCase("male")) {

System.out.println("Not a valid name. Try again");

} else {

this.gender = gender;

this.genderLocked = true;

System.out.println("Gender is not locked, locking " + this.name + "'s gender now now");

}

}

}

public String setGender\_returnString(String gender) {

String ret;

if (this.genderLocked != false) {

ret = ("Gender is Locked");

} else {

String temp = this.gender;

temp.toLowerCase();

if (temp.equalsIgnoreCase("female") || temp.equalsIgnoreCase("male")) {

ret = ("Not a valid name. Try again");

} else {

this.gender = gender;

this.genderLocked = true;

ret = ("Gender is not locked, locking " + this.name + "'s gender now now");

}

}

return ret;

}

public void setDiet(String diet) {

for (int i = 0; i < diets.length; i++) {

if (diet.equalsIgnoreCase((diets[i]))) {

this.diet = diets[i];

}

}

}

private static double[] pundit(String allele\_1, String allele\_2)//omit for now

{

int p1 = allele\_1.length(), p2 = allele\_2.length();

String[][] alleles = new String[p2][p1];

double[] prog = {0.0};

return prog;

}

private void setRandomGenoType() {

int chromNum = testAlleles.length;

for (int n = 0; n < chromNum; n++){

String comp = testAlleles[n];

for(int q = 0; q < comp.length(); q++)

{

int indx = (int) (Math.random() \* (comp.length()));

if(indx >1)

indx = 1;

this.genotype += testAlleles[n].charAt(indx);

}

}

}

private void reproduce(Organism parent, Organism parent\_2) {

Organism offspring = new Organism();

Location loc;

int rndmColor = (int) (Math.random() \* this.color.length);

if (parent.getGender().equals("female")){

loc = parent.getLocation();

} else {

loc = parent\_2.getLocation();

}

ArrayList<Location> emptyAdjLocs = getGrid().getEmptyAdjacentLocations(loc);

Grid<Actor> grid = parent.getGrid();

offspring.setColor(color[rndmColor]);

offspring.generation = parent.generation + 1;// -0.1 and 0.1

offspring.maxSteps = (int) ((1 + (Math.random() \* 0.2) - 0.1) \* ((parent.maxSteps + parent\_2.maxSteps)/2));

offspring.crossOver(parent, parent\_2);

System.out.println("Max Steps of Offspring: " + offspring.maxSteps);

if (emptyAdjLocs.size() >= 1) {

offspring.putSelfInGrid(grid, emptyAdjLocs.get((int) (Math.random() \* emptyAdjLocs.size())));

}

}

private void crossOver(Organism p1, Organism p2) {

for (int n = 0; (n < p1.genotype.length()) && (n < p2.genotype.length()); n++) {

int per = (int) (Math.random());

this.genotype += "T";

}

}

//Mix together parent genotypes

//Compare in pairs - Take 1 letter from each parent randomly - if parent only can give A (only has AA in genotype), give A

public StringBuffer getInfo() {

StringBuffer sb = new StringBuffer();

sb.append(("Name: " + this.name + "\n"));

sb.append(("Gender: " + this.gender + "\n"));

sb.append(("Diet: " + this.diet + "\n"));

sb.append(("Genotype: " + this.genotype + "\n"));

sb.append(("Steps Before Death: " + this.death + "\n"));

sb.append(("Generation: " + this.generation + "\n"));

sb.append(("Genotype: ") + this.genotype + "\n");

return sb;

}

//---------------------------------------------------------------------

private Location getNextLocation() {

Location loc = getLocation();

Location next = loc.getAdjacentLocation(getDirection());

return next;

}

private boolean getID(){

return this.id;

}

public ArrayList<Organism> getNeighbourOrganismsFromArrayList(ArrayList<Object> arrList) {

ArrayList<Organism> orgList = new ArrayList<>();

for (int n = 0; n < arrList.size(); n++) {

Object testOrg = arrList.get(n);

try {

if (testOrg instanceof Organism) {

//System.out.println("Organism at index " + n + "\nNow adding to list");

orgList.add((Organism) testOrg);

} else {

//System.out.println("Not an organism at index " + n);

continue;

}

} catch (Exception E) {

//System.out.println("Not an organism");

}

}

return orgList;

}

public ArrayList<Object> getNeighbours(Location loc) {

ArrayList<Object> list = new ArrayList<>();

for (int i = 0; i < this.direction.length; i++) {

Location adj = loc.getAdjacentLocation(this.direction[i]);

if (getGrid().isValid(adj) == true) {

//System.out.println("IsValid");

list.add(getGrid().get(adj)); //Add the object at the location adj

//System.out.println("Organism " + this.name + ", direction checked: " + this.direction[i]);

} else {

//System.out.println("Location " + this.direction[i] + " is not Valid");

continue;

}

}

return list;

}

public String getGender() {

return this.gender;

}

@Override

public void move() {

Location next = getNextLocation();

moveTo(next);

}

public void randomTurn() {

int index = (int) (Math.random() \* this.direction.length);

this.setDirection(direction[index]);

}

@Override

public void act() {

//Checking for adjacent Organisms

ArrayList<Organism> list = this.getNeighbourOrganismsFromArrayList(getNeighbours(this.getLocation()));

for (int i = 0; i < list.size(); i++) {

if (getGender().equalsIgnoreCase(list.get(i).getGender())) {

//System.out.println(this.name + " has same gender as " + list.get(i).name);

list.remove(i);

} else {

//System.out.println(this.name + " has different gender than " + list.get(i).name);

continue;

}

}

if (list.size() >= 1) {

int rndmIndx = (int) (Math.random() \* list.size());

if ((this.generation == (list.get(rndmIndx).generation) && this.oneChild == false)) {

if(BiologySimulation.orgCount == 60){

GUI.remAppend("Max population reached");

} else {

reproduce(this, list.get(rndmIndx));

this.reproducing = true;

this.oneChild = true;

list.get(rndmIndx).oneChild = true;

System.out.println("Number of Organisms in grid: " + BiologySimulation.orgCount);

}

}

}

// Random move

double rndmMove = Math.random();

if (rndmMove < 0.75 && !reproducing) {

if (canMove()) {

move();

} else {

turn();

}

} else {

randomTurn();

}

this.reproducing = false; //Allows organism to reproduce more than once

//Aging - Steps until death

this.age++;

if (this.age >= this.maxSteps) {

Location loc = this.getLocation();

Grid<Actor> grid = this.getGrid();

Grave grave = new Grave(this);

this.removeSelfFromGrid();

BiologySimulation.orgCount--;

System.out.println("Number of Organisms in grid: " + BiologySimulation.orgCount);

grave.putSelfInGrid(grid, loc);

