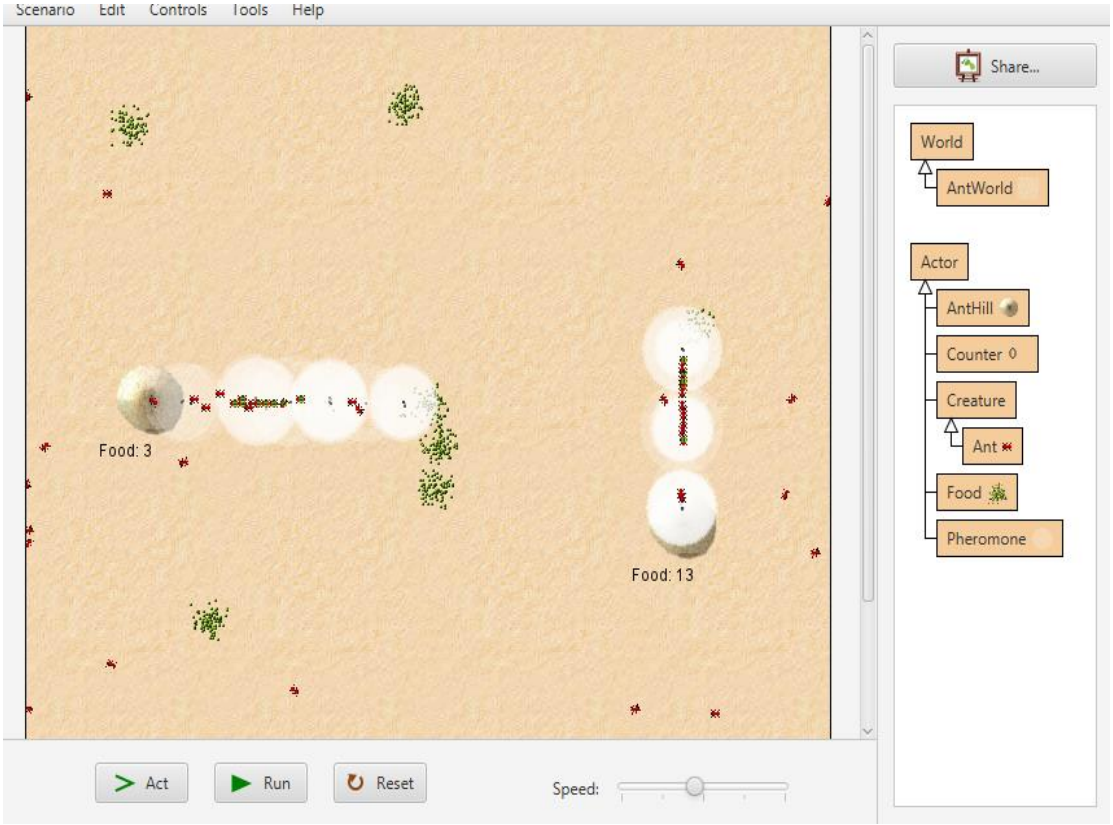
Άσκηση 11η

Δημιουργήστε τις κλάσεις και τις υποκλάσεις όπως απεικονίζονται παρακάτω, γράψτε τους κώδικες στους αντίστοιχους editor και «τρέξτε» το πρόγραμμα….



Το παρακάνω πρόγραμμα Java δημιουργεί έναν κόσμο όπου ζουν μυρμήγκια σε ένα περιβάλλον Greenfoot. Ας αναλύσουμε τα σχόλια για κάθε μέθοδο:

Κατασκευαστής (Constructor) AntWorld: Η μέθοδος δημιουργεί έναν κόσμο με συγκεκριμένο μέγεθος και θέτει τη σειρά στην οποία θα σχεδιαστούν τα αντικείμενα. Καλεί τη μέθοδο setup2() για να προσθέσει αντικείμενα στον κόσμο.

Μέθοδος setup1(): Δημιουργεί έναν κόσμο με ένα λόφο μυρμηγκιού και τροφή. Τα αντικείμενα τοποθετούνται σε συγκεκριμένες θέσεις.