}

//Preventing continuous reproduction (takes a break)

this.lagPeriod--;

if (this.lagPeriod <= 0){

this.oneChild = false;

this.lagPeriod = (int) (this.maxSteps \* 0.8);

}

this.death--;

}

}

//does not place flowers

//checks adjacent squares for organism objects

//checks those objects for opposite gender

//if more than 1 adjacent, picks random one

//ReaperBug doesnt die, kills other Organisms to reduce population

|  |
| --- |
| Updated GUI and Reaper Classes below |

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package biology.simulation;

import biology.Organisms.Organism;

import biology.Organisms.Reaper;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.IOException;

import java.util.EventListener;

import java.util.Scanner;

import java.util.\*;

import javax.swing.\*;

import javax.swing.text.BadLocationException;

import javax.swing.text.DefaultCaret;

/\*\*

\*

\* @author Will

\*/

public class GUI extends JFrame implements EventListener {

public static String pub;

public GUI()//try to fix the menuBar error(no sub menus)

{

boolean isVisible = true, canEdit = false;

int fx = 400, fy = 400;

JFrame frame = new JFrame();

JMenuBar menubar = new JMenuBar();

JMenu file = new JMenu("File");

JMenu \_new = new JMenu("New");

JMenuItem meg = new JMenuItem("Megrez");

JMenuItem organ = new JMenuItem("Organism");

JMenuItem reaper = new JMenuItem("Reaper");

\_new.add(organ);

\_new.add(meg);

\_new.add(reaper);

file.add(\_new);

menubar.add(file);

frame.setSize(fx, fy);

frame.getContentPane().add(BorderLayout.NORTH, menubar);

frame.getContentPane().add(BorderLayout.CENTER, tArea);

pub = ("Welcome to Terminal Dogma\n>> ");

tArea.setEditable(canEdit);

frame.setLocation(650, 20);

frame.setDefaultCloseOperation(3);

organ.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent b1) {

GUI.pub = ("Humph");

BiologySimulation.world.add(BiologySimulation.world.getRandomEmptyLocation(), new Organism());

BiologySimulation.world.show();

}

});

reaper.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent b1) {

pub = ("Death Approaches...");

BiologySimulation.world.add(BiologySimulation.world.getRandomEmptyLocation(), new Reaper());

BiologySimulation.world.show();

}

});

DefaultCaret caret = (DefaultCaret) tArea.getCaret();

caret.setUpdatePolicy(DefaultCaret.ALWAYS\_UPDATE);

frame.add(new JScrollPane(tArea),BorderLayout.CENTER);

frame.setVisible(isVisible);

new UpdateText().execute();

}

//587-352-9199

class UpdateText extends SwingWorker<String, String> {

@Override

public String doInBackground() {

for(int n = 0; n < 100; n++){

publish("Test "+(n+1));

try {

Thread.sleep(500);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return null;

}

@Override

public void process(List<String> chunks) {

for (String s : chunks) {

if (tArea.getDocument().getLength() > 0) {

tArea.append("\n");

}

tArea.append(s);

}

try {

tArea.setCaretPosition(tArea.getLineStartOffset(tArea.getLineCount() - 1));

} catch (BadLocationException e) {

// TODO Auto-generated catch block

}

}

@Override

public void done() {

}

}

public static void remAppend(String foo) {

GUI.pub = foo;

}

public static void newKillButton() {

JFrame frame = new JFrame();

JButton killer = new JButton("Kill?");

int x = 100, y = 100;

killer.setBackground(Color.RED);

killer.setOpaque(true);

killer.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent send) {

//In response to a button click:

}//keyA Action

});//keyA event

frame.setSize(x, y);

frame.getContentPane().add(BorderLayout.CENTER, killer);

frame.show();

}

public static void newMegrezGui() {

int fx = 400, fy = 400;

boolean tAreaIsEditable = false;

JFrame gui = new JFrame();

JTextArea textArea1 = new JTextArea();

JMenuBar menuBar = new JMenuBar();

JMenu file = new JMenu("File");

JMenuItem meg = new JMenuItem("Megrez");

JMenuItem organ = new JMenuItem("Organism");

JMenu \_new = new JMenu("new");

JTextArea t\_area = new JTextArea();

//Item Additions

\_new.add(meg);

\_new.add(organ);

file.add(\_new);

menuBar.add(file);

tArea.setEditable(tAreaIsEditable);

gui.getContentPane().add(BorderLayout.NORTH, menuBar);

gui.getContentPane().add(BorderLayout.CENTER, tArea);

gui.setSize(fx, fy);

gui.setDefaultCloseOperation(3);

gui.setVisible(true);

}

public static JFrame newWindow(int length, int height) {

int fx, fy;

fx = length;

fy = height;

JFrame frame = new JFrame();

frame.setSize(fx, fy);

return frame;

}

public static void newKeystoneGUI(int x, int y) throws IOException {

int q = 2500;

int fx = x;

int fy = y;

int select = 0;

Font timesNewRoman = new Font("Times New Roman", Font.PLAIN, 13);

Font kremlin = new Font("Kremlin", Font.PLAIN, 12);

final String[] file = new String[q];

final String[] path = new String[q];

path[0] = "C:/Users/Will/Desktop/Music/Metal Slug Sound fx WAVS/";

path[1] = "C:/Users/Desktop/Music/";

final String rocketLauncher = file[0] = path[0] + "W Rocket Launcher.wav";

final String enemyChaser = file[1] = path[0] + "W Enemy Chaser.wav";

final String metalSlug = file[2] = path[0] + "Metal Slug.wav";

Scanner sc = new Scanner(new File(file[select]));

JFrame window = GUI.newWindow(fx, fy);

window.setDefaultCloseOperation(3);

//JButton b1 = new JButton("Rocket Lawnchair");

//window.add(BorderLayout.EAST,b1);

//JButton b2 = new JButton("Enemy Chaser");

//window.add(b2);

//b2.setBounds(50, 50, 140, 140);

JMenu file2 = new JMenu("File");

final JTextArea ta = new JTextArea();

JMenuBar menuBar = new JMenuBar();

JMenu play = new JMenu("Play");

JMenu metal\_Slug = new JMenu("Metal Slug SFX");

JMenuItem rLauncher = new JMenuItem("Rocket Launcher");

JMenuItem eChaser = new JMenuItem("Enemy Chaser");

JMenuItem mSlug = new JMenuItem("Metal Slug!");

JMenu open = new JMenu("Open");

JMenuItem kboard = new JMenuItem("Keyboard");

JMenu submenu1 = new JMenu("Submenu Test");

JMenuItem subtank = new JMenuItem("Subtank");

metal\_Slug.add(rLauncher);

metal\_Slug.add(eChaser);

metal\_Slug.add(mSlug);

play.add(metal\_Slug);

submenu1.add(subtank);

open.add(submenu1);

open.add(kboard);

file2.add(open);

menuBar.add(play);

menuBar.add(file2);

menuBar.setVisible(true);

window.add(BorderLayout.NORTH, menuBar);

window.add(BorderLayout.CENTER, ta);

ta.setEditable(false);

JPanel panel = new JPanel(); // the panel is not visible in output

JLabel label = new JLabel("What do you do ?");

final JTextField tf = new JTextField(10);// accepts upto 10 characters

JButton send = new JButton("Send");

JButton reset = new JButton("Clear");

panel.add(label);// Components Added using Flow Layout

panel.add(tf);

panel.add(send);

panel.add(reset);

window.getContentPane().add(BorderLayout.SOUTH, panel);

window.getContentPane().add(BorderLayout.CENTER, ta);

ta.setEditable(false);

tf.setFont(timesNewRoman);

ta.setFont(timesNewRoman);

final String h = tf.getText();

window.setVisible(true);

send.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent send) {

//In response to a button click:

String h = tf.getText();

if (h.equals("")) {

ta.append("");

} else {

ta.append(h + "\n\n");

}

tf.setText("");

}//keyA Action

});//keyA event

reset.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent reset) {

//In response to a button click:

ta.setText("");

}//keyA Action

});//keyA event

// kboard.addActionListener(new ActionListener(){

// @Override

// public void actionPerformed(ActionEvent b3){

// try {

// Files.GetOSKeyboard();

// } catch (IOException ex) {

// Logger.getLogger(Spiderweb.class.getName()).log(Level.SEVERE, null, ex);

// }

// }

// });

}//keystone GUI

public static void newMegrezTerminalWindow() {

int fx = 400, fy = 400;

boolean isVisible = true;

JFrame term = new JFrame();

JTextArea tArea = new JTextArea();

tArea.setEditable(false);

tArea.append("Megrez Terminal\n");

term.setSize(fx, fy);

term.getContentPane().add(BorderLayout.CENTER, tArea);

term.setDefaultCloseOperation(3);

term.setVisible(isVisible);

}

public static JFrame newMegrezTermWindow() {

int fx = 400, fy = 400;

boolean isVisible = true;

JFrame term = new JFrame();

JTextArea t\_Area = new JTextArea();

t\_Area.setEditable(false);

t\_Area.append("Megrez Terminal\n");

term.setSize(fx, fy);

term.getContentPane().add(BorderLayout.CENTER, t\_Area);

term.setDefaultCloseOperation(3);

term.setVisible(isVisible);

return term;

}

public static JTextArea tArea = new JTextArea();

}

|  |
| --- |
| **Reaper Class, Uses Reaper.gif** |

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package biology.Organisms;

import biology.simulation.BiologySimulation;

import biology.simulation.GUI;

import biology.simulation.Grave;

import info.gridworld.actor.Actor;

import info.gridworld.actor.Bug;

import info.gridworld.grid.Grid;

import info.gridworld.grid.Location;

import java.util.ArrayList;

/\*\*

\*

\* @author Will

\*/

public class Reaper extends Organism{

public int kills;

public Reaper()

{

this.kills = 0;

}

@Override

public void act() {

//Checking for adjacent Organisms

ArrayList<Organism> list = this.getNeighbourOrganismsFromArrayList(getNeighbours(this.getLocation()));

this.destroy(list);

// Random move

double rndmMove = Math.random();

if(this.kills >= 10){

GUI.remAppend("Death's job is complete. \n "+this.kills+" kills");

this.removeSelfFromGrid();

BiologySimulation.world.show();

}

if (rndmMove < 0.75) {

if (canMove()) {

move();

} else {

turn();

}

} else {

randomTurn();

}

//Checking for adjacent Organisms after moving

/\*ArrayList<Organism> list2 = this.getNeighbourOrganismsFromArrayList(getNeighbours(this.getLocation()));

for (int i = 0; i < list2.size(); i++) {

if (getGender().equalsIgnoreCase(list2.get(i).getGender())) {

System.out.println(this.name + " has same gender as " + list2.get(i).name);

} else {

System.out.println(this.name + " has different gender than " + list2.get(i).name);

}

}\*/

//Aging - Steps until death

}

public void destroy(ArrayList<Organism> list)

{

for(int n = 0; n < list.size(); n++)

{

Grave grave = new Grave();

Location loc = list.get(n).getLocation();

list.get(n).removeSelfFromGrid();

BiologySimulation.world.add(loc, grave);

++kills;

BiologySimulation.world.show();

}

}

}

|  |
| --- |
| **Organism Class Code June 5/2013** |

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package biology.Organisms;

import biology.simulation.BiologySimulation;

import biology.simulation.GUI;

import biology.simulation.Grave;

import info.gridworld.actor.Actor;

import info.gridworld.actor.Bug;

import info.gridworld.grid.Grid;

import info.gridworld.grid.Location;

import java.awt.Color;

import java.util.ArrayList;

import java.util.Arrays;

/\*\*

\*

\* @author Will

\*/

public class Organism extends Bug{

public String genotype, gender, name, diet, testGene, wingType;

public int cipher, death, key, generation, numLimbs;

private boolean id, reproducing, oneChild;

public int x, y, age, lagPeriod, maxSteps;

public boolean isCarnivorous, isVegetarian, isOmnivorous, genderLocked, dietLocked;

public static String [] testAlleles = {"Aa", "Ii", "Nn"};

private ArrayList<Organism> genes = new ArrayList<>();

public static String[] diets = {"Omnivorous", "Vegetarian", "Carnivorous"}, names = {"Alouette", "Beta", "Gamma", "Rho"}, genders = {"Female", "Male"};

public static int[] direction = {Location.NORTH, Location.EAST,Location.SOUTH,Location.WEST,Location.NORTHEAST, Location.NORTHWEST, Location.SOUTHEAST, Location.SOUTHWEST};

public static Color color[] = {Color.BLACK, Color.BLUE, Color.CYAN, Color.DARK\_GRAY, Color.GRAY, Color.GREEN, Color.LIGHT\_GRAY, Color.MAGENTA, Color.ORANGE, Color.PINK, Color.WHITE, Color.YELLOW, Color.RED};

public Organism() {

int ranName = (int) (Math.random() \* names.length);

int ranGender = (int) (Math.random() \* genders.length);

int ranDiet = (int) (Math.random() \* diets.length);

int ranDirec = (int) (Math.random() \* direction.length);

this.age = 1;

this.lagPeriod = 0;

this.name = names[ranName];

this.genotype = "";

this.diet = diets[ranDiet];

this.dietLocked = false;

this.gender = genders[ranGender];

this.genderLocked = false;

this.genes = null;

this.reproducing = false;

this.generation = 1;

this.oneChild = false;

this.maxSteps = 300;

this.death = this.maxSteps;

this.setDirection(ranDirec);

this.testGene = "";

this.wingType = "";

this.numLimbs = 6;