Μέθοδος setup2(): Παρόμοια με την setup1(), αλλά δημιουργεί έναν κόσμο με δύο λόφους μυρμηγκιών και τροφή.

Μέθοδος setup3(): Και πάλι, παρόμοια με τις προηγούμενες, αλλά δημιουργεί έναν κόσμο με δύο διαφορετικούς λόφους μυρμηγκιών και τροφή.

```
import greenfoot.*;  // (World, Actor, GreenfootImage, and Greenfoot)
/**
 * Ο κόσμος όπου ζουν οι μυρμήγκια.
 *
 * Συγγραφέας: Michael Kölling
 * Έκδοση: 1.1
 */
public class AntWorld extends World
{
    public static final int SIZE = 620;

    /**
     * Δημιουργεί έναν νέο κόσμο. Θα αρχικοποιηθεί με μερικές φωλιές μυρμηγκιών
     * και πηγές τροφής.
     */
    public AntWorld()
    {
        super(SIZE, SIZE, 1);
        setPaintOrder(Ant.class, Pheromone.class, AntHill.class, Food.class);
        setup2();  // Καλεί τη μέθοδο setup2() για τη δημιουργία του κόσμου.
    }
    /**
     * Δημιουργεί τα περιεχόμενα του κόσμου: μία φωλιά μυρμηγκιών και τροφή.
     */
    public void setup1()
    {
        removeObjects(getObjects(null));  // Αφαιρεί όλα τα υπάρχοντα αντικείμενα
        addObject(new AntHill(70), SIZE / 2, SIZE / 2);
        addObject(new Food(), SIZE / 2, SIZE / 2 - 260);
        addObject(new Food(), SIZE / 2 + 215, SIZE / 2 - 100);
        addObject(new Food(), SIZE / 2 + 215, SIZE / 2 + 100);
        addObject(new Food(), SIZE / 2, SIZE / 2 + 260);
        addObject(new Food(), SIZE / 2 - 215, SIZE / 2 + 100);
        addObject(new Food(), SIZE / 2 - 215, SIZE / 2 - 100);
    }
    /**
     * Δημιουργεί τα περιεχόμενα του κόσμου: δεύτερη φωλιά μυρμηγκιών και τροφή.
     */
    public void setup2()
    {
        removeObjects(getObjects(null));  // Αφαιρεί όλα τα υπάρχοντα αντικείμενα
        addObject(new AntHill(40), 506, 356);
        addObject(new AntHill(40), 95, 267);

        addObject(new Food(), 80, 71);
        addObject(new Food(), 291, 56);
        addObject(new Food(), 516, 212);
        addObject(new Food(), 311, 269);
        addObject(new Food(), 318, 299);
        addObject(new Food(), 315, 331);
        addObject(new Food(), 141, 425);
        addObject(new Food(), 378, 547);
        addObject(new Food(), 566, 529);
    }
```

```
    /**
     * Δημιουργεί τα περιεχόμενα του κόσμου: τρίτη φωλιά μυρμηγκιών και τροφή.
     */
    public void setup3()
    {
        removeObjects(getObjects(null));  // Αφαιρεί όλα τα υπάρχοντα αντικείμενα
        addObject(new AntHill(40), 576, 134);
        addObject(new AntHill(40), 59, 512);

        addObject(new Food(), 182, 84);
        addObject(new Food(), 39, 308);
        addObject(new Food(), 249, 251);
        addObject(new Food(), 270, 272);
        addObject(new Food(), 291, 253);
        addObject(new Food(), 339, 342);
        addObject(new Food(), 593, 340);
        addObject(new Food(), 487, 565);
    }
}
```

```
import greenfoot.*;  // Εισαγωγή των απαραίτητων κλάσεων από τη βιβλιοθήκη Greenfoot
/**
 * Ένα λόφος γεμάτος μυρμήγκια.