BiologySimulation.orgCount++;

}

public static ArrayList newMetric\_TonOfOrganisms() {

int ran = (int) (Math.random() \* 50);

ArrayList<Organism> metric = new ArrayList<>();

for (int n = 0; n < ran; n++) {

Organism org = new Organism();

int ranDiet = (int) (Math.random() \* diets.length);

int ranName = (int) (Math.random() \* names.length);

int ranGender = (int) (Math.random() \* genders.length);

int ranDirec = (int) (Math.random() \* direction.length);

org.setName(names[ranName] + " " + n);

org.setGender(genders[ranGender]);

org.setDiet(diets[ranDiet]);

org.setDirection(direction[ranDirec]);

metric.add(org);

org.id = true;

}

return metric;

}

public static ArrayList newMetricTonOfOrganisms(int num) {

ArrayList<Organism> metric = new ArrayList<>();

Arrays.sort(names);

for (int n = 0; n < num; n++) {

Organism org = new Organism();

int ranDiet = (int) (Math.random() \* diets.length);

int ranName = (int) (Math.random() \* names.length);

int ranGender = (int) (Math.round(Math.random()));

int ranDirec = (int) (Math.random() \* direction.length);

org.setRandomGenoType();

org.setName(names[ranName] + " " + n);

org.setGender(genders[ranGender]);

org.setDiet(diets[ranDiet]);

org.setDirection(direction[ranDirec]);

metric.add(org);

}

return metric;

}

public void setName(String name) {

this.name = name;

}

public void mutate()//implement later

{

int ran = (int) (Math.random());

}

public void setGender(String gender) {

if (this.genderLocked != false) {

System.out.println("Gender is Locked");

} else {

String temp = this.gender;

temp.toLowerCase();

if (temp.equalsIgnoreCase("female") || temp.equalsIgnoreCase("male")) {

System.out.println("Not a valid name. Try again");

} else {

this.gender = gender;

this.genderLocked = true;

System.out.println("Gender is not locked, locking " + this.name + "'s gender now now");

}

}

}

public String setGender\_returnString(String gender) {

String ret;

if (this.genderLocked != false) {

ret = ("Gender is Locked");

} else {

String temp = this.gender;

temp.toLowerCase();

if (temp.equalsIgnoreCase("female") || temp.equalsIgnoreCase("male")) {

ret = ("Not a valid name. Try again");

} else {

this.gender = gender;

this.genderLocked = true;

ret = ("Gender is not locked, locking " + this.name + "'s gender now now");

}

}

return ret;

}

public void setDiet(String diet) {

for (int i = 0; i < diets.length; i++) {

if (diet.equalsIgnoreCase((diets[i]))) {

this.diet = diets[i];

}

}

}

private static double[] pundit(String allele\_1, String allele\_2)//omit for now

{

int p1 = allele\_1.length(), p2 = allele\_2.length();

String[][] alleles = new String[p2][p1];

double[] prog = {0.0};

return prog;

}

private void setRandomGenoType() {

int chromNum = testAlleles.length;

for (int n = 0; n < chromNum; n++){

String comp = testAlleles[n];

for(int q = 0; q < comp.length(); q++)

{

int indx = (int) (Math.random() \* (comp.length()));

if(indx >1)

indx = 1;

this.genotype += testAlleles[n].charAt(indx);

}

}

}

private void reproduce(Organism parent, Organism parent\_2) {

Organism offspring = new Organism();

Location loc;

//int rndmColor = (int) (Math.random() \* this.color.length);

if (parent.getGender().equals("female")){

loc = parent.getLocation();

} else {

loc = parent\_2.getLocation();

}

ArrayList<Location> emptyAdjLocs = getGrid().getEmptyAdjacentLocations(loc);

Grid<Actor> grid = parent.getGrid();

//offspring.setColor(color[rndmColor]);

offspring.generation = parent.generation + 1;// -0.1 and 0.1

offspring.maxSteps = (int) ((1 + (Math.random() \* 0.2) - 0.1) \* ((parent.maxSteps + parent\_2.maxSteps)/2));

offspring.crossOver(parent, parent\_2);

offspring.genotypeScanner();

System.out.println("Max Steps of Offspring: " + offspring.maxSteps);

if (emptyAdjLocs.size() >= 1) {

offspring.putSelfInGrid(grid, emptyAdjLocs.get((int) (Math.random() \* emptyAdjLocs.size())));

}

}

private void crossOver(Organism p1, Organism p2) {

String tempGenotype[] = new String[3];

int i = 0;

for (int n = 0; n < p1.genotype.length(); n+=2) {

if ((n + 2) >= 6) {

tempGenotype[i] = p1.genotype.substring(n);

} else {

tempGenotype[i] = p1.genotype.substring(n, n + 2);

}

//--------------

int rndm = (int) Math.round(Math.random());

this.genotype += tempGenotype[i].charAt(rndm);

//---------------

if ((n + 2) >= 6) {

tempGenotype[i] = p2.genotype.substring(n);

} else {

tempGenotype[i] = p2.genotype.substring(n, n + 2);

}

//-------------

int rndm2 = (int) Math.round(Math.random());

this.genotype += tempGenotype[i].charAt(rndm2);

//---------------

i++;

}

}

//Mix together parent genotypes

//Compare in pairs - Take 1 letter from each parent randomly - if parent only can give A (only has AA in genotype), give A

public void genotypeScanner(){

//Checks first two alleles and sets colour accordingly

switch(this.genotype.substring(0, 2)){

case "AA":

this.setColor(Color.RED);

//System.out.println("Color Red with: AA");

break;

case "aa":

this.setColor(Color.BLUE);

//System.out.println("Color Blue with: aa");

break;

case "Aa":

this.setColor(Color.PINK);

//System.out.println("Color Pink with: Aa");

break;

case "aA":

Color color = new Color(110, 0, 200);

this.setColor(color);

//System.out.println("Color Purple with: aA");

break;

}

switch(this.genotype.substring(2, 4)){

case "II":

this.wingType = "Long";

break;

case "ii":

this.wingType = "Short";

break;

default:

this.wingType = "Average";

break;

}

switch(this.genotype.substring(4)){

case "II":

this.numLimbs = 8;

break;

case "ii":

this.numLimbs = 4;

break;

default:

this.numLimbs = 6;

break;

}

}

//first two allele sets are colours dominant (A) = red, recessive (a) = blue

// - scanner feature scans in pairs - new method genotypeScanner

//last two are number of legs - int (even numbers) 6 or 8

//middle number wing type (short/long/average) string

public StringBuffer getInfo() {

StringBuffer sb = new StringBuffer();

sb.append(("Name: " + this.name + "\n"));

sb.append(("Gender: " + this.gender + "\n"));

sb.append(("Diet: " + this.diet + "\n"));

sb.append(("Genotype: " + this.genotype + "\n"));

sb.append(("Steps Taken: " + (this.age - 1) + "\n"));

sb.append(("Steps Before Death: " + this.death + "\n"));

sb.append(("Generation: " + this.generation + "\n"));

sb.append(("Genotype: ") + this.genotype + "\n");

sb.append(("Wing type: ") + this.wingType + "\n");

sb.append(("Limbs: ") + this.numLimbs + "\n");

return sb;

}

//---------------------------------------------------------------------

private Location getNextLocation() {

Location loc = getLocation();

Location next = loc.getAdjacentLocation(getDirection());

return next;

}

private boolean getID(){

return this.id;

}

public ArrayList<Organism> getNeighbourOrganismsFromArrayList(ArrayList<Object> arrList) {

ArrayList<Organism> orgList = new ArrayList<>();

for (int n = 0; n < arrList.size(); n++) {

Object testOrg = arrList.get(n);

try {

if (testOrg instanceof Organism) {

//System.out.println("Organism at index " + n + "\nNow adding to list");

orgList.add((Organism) testOrg);

} else {

//System.out.println("Not an organism at index " + n);

continue;

}

} catch (Exception E) {

//System.out.println("Not an organism");

}

}

return orgList;

}

public ArrayList<Object> getNeighbours(Location loc) {

ArrayList<Object> list = new ArrayList<>();

for (int i = 0; i < this.direction.length; i++) {

Location adj = loc.getAdjacentLocation(this.direction[i]);

if (getGrid().isValid(adj) == true) {

//System.out.println("IsValid");

list.add(getGrid().get(adj)); //Add the object at the location adj

//System.out.println("Organism " + this.name + ", direction checked: " + this.direction[i]);

} else {

//System.out.println("Location " + this.direction[i] + " is not Valid");

continue;

}

}

return list;

}

public String getGender() {

return this.gender;

}

@Override

public void move() {

Location next = getNextLocation();

moveTo(next);

}

public void randomTurn() {

int index = (int) (Math.random() \* this.direction.length);

this.setDirection(direction[index]);

}

@Override

public void act() {

//Checking for adjacent Organisms

ArrayList<Organism> list = this.getNeighbourOrganismsFromArrayList(getNeighbours(this.getLocation()));

for (int i = 0; i < list.size(); i++) {

if (getGender().equalsIgnoreCase(list.get(i).getGender())) {

//System.out.println(this.name + " has same gender as " + list.get(i).name);

list.remove(i);

} else {

//System.out.println(this.name + " has different gender than " + list.get(i).name);

continue;

}

}

if (list.size() >= 1) {

int rndmIndx = (int) (Math.random() \* list.size());

if ((this.generation == (list.get(rndmIndx).generation) && this.oneChild == false && this.age >= 50)) {

if(BiologySimulation.orgCount == 60){

GUI.remAppend("Max population reached");

} else {

reproduce(this, list.get(rndmIndx));

this.reproducing = true;

this.oneChild = true;

list.get(rndmIndx).oneChild = true;

System.out.println("Number of Organisms in grid: " + BiologySimulation.orgCount + "\n");

}

}

}

// Random move

double rndmMove = Math.random();

if (rndmMove < 0.75 && !reproducing) {

if (canMove()) {

move();

} else {

turn();

}

} else {

randomTurn();

}

this.reproducing = false; //Allows organism to reproduce more than once

//Aging - Steps until death

this.age++;

if (this.age >= this.maxSteps) {

Location loc = this.getLocation();

Grid<Actor> grid = this.getGrid();

Grave grave = new Grave(this);

this.removeSelfFromGrid();

BiologySimulation.orgCount--;

System.out.println("Number of Organisms in grid: " + BiologySimulation.orgCount + "\n");

grave.putSelfInGrid(grid, loc);

}

//Preventing continuous reproduction (takes a break)

this.lagPeriod--;

if (this.lagPeriod <= 0){

this.oneChild = false;

this.lagPeriod = (int) (this.maxSteps \* 0.8);

}

this.death--;

}

}

//does not place flowers

//checks adjacent squares for organism objects

//checks those objects for opposite gender

//if more than 1 adjacent, picks random one

//ReaperBug doesnt die, kills other Organisms to reduce population

//Implement method: "Bottleneck"

//Founder method remove organisms from grid (either killing, or moving to different grid/ActorWorld)

//-Fix crossover method

// 2 int variables - tells program where the alleles are

//first two allele sets are colours dominant (A) = red, recessive (a) = blue

// - scanner feature scans in pairs - new method genotypeScanner

//last two are number of legs - int (even numbers) 6 or 8

//middle number wing type (short/long) string

|  |
| --- |
| Main Method June 5/2013 |

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package biology.simulation;

import biology.Organisms.Organism;

import info.gridworld.actor.ActorWorld;

import info.gridworld.actor.Rock;

import info.gridworld.grid.BoundedGrid;

import javax.swing.JFrame;

import java.util.\*;

import java.io.\*;

/\*\*

\*

\* @author Will

\*/

public class BiologySimulation {

/\*\*

\* @param args the command line arguments

\*/

static String in1;

public static int orgCount;

public static void main(String[] args) {

// TODO code application logic here

GUI gui = new GUI();

ArrayList orgList = Organism.newMetricTonOfOrganisms(5);

BoundedGrid grid = new BoundedGrid(13, 13);

ActorWorld world = new ActorWorld();

world.setGrid(grid);

for(int n = 0; n < orgList.size(); n++)

{

Organism org = (Organism)orgList.get(n);

Rock rock = new Rock();

world.add(org);

world.add(rock);

}

world.show();

GUI.remAppend("");

GUI.remAppend(""+Organism.testAlleles[0]);

while(!"exit".equals(in1))

{

Scanner sc = new Scanner(System.in);

in1 = sc.nextLine();

GUI.remAppend(in1);

}

}//main

}

|  |
| --- |
| Grave Code June 5/2013 |

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package biology.simulation;

import biology.Organisms.Organism;

import info.gridworld.actor.Rock;