 *
 * Ο συγγραφέας αναφέρεται στον Michael Kölling
 */
public class AntHill extends Actor
{
    /** Ο αριθμός των μυρμηγκιών που έχουν βγει μέχρι τώρα. */
    private int ants = 0;
    /** Ο συνολικός αριθμός των μυρμηγκιών σε αυτόν το λόφο. */
    private int maxAnts = 40;
    /** Μετρητής για να εμφανίζει πόση τροφή έχει συλλεχθεί μέχρι τώρα. */
    private Counter foodCounter;

    /**
     * Κατασκευαστής για το λόφο μυρμηγκιών με προκαθορισμένο αριθμό μυρμηγκιών (40).
     */
    public AntHill()
    {
    }
    /**
     * Κατασκευαστής για το λόφο μυρμηγκιών με καθορισμένο αριθμό μυρμηγκιών.
     */
    public AntHill(int numberOfAnts)
    {
        maxAnts = numberOfAnts;
    }
    /**
     * Ενέργεια: Εάν υπάρχουν ακόμα μυρμήγκια μέσα, ελέγξτε εάν θα πρέπει να βγει ένα.
     */
    public void act()
    {
        if(ants < maxAnts)
        {
            if(Greenfoot.getRandomNumber(100) < 10)
            {
                getWorld().addObject(new Ant(this), getX(), getY());
                ants++;
            }
        }
    }
    /**
     * Καταγραφή ότι έχουμε συλλέξει άλλο κομμάτι τροφής.
     */
    public void countFood()
    {
        if(foodCounter == null)
        {
            foodCounter = new Counter("Food: ");
            int x = getX();
            int y = getY() + getImage().getWidth()/2 + 8;
            getWorld().addObject(foodCounter, x, y);
        }
        foodCounter.increment();
    }
}
```

```java
import greenfoot.*;  // Εισαγωγή των απαραίτητων κλάσεων από τη βιβλιοθήκη Greenfoot
/**
 * Κλάση Counter που εμφανίζει κείμενο και έναν αριθμό.
 *
 * Ο συγγραφέας της κλάσης είναι ο Michael Kölling και η έκδοση είναι 1.1.
 */
public class Counter extends Actor
{
    private int value = 0; // Η τρέχουσα τιμή του μετρητή
    private String text;   // Το κείμενο προθέματος του μετρητή

    /**
     * Δημιουργία ενός μετρητή χωρίς προθέματος κειμένου, αρχικοποιημένος στο μηδέν.
     */
    public Counter()
    {
        this(""); // Κλήση του άλλου κατασκευαστή με κενό κείμενο προθέματος
    }
    /**
     * Δημιουργία ενός μετρητή με καθορισμένο πρόθεμα κειμένου, αρχικοποιημένος στο μηδέν.
     */
    public Counter(String prefix)
    {
        text = prefix; // Καθορισμός του κειμένου προθέματος
        int imageWidth = (text.length() + 2) * 10; // Υπολογισμός πλάτους της εικόνας
        setImage(new GreenfootImage(imageWidth, 16)); // Δημιουργία εικόνας με τον υπολογισμένο πλάτος
        updateImage(); // Ανανέωση της εικόνας
    }
    /**
     * Αυξάνει την τιμή του μετρητή κατά ένα.
     */
    public void increment()
    {
        value++; // Αύξηση της τιμής του μετρητή
        updateImage(); // Ανανέωση της εικόνας
    }
    /**
     * Αυξάνει την τιμή του μετρητή κατά ένα.
     */
    public void increment()
    {
        value++; // Αύξηση της τιμής του μετρητή
        updateImage(); // Ανανέωση της εικόνας
    }
    /**
     * Εμφανίζει το τρέχον κείμενο και τον αριθμό στην εικόνα του αντικειμένου.
     */
    private void updateImage()
    {
        GreenfootImage image = getImage(); // Λήψη της εικόνας
        image.clear(); // Καθαρισμός της εικόνας
        image.drawString(text + value, 1, 12); // Εμφάνιση του κειμένου και της τιμής στην εικόνα
    }
}
```

```java
import greenfoot.*;  // Εισαγωγή των απαραίτητων κλάσεων από τη βιβλιοθήκη Greenfoot
/**
 * Κλάση Creature που αναπαριστά ένα πλάσμα σε μια προσομοίωση.