/\*\*

\*

\* @author student

\*/

public class Grave extends Rock {

public int generation, chronos;

public String name, gender, diet, genotype;

public Grave(Organism organism) {

this.generation = organism.generation;

this.name = organism.name;

this.gender = organism.gender;

this.diet = organism.diet;

this.genotype = organism.genotype;

this.chronos = (int) (Math.random() \* (organism.age/2));

}

@Override

public void act() {

this.chronos = chronos - 1;

if (this.chronos <= 0){

this.removeSelfFromGrid();

}

}

public StringBuffer getInfo() {

StringBuffer sb = new StringBuffer();

sb.append(("Name: " + this.name + "\n"));

sb.append(("Gender: " + this.gender + "\n"));

sb.append(("Diet: " + this.diet + "\n"));

sb.append(("Genotype: " + this.genotype + "\n"));

sb.append(("Generation: " + this.generation + "\n"));

sb.append(("Time till Disappearance: " + this.chronos + "\n"));

return sb;

}

public int getChronos() {

return this.chronos;

}

}

|  |
| --- |
| **Updated Organism code June 6th, 2013** |

**/\***

**\* To change this template, choose Tools | Templates**

**\* and open the template in the editor.**

**\*/**

**package biology.Organisms;**

**import biology.simulation.BiologySimulation;**

**import biology.simulation.GUI;**

**import biology.simulation.Grave;**

**import info.gridworld.actor.Actor;**

**import info.gridworld.actor.Bug;**

**import info.gridworld.grid.Grid;**

**import info.gridworld.grid.Location;**

**import java.awt.Color;**

**import java.util.ArrayList;**

**import java.util.Arrays;**

**/\*\***

**\***

**\* @author Will**

**\*/**

**public class Organism extends Bug{**

**public String genotype, gender, name, diet, testGene, wingType;**

**public int cipher, death, key, generation, numLimbs;**

**private boolean id, reproducing, oneChild;**

**public int x, y, age, lagPeriod, maxSteps, steps;**

**public boolean isCarnivorous, isVegetarian, isOmnivorous, genderLocked, dietLocked;**

**public static String [] testAlleles = {"Aa", "Ii", "Nn"};**

**private ArrayList<Organism> genes = new ArrayList<>();**

**public static String[] diets = {"Omnivorous", "Vegetarian", "Carnivorous"}, names = {"Alouette", "Beta", "Gamma", "Rho"}, genders = {"Female", "Male"};**

**public static int[] direction = {Location.NORTH, Location.EAST,Location.SOUTH,Location.WEST,Location.NORTHEAST, Location.NORTHWEST, Location.SOUTHEAST, Location.SOUTHWEST};**

**public static Color color[] = {Color.BLACK, Color.BLUE, Color.CYAN, Color.DARK\_GRAY, Color.GRAY, Color.GREEN, Color.LIGHT\_GRAY, Color.MAGENTA, Color.ORANGE, Color.PINK, Color.WHITE, Color.YELLOW, Color.RED};**