 * Το πλάσμα έχει μια γνωστή φωλιά. Μπορεί να κινείται και να κατευθύνεται προς ή μακριά από τη φωλιά του.
 *
 * Η κίνηση του πλάσματος οργανώνεται με την αποθήκευση ενός ζεύγους deltaX/deltaY: τις μετατοπίσεις στις x/y κατευθύνσεις
 * που το πλάσμα θα κινηθεί στο επόμενο βήμα. Η τιμή για αυτές περιορίζεται από τη σταθερά SPEED:
 * οι τιμές των διαφορών θα είναι πάντα στο εύρος [-SPEED..SPEED].
 *
 * Ο συγγραφέας της κλάσης είναι ο Michael Kölling και η έκδοση είναι 1.0.
 */
public class Creature extends Actor
{
    /** Η μέγιστη ταχύτητα κίνησης του μυρμηγκιού. */
    private static final int SPEED = 3;

    /** Τρέχουσα κίνηση. Ορίζεται ως η μετατόπιση στις x και y κατευθύνσεις που κινείται σε κάθε βήμα. */
    private int deltaX;
    private int deltaY;

    /** Η φωλιά του μυρμηγκιού. */
    private AntHill home;

    /** Η φωλιά του μυρμηγκιού. */
    private AntHill home;

    /**
     * Δημιουργία ενός νέου πλάσματος με αδρανή κίνηση (η ταχύτητα κίνησης είναι μηδέν).
     */
    public Creature()
    {
        deltaX = 0;
        deltaY = 0;
    }
```

```java
/**
 * Ορισμός της φωλιάς του πλάσματος.
 */
public void setHomeHill(AntHill homeHill)
{
    home = homeHill;
}
/**
 * Επιστροφή της φωλιάς του πλάσματος.
 */
public AntHill getHomeHill()
{
    return home;
}
/**
 * Περίπατος τυχαία (τυχαία κατεύθυνση και ταχύτητα).
 */
public void randomWalk()
{
    if (randomChance(50)) {
        deltaX = adjustSpeed(deltaX);
        deltaY = adjustSpeed(deltaY);
    }
    walk();
}
/**
 * Προσπάθει να περπατήσει προς το σπίτι. Μερικές φορές τα πλάσματα αποσπώνται ή συναντούν μικρά εμπόδια, έτσι
 * περιστρέφονται περιστασιακά σε διαφορετική κατεύθυνση για ένα σύντομο χρονικό διάστημα.
 */
public void walkTowardsHome()
{
    if(home == null) {
        // αν δεν έχουμε ένα σπίτι, δεν μπορούμε να πάμε εκεί.
        return;
    }
    if (randomChance(2)) {
        randomWalk();  // δεν μπορεί πάντα να περπατήσει ευθεία. 2% πιθανότητα να αποκλίνει από την πορεία.
    }
    else {
        headRoughlyTowards(home);
        walk();
    }
}
/**
 * Προσπάθει να περπατήσει μακριά από το σπίτι. (Συχνά αποκλίνει λίγο.)
 */
public void walkAwayFromHome()
{
    if(home == null) {
        // αν δεν έχουμε ένα σπίτι, δεν μπορούμε να προχωρήσουμε μακριά από αυτό.
        return;
    }
    if (randomChance(2)) {
        randomWalk();  // δεν μπορεί πάντα να περπατήσει ευθεία. 2% πιθανότητα να αποκλίνει από την πορεία.
    }
    else {
        headRoughlyTowards(home);   // πρώτα προσπάθεια περπατήματος προς το σπίτι...
        deltaX = -deltaX;           // ...και μετά περιστροφή 180 βαθμών
        deltaY = -deltaY;
        walk();
    }
}
/**
 * Προσαρμόζει την κατεύθυνση περπατήματος για να κοιτάξει προς τις δοθείσες συντεταγμένες.