**public Organism() {**

**int ranName = (int) (Math.random() \* names.length);**

**int ranGender = (int) (Math.random() \* genders.length);**

**int ranDiet = (int) (Math.random() \* diets.length);**

**int ranDirec = (int) (Math.random() \* direction.length);**

**this.age = 1;**

**this.lagPeriod = 0;**

**this.name = names[ranName];**

**this.genotype = "";**

**this.diet = diets[ranDiet];**

**this.dietLocked = false;**

**this.gender = genders[ranGender];**

**this.genderLocked = false;**

**this.genes = null;**

**this.reproducing = false;**

**this.generation = 1;**

**this.oneChild = false;**

**this.maxSteps = 300;**

**this.death = this.maxSteps;**

**this.setDirection(ranDirec);**

**this.testGene = "";**

**this.wingType = "";**

**this.numLimbs = 6;**

**BiologySimulation.orgCount++;**

**}**

**public static ArrayList newMetric\_TonOfOrganisms() {**

**int ran = (int) (Math.random() \* 50);**

**ArrayList<Organism> metric = new ArrayList<>();**

**for (int n = 0; n < ran; n++) {**

**Organism org = new Organism();**

**int ranDiet = (int) (Math.random() \* diets.length);**

**int ranName = (int) (Math.random() \* names.length);**

**int ranGender = (int) (Math.random() \* genders.length);**

**int ranDirec = (int) (Math.random() \* direction.length);**

**org.setName(names[ranName] + " " + n);**

**org.setGender(genders[ranGender]);**

**org.setDiet(diets[ranDiet]);**

**org.setDirection(direction[ranDirec]);**

**metric.add(org);**

**org.id = true;**

**}**

**return metric;**

**}**

**public static ArrayList newMetricTonOfOrganisms(int num) {**

**ArrayList<Organism> metric = new ArrayList<>();**

**Arrays.sort(names);**

**for (int n = 0; n < num; n++) {**

**Organism org = new Organism();**

**int ranDiet = (int) (Math.random() \* diets.length);**

**int ranName = (int) (Math.random() \* names.length);**

**int ranGender = (int) (Math.round(Math.random()));**

**int ranDirec = (int) (Math.random() \* direction.length);**

**org.setRandomGenoType();**

**org.setName(names[ranName] + " " + n);**

**org.setGender(genders[ranGender]);**

**org.setDiet(diets[ranDiet]);**

**org.setDirection(direction[ranDirec]);**

**metric.add(org);**

**}**

**return metric;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public void mutate()//implement later**

**{**

**int ran = (int) (Math.random());**

**}**

**public void setGender(String gender) {**

**if (this.genderLocked != false) {**

**System.out.println("Gender is Locked");**

**} else {**

**String temp = this.gender;**

**temp.toLowerCase();**

**if (temp.equalsIgnoreCase("female") || temp.equalsIgnoreCase("male")) {**

**System.out.println("Not a valid name. Try again");**

**} else {**

**this.gender = gender;**

**this.genderLocked = true;**

**System.out.println("Gender is not locked, locking " + this.name + "'s gender now now");**

**}**

**}**

**}**

**public String setGender\_returnString(String gender) {**

**String ret;**

**if (this.genderLocked != false) {**

**ret = ("Gender is Locked");**

**} else {**

**String temp = this.gender;**

**temp.toLowerCase();**

**if (temp.equalsIgnoreCase("female") || temp.equalsIgnoreCase("male")) {**

**ret = ("Not a valid name. Try again");**

**} else {**

**this.gender = gender;**

**this.genderLocked = true;**

**ret = ("Gender is not locked, locking " + this.name + "'s gender now now");**

**}**

**}**

**return ret;**

**}**

**public void setDiet(String diet) {**

**for (int i = 0; i < diets.length; i++) {**

**if (diet.equalsIgnoreCase((diets[i]))) {**

**this.diet = diets[i];**

**}**

**}**

**}**

**private static double[] pundit(String allele\_1, String allele\_2)//omit for now**

**{**

**int p1 = allele\_1.length(), p2 = allele\_2.length();**

**String[][] alleles = new String[p2][p1];**

**double[] prog = {0.0};**

**return prog;**

**}**

**private void setRandomGenoType() {**

**int chromNum = testAlleles.length;**

**for (int n = 0; n < chromNum; n++){**

**String comp = testAlleles[n];**

**for(int q = 0; q < comp.length(); q++)**

**{**

**int indx = (int) (Math.random() \* (comp.length()));**

**if(indx >1)**

**indx = 1;**

**this.genotype += testAlleles[n].charAt(indx);**

**}**

**}**

**}**

**private void reproduce(Organism parent, Organism parent\_2) {**

**Organism offspring = new Organism();**

**Location loc;**

**//int rndmColor = (int) (Math.random() \* this.color.length);**

**if (parent.getGender().equals("female")){**

**loc = parent.getLocation();**

**} else {**

**loc = parent\_2.getLocation();**

**}**

**ArrayList<Location> emptyAdjLocs = getGrid().getEmptyAdjacentLocations(loc);**

**Grid<Actor> grid = parent.getGrid();**

**//offspring.setColor(color[rndmColor]);**

**offspring.generation = parent.generation + 1;// -0.1 and 0.1**

**if (parent.age > 100 && parent\_2.age > 100) {**

**offspring.maxSteps = (int) ((1 + (Math.random() \* 0.2) - 0.1) \* ((parent.steps + parent\_2.steps) / 2));**

**offspring.death = offspring.maxSteps;**

**} else {**

**offspring.maxSteps = (int) ((1 + (Math.random() \* 0.2) - 0.1) \* ((parent.steps + parent\_2.steps + (this.maxSteps + 200)) / 2));**

**offspring.death = offspring.maxSteps;**

**}**

**offspring.crossOver(parent, parent\_2);**

**offspring.genotypeScanner();**

**System.out.println("Max Steps of Offspring: " + offspring.maxSteps);**

**if (emptyAdjLocs.size() >= 1) {**

**offspring.putSelfInGrid(grid, emptyAdjLocs.get((int) (Math.random() \* emptyAdjLocs.size())));**

**}**

**}**

**private void crossOver(Organism p1, Organism p2) {**

**String tempGenotype[] = new String[3];**

**int i = 0;**

**for (int n = 0; n < p1.genotype.length(); n+=2) {**

**if ((n + 2) >= 6) {**

**tempGenotype[i] = p1.genotype.substring(n);**

**} else {**

**tempGenotype[i] = p1.genotype.substring(n, n + 2);**

**}**

**//--------------**

**int rndm = (int) Math.round(Math.random());**

**this.genotype += tempGenotype[i].charAt(rndm);**

**//---------------**

**if ((n + 2) >= 6) {**

**tempGenotype[i] = p2.genotype.substring(n);**

**} else {**

**tempGenotype[i] = p2.genotype.substring(n, n + 2);**

**}**

**//-------------**

**int rndm2 = (int) Math.round(Math.random());**

**this.genotype += tempGenotype[i].charAt(rndm2);**

**//---------------**

**i++;**

**}**

**}**

**//Mix together parent genotypes**

**//Compare in pairs - Take 1 letter from each parent randomly - if parent only can give A (only has AA in genotype), give A**