 */
public void headTowards(Actor target)
{
    deltaX = capSpeed(target.getX() - getX());
    deltaY = capSpeed(target.getY() - getY());
}
/**
 * Περπάτημα προς τα εμπρός στην τρέχουσα κατεύθυνση με την τρέχουσα ταχύτητα.
 * (Δεν αλλάζει κατεύθυνση ή ταχύτητα.)
 */
public void walk()
{
    setLocation(getX() + deltaX, getY() + deltaY);
    setRotation((int) (180 * Math.atan2(deltaY, deltaX) / Math.PI));
}
/**
 * Προσαρμόζει την κατεύθυνση περπατήματος για να κοιτάξει προς κάποιες συγκεκριμένες συντεταγμένες.
 * Αυτό δεν κοιτάει πάντα προς την ίδια κατεύθυνση. Η κατεύθυνση είναι ελαφρώς τυχαία (αλλά πιθανά προς την κατεύθυνση
 * του στόχου) για να δείχνει πιο φυσική.
 */
private void headRoughlyTowards(Actor target)
{
    int distanceX = Math.abs(getX() - target.getX());
    int distanceY = Math.abs(getY() - target.getY());
    boolean moveX = (distanceX > 0) && (Greenfoot.getRandomNumber(distanceX + distanceY) < distanceX);
    boolean moveY = (distanceY > 0) && (Greenfoot.getRandomNumber(distanceX + distanceY) < distanceY);

    deltaX = computeHomeDelta(moveX, getX(), target.getX());
    deltaY = computeHomeDelta(moveY, getY(), target.getY());
}
```

```
    /**
     * Υπολογίζει και επιστρέφει την κατεύθυνση (διαφορά) που πρέπει να στρίψουμε όταν
     * είμαστε στο δρόμο για το σπίτι.
     */
    private int computeHomeDelta(boolean move, int current, int home)
    {
        if (move) {
            if (current > home)
                return -SPEED;
            else
                return SPEED;
        }
        else
            return 0;
    }
    /**
     * Προσαρμόζει τυχαία την ταχύτητα (ξεκινά, συνεχίζει ή επιβραδύνει). Η
     * ταχύτητα που επιστρέφεται είναι στο εύρος [-SPEED .. SPEED].
     */
    private int adjustSpeed(int speed)
    {
        speed = speed + Greenfoot.getRandomNumber(2 * SPEED - 1) - SPEED + 1;
        return capSpeed(speed);
    }
    /**
     * Βεβαιωθείτε ότι η ταχύτητα που επιστρέφεται είναι στο εύρος [-SPEED .. SPEED].
     */
    private int capSpeed(int speed)
    {
        if (speed < -SPEED)
            return -SPEED;
        else if (speed > SPEED)
            return SPEED;
        else
            return speed;
    }
    /**
     * Επιστροφή 'true' ακριβώς σε 'percent' αριθμό κλήσεων. Δηλαδή: ένα κάλεσμα
     * randomChance(25) έχει μια 25% πιθανότητα να επιστρέψει true.
     */
    private boolean randomChance(int percent)
    {
        return Greenfoot.getRandomNumber(100) < percent;
    }
}
```

```
import greenfoot.*;  // Εισαγωγή των απαραίτητων κλάσεων από τη βιβλιοθήκη Greenfoot
/**
 * Κλάση Ant που αναπαριστά ένα μυρμήγκι που συλλέγει φαγητό.
 *
 * Ο συγγραφέας της κλάσης είναι ο Michael Kölling και η έκδοση είναι 1.1.
 */
public class Ant extends Creature
{
    /** Κάθε πόσα βήματα μπορούμε να τοποθετήσουμε ένα σταγονίδιο φερομόνης. */
    private static final int MAX_PH_LEVEL = 18;

    /** Πόσο καιρό κρατάμε κατεύθυνση μετά τον εντοπισμό των φερομόνων. */
    private static final int PH_TIME = 30;

    /** Επισημαίνει αν έχουμε φαγητό μαζί μας. */
    private boolean carryingFood = false;

    /** Πόση φερομόνη έχουμε αυτή τη στιγμή. */
    private int pheromoneLevel = MAX_PH_LEVEL;

    /** Πόσο καλά θυμόμαστε την τελευταία φερομόνη - μεγαλύτερος αριθμός: πιο πρόσφατο */
    private int foundLastPheromone = 0;

    /**
     * Δημιουργία ενός μυρμηγκιού με δεδομένη φωλιά. Η αρχική ταχύτητα είναι μηδέν (δεν κινείται).
     */
    public Ant(AntHill home)
    {
        setHomeHill(home);
    }
    /**
     * Εκτέλεση των ενεργειών που πρέπει να κάνει ένα μυρμήγκι.
     */
    public void act()
    {
        if (carryingFood) {
            walkTowardsHome();
            handlePheromoneDrop();
            checkHome();
        }
        else {
            searchForFood();
        }
    }
```

```
/**
 * Περπάτημα ψάχνοντας φαγητό.
 */
private void searchForFood()
{
    if (foundLastPheromone > 0) { // αν μπορούμε ακόμα να θυμόμαστε...
        foundLastPheromone--;
        walkAwayFromHome();
    }
    else if (smellPheromone()) {
        walkTowardsPheromone();
    }
    else {
        randomWalk();
    }
    checkFood();
}
/**
 * Είμαστε σπίτι; Αφήστε το φαγητό αν είμαστε και ξεκινήστε να κινείστε προς τα έξω.
 */
private void checkHome()
{
    if (atHome()) {
        dropFood();
    }
}
/**
 * Είμαστε σπίτι;
 */
private boolean atHome()
{
    if (getHomeHill() != null) {
        return (Math.abs(getX() - getHomeHill().getX()) < 4) && (Math.abs(getY() - getHomeHill().getY()) < 4);
    }
    else {
        return false;
    }
}
/**
 * Υπάρχει κάποιο φαγητό εδώ όπου είμαστε; Αν ναι, πάρτε λίγο!
 */
public void checkFood()
{
    Food food = (Food) getOneIntersectingObject(Food.class);
    if (food != null) {
        takeFood(food);
    }
}
/**
 * Πάρτε λίγο φαγητό από ένα σωρό.
 */
private void takeFood(Food food)
{
    carryingFood = true;
    food.takeSome();
    setImage("ant-with-food.gif");
}
/**
 * Αφήστε το φαγητό μας στη φωλιά του μυρμηγκιού.
 */
private void dropFood()
{
    carryingFood = false;
    getHomeHill().countFood();
    setImage("ant.gif");
}
/**
 * Ελέγξτε αν μπορούμε να τοποθετήσουμε κάποια φερομόνη ακόμη. Αν μπορούμε, κάντε το.
 */
private void handlePheromoneDrop()
{
    if (pheromoneLevel == MAX_PH_LEVEL) {
        Pheromone ph = new Pheromone();
        getWorld().addObject(ph, getX(), getY());
        pheromoneLevel = 0;
    }
    else {
        pheromoneLevel++;
    }
}
/**
 * Ελέγξτε αν μπορούμε να μυρίσουμε φερομόνες. Αν μπορούμε, επιστρέψτε true, αλλιώς επιστρέψτε false.
 */
public boolean smellPheromone()
{
    Actor ph = getOneIntersectingObject(Pheromone.class);
    return (ph != null);
}
/**
 * Αν μπορούμε να μυρίσουμε φερομόνες, περπατήστε προς αυτές. Αν όχι, μην κάνετε τίποτα.
 */
public void walkTowardsPheromone()
{
    Actor ph = getOneIntersectingObject(Pheromone.class);
    if (ph != null) {
        headTowards(ph);
        walk();
        if (ph.getX() == getX() && ph.getY() == getY()) {
            foundLastPheromone = PH_TIME;
        }
    }
}
```

```java
import greenfoot.*;  // Εισαγωγή των απαραίτητων κλάσεων από τη βιβλιοθήκη Greenfoot

import java.util.Random;
/**
 * Ένα σωρό φαγητού. Αρχικά, ο σωρός αποτελείται από 100 κομμάτια φαγητού.