**private void bettercrossOver(Organism p1, Organism p2) {**

**String tempGenotype[] = new String[3];**

**int i = 0;**

**for (int n = 0; n < p1.genotype.length(); n+=2) {**

**String chs = (genotype.substring(n)+genotype.substring(n+1));**

**if(chs.equals("AA"))**

**tempGenotype[i] = p1.genotype.substring(n,n+1);**

**else if(chs.equals("aa"))**

**tempGenotype[i] = p1.genotype.substring(n,n+1);**

**else**

**{**

**int ran = (int) (Math.random() \* n);**

**}**

**//--------------**

**int rndm = (int) Math.round(Math.random());**

**this.genotype += tempGenotype[i].charAt(rndm);**

**//---------------**

**if ((n + 2) >= 6) {**

**tempGenotype[i] = p2.genotype.substring(n);**

**} else {**

**tempGenotype[i] = p2.genotype.substring(n, n + 2);**

**}**

**//-------------**

**int rndm2 = (int) Math.round(Math.random());**

**this.genotype += tempGenotype[i].charAt(rndm2);**

**//---------------**

**i++;**

**}**

**}**

**public void genotypeScanner(){**

**//Checks first two alleles and sets colour accordingly**

**switch(this.genotype.substring(0, 2)){**

**case "AA":**

**this.setColor(Color.RED);**

**//System.out.println("Color Red with: AA");**

**break;**

**case "aa":**

**this.setColor(Color.BLUE);**

**//System.out.println("Color Blue with: aa");**

**break;**

**default:**

**this.setColor(Color.GREEN);**

**//System.out.println("Color Green with: Aa");**

**break;**

**}**

**switch(this.genotype.substring(2, 4)){**

**case "II":**

**this.wingType = "Long";**

**break;**

**case "ii":**

**this.wingType = "Short";**

**break;**

**default:**

**this.wingType = "Average";**

**break;**

**}**

**switch(this.genotype.substring(4)){**

**case "II":**

**this.numLimbs = 8;**

**break;**

**case "ii":**

**this.numLimbs = 4;**

**break;**

**default:**

**this.numLimbs = 6;**

**break;**

**}**

**}**

**//first two allele sets are colours dominant (A) = red, recessive (a) = blue**

**// - scanner feature scans in pairs - new method genotypeScanner**

**//last two are number of legs - int (even numbers) 6 or 8**

**//middle number wing type (short/long/average) string**

**public StringBuffer getInfo() {**

**StringBuffer sb = new StringBuffer();**

**sb.append(("Name: " + this.name + "\n"));**

**sb.append(("Gender: " + this.gender + "\n"));**

**sb.append(("Diet: " + this.diet + "\n"));**

**sb.append(("Genotype: " + this.genotype + "\n"));**

**sb.append(("Steps Taken: " + (this.steps) + "\n"));**

**sb.append(("Steps Before Death: " + this.death + "\n"));**

**sb.append(("Generation: " + this.generation + "\n"));**

**sb.append(("Genotype: ") + this.genotype + "\n");**

**sb.append(("Wing type: ") + this.wingType + "\n");**

**sb.append(("Limbs: ") + this.numLimbs + "\n");**

**return sb;**

**}**

**//---------------------------------------------------------------------**

**private Location getNextLocation() {**

**Location loc = getLocation();**

**Location next = loc.getAdjacentLocation(getDirection());**

**return next;**

**}**

**private boolean getID(){**

**return this.id;**

**}**

**public ArrayList<Organism> getNeighbourOrganismsFromArrayList(ArrayList<Object> arrList) {**

**ArrayList<Organism> orgList = new ArrayList<>();**

**for (int n = 0; n < arrList.size(); n++) {**

**Object testOrg = arrList.get(n);**

**try {**

**if (testOrg instanceof Organism) {**

**//System.out.println("Organism at index " + n + "\nNow adding to list");**

**orgList.add((Organism) testOrg);**

**} else {**

**//System.out.println("Not an organism at index " + n);**

**continue;**

**}**

**} catch (Exception E) {**

**//System.out.println("Not an organism");**

**}**

**}**

**return orgList;**

**}**

**public ArrayList<Object> getNeighbours(Location loc) {**

**ArrayList<Object> list = new ArrayList<>();**

**for (int i = 0; i < this.direction.length; i++) {**

**Location adj = loc.getAdjacentLocation(this.direction[i]);**

**if (getGrid().isValid(adj) == true) {**

**//System.out.println("IsValid");**

**list.add(getGrid().get(adj)); //Add the object at the location adj**

**//System.out.println("Organism " + this.name + ", direction checked: " + this.direction[i]);**

**} else {**

**//System.out.println("Location " + this.direction[i] + " is not Valid");**

**continue;**

**}**

**}**

**return list;**

**}**

**public String getGender() {**

**return this.gender;**

**}**

**@Override**

**public void move() {**

**Location next = getNextLocation();**

**moveTo(next);**

**}**

**public void randomTurn() {**

**int index = (int) (Math.random() \* this.direction.length);**

**this.setDirection(direction[index]);**

**}**

**@Override**

**public void act() {**

**//Checking for adjacent Organisms**

**ArrayList<Organism> list = this.getNeighbourOrganismsFromArrayList(getNeighbours(this.getLocation()));**

**for (int i = 0; i < list.size(); i++) {**

**if (getGender().equalsIgnoreCase(list.get(i).getGender())) {**

**//System.out.println(this.name + " has same gender as " + list.get(i).name);**

**list.remove(i);**

**} else {**

**//System.out.println(this.name + " has different gender than " + list.get(i).name);**

**continue;**

**}**

**}**

**if (list.size() >= 1) {**

**int rndmIndx = (int) (Math.random() \* list.size());**

**if ((this.generation == (list.get(rndmIndx).generation) && this.oneChild == false && this.age >= 50)) {**

**if(BiologySimulation.orgCount == 60){**

**GUI.remAppend("Max population reached");**

**} else {**

**reproduce(this, list.get(rndmIndx));**

**this.reproducing = true;**

**this.oneChild = true;**

**list.get(rndmIndx).oneChild = true;**

**System.out.println("Number of Organisms in grid: " + BiologySimulation.orgCount + "\n");**

**}**

**}**

**}**

**// Random move**

**double rndmMove = Math.random();**

**if (rndmMove < 0.75 && !reproducing) {**

**if (canMove()) {**

**move();**

**this.steps++;**

**} else {**

**turn();**

**}**

**} else {**

**randomTurn();**

**}**

**this.reproducing = false; //Allows organism to reproduce more than once**

**//Aging - Steps until death**

**this.age++;**

**if (this.age >= this.maxSteps) {**

**Location loc = this.getLocation();**

**Grid<Actor> grid = this.getGrid();**

**Grave grave = new Grave(this);**

**this.removeSelfFromGrid();**

**BiologySimulation.orgCount--;**

**System.out.println("Number of Organisms in grid: " + BiologySimulation.orgCount + "\n");**

**grave.putSelfInGrid(grid, loc);**

**}**

**//Preventing continuous reproduction (takes a break)**

**this.lagPeriod--;**

**if (this.lagPeriod <= 0){**

**this.oneChild = false;**

**this.lagPeriod = (int) (this.maxSteps \* 0.8);**

**}**

**this.death--;**

**}**

**}**

**//does not place flowers**

**//checks adjacent squares for organism objects**

**//checks those objects for opposite gender**

**//if more than 1 adjacent, picks random one**

**//ReaperBug doesnt die, kills other Organisms to reduce population**

**//Implement method: "Bottleneck"**

**//Founder method remove organisms from grid (either killing, or moving to different grid/ActorWorld)**

**//-Fix crossover method**

**// 2 int variables - tells program where the alleles are**

**//first two allele sets are colours dominant (A) = red, recessive (a) = blue**

**// - scanner feature scans in pairs - new method genotypeScanner**

**//last two are number of legs - int (even numbers) 6 or 8**

**//middle number wing type (short/long) string**

|  |
| --- |
| **Pitch: Step 1 of 4 (Sorry, I don’t know how to incorporate the biology terms properly) -please add!** |

Our project is basically a biology simulation that models the evolution that occurs in an enclosed environment, speaking in terms of biology, it Mimics the Hardy-weinberg principle and natural growth and declination of genes. Organisms move around in search of a partner to reproduce and expand the population. The offspring of two parent organisms inherit certain traits that define its number of limbs, wing size, and body colour. Its lifespan also depends on how long the parents were meant to live for. The simulation will demonstrate how species struggle against overpopulation and sometimes extinction when they fail to reproduce. The name is still yet to be determined, but for now its just been referred to as “Project Rosetta”

|  |
| --- |
| **Description of Project: Step 2 of 4** |

- An equal number Organism objects and Rock objects are created in a grid.

- Organism objects randomly move around the grid and check their surroundings for another organism of the opposite gender to reproduce. Organisms can only mate with other Organisms of the same generation. One Organism offspring is created and placed in a random available spot beside the female. If there are no available spaces beside the female to place the offspring, no offspring is made.

- Organism offspring inherit certain traits from its parents depending on its genotype. Its lifespan also depends on its parents. All information regarding a certain organism is available via the getInfo method.

- Organism objects take a random predetermined number of steps before they “die” and get replaced with a Grave object. This Grave object acts as a Rock and block the living Organisms’ path. The Grave object is also eventually removed from grid depending on the number of steps the Organism took.

- A Reaper object can be put into the grid to reduce the population of the Organisms. The Reaper will move randomly like the Organism, but it does not die after a certain number of steps. It will only remove itself when it has accomplished killing a certain number of Organisms, which has now been rendered random..