 *
 * Ο συγγραφέας της κλάσης είναι ο Michael Kölling και η έκδοση είναι 1.1.
 */
public class Food extends Actor
{
    private static final int SIZE = 30;
    private static final int HALFSIZE = SIZE / 2;
    private static final Color color1 = new Color(160, 200, 60);
    private static final Color color2 = new Color(80, 100, 30);
    private static final Color color3 = new Color(10, 50, 0);

    private static final Random randomizer = new Random();

    private int crumbs = 100;  // πλήθος κομματιών φαγητού σε αυτόν τον σωρό

    /**
     * Δημιουργία ενός σωρού φαγητού με μια εικόνα που απεικονίζει το ποσό.
     */
    public Food()
    {
        updateImage();
    }
    /**
     * Αφαίρεση κάποιου φαγητού από αυτόν τον σωρό φαγητού.
     */
    public void takeSome()
    {
        crumbs = crumbs - 3;
        if (crumbs <= 0)
        {
            getWorld().removeObject(this);
        }
        else
        {
            updateImage();
        }
    }
    /**
     * Ενημέρωση της εικόνας
     */
    private void updateImage()
    {
        GreenfootImage image = new GreenfootImage(SIZE, SIZE);
        for (int i = 0; i < crumbs; i++) {
            int x = randomCoord();
            int y = randomCoord();

            image.setColorAt(x, y, color1);
            image.setColorAt(x + 1, y, color2);
            image.setColorAt(x, y + 1, color2);
            image.setColorAt(x + 1, y + 1, color3);
        }
        setImage(image);
    }
    /**
     * Επιστρέφει ένα τυχαίο αριθμό σχετικά με το μέγεθος του σωρού φαγητού.
     */
    private int randomCoord()
    {
        int val = HALFSIZE + (int) (randomizer.nextGaussian() * (HALFSIZE / 2));

        if (val < 0)
            return 0;

        if (val > SIZE - 2)
            return SIZE - 2;
        else
            return val;
    }
}
```

```java
import greenfoot.*;  // Εισαγωγή των απαραίτητων κλάσεων από τη βιβλιοθήκη Greenfoot

/**
 * Οι φερομόνες πέφτουν από τις μυρμήγκιες όταν θέλουν να επικοινωνήσουν κάτι
 * με άλλες μυρμήγκιες.
 *
 * Ο συγγραφέας της κλάσης είναι ο Michael Kölling και η έκδοση είναι 1.1.
 */
public class Pheromone extends Actor
{
    private final static int MAX_INTENSITY = 180;
    private int intensity;

    /**
     * Δημιουργία ενός νέου σταγονιδίου φερομόνης με πλήρη ένταση.
     */
    public Pheromone()
    {
        intensity = MAX_INTENSITY;
        updateImage();
    }
    /**
     * Η φερομόνη μειώνει την έντασή της. Όταν η ένταση φτάσει στο μηδέν, εξαφανίζεται.
     */
    public void act()
    {
        intensity -= 1;
        if (intensity <= 0)
        {
            getWorld().removeObject(this);
        }
        else
        {
            if ((intensity % 4) == 0)
            {     // κάθε τέσσερα βήματα...
                updateImage();
            }
        }
    }
    /**
     * Δημιουργία της εικόνας. Το μέγεθος και η διαφάνεια είναι ανάλογα με την ένταση.
     */
    private void updateImage()
    {
        int size = intensity / 3 + 5;
        GreenfootImage image = new GreenfootImage(size + 1, size + 1);
        int alpha = intensity / 3;
        image.setColor(new Color(255, 255, 255, alpha));
        image.fillOval(0, 0, size, size);
        image.setColor(Color.DARK_GRAY);
        image.fillRect(size / 2, size / 2, 2, 2);   // μικρό κουκκίδι στη μέση
        setImage(image);
    }
}
```

Αναφορές

- https://www.youtube.com/@CodingClub
- https://www.youtube.com/watch?v=LT6IPr2W8dw (ανακτήθηκε στις 04/03/2023)