

PROJECT ΣΤΗΝ JAVA - 2η ΦΑΣΗ

Προσομοίωση ενός Οικοσυστήματος

EcoSim



Project Εαρινού εξαμήνου 2002 στο μάθημα HY252

Φάση 2η – Υλοποίηση

Ομάδα Εργασίας:

- Γεωργόπουλος Γιώργος A.M: 1340 ggeorgop@csd.uoc.gr
- Κριτσωτάκης Μανώλης A.M: 1395 kritsot@csd.uoc.gr
- Παπαδογιαννάκης Αντώνης A.M: 1415 papadog@csd.uoc.gr

ΑΝΑΦΟΡΑ 2^{ης} ΦΑΣΗΣ

1. Περιγραφή της Εργασίας

Το EcoSim (Ecosystem Simulator) είναι ένα πρόγραμμα το οποίο προσομοιώνει την εξέλιξη ενός οικοσυστήματος μέσα στο οποίο ζουν και αναπτύσσονται ταυτόχρονα διάφορα ήδη οργανισμών: σαρκοφάγα ζώα, φυτοφάγα ζώα και φυτά. Συγκεκριμένα, τα ζώα που υπάρχουν μέσα στο σύστημα είναι λιοντάρια, τίγρεις, αρκούδες, ζέβρες, ελέφαντες, καμηλοπαρδάλεις και τα φυτά, δέντρα και θάμνοι.

Ο χρήστης παρακολουθεί την εξέλιξη του οικοσυστήματος αλλά επεμβαίνει και σε αυτήν επιλέγοντας τους οργανισμούς που θα υπάρχουν στο οικοσύστημα και τη θέση στην οποία αυτοί εισάγονται αρχικά. Έτσι, πριν και κατά την διάρκεια της προσομοίωσης, ο χρήστης έχει την δυνατότητα να εισάγει οργανισμούς στο οικοσύστημα. Οι υπόλοιπες λειτουργίες που παρέχονται στον χρήστη είναι συνοπτικά:

- Λειτουργίες Προσομοίωσης, οι οποίες περιλαμβάνουν την έναρξη (start), το πάγωμα (pause), την συνέχιση (resume) και την επανεκκίνηση (restart) της προσομοίωσης.
- Λειτουργίες Πληροφοριών με τις οποίες δίνεται στον χρήστη η δυνατότητα να βλέπει πληροφορίες σχετικά με ένα πληθυσμό οργανισμών ή και σχετικά με ένα οργανισμό συγκεκριμένα.

Το περιβάλλον της εφαρμογής αποτελείται από 4 κυρίως μέρη. Το πρώτο και βασικό είναι ο χάρτης πάνω στον οποίο ζουν ή κινούνται όλοι οι οργανισμοί. Τα ζώα κινούνται πάνω στον χάρτη με τυχαίο τρόπο, ενώ και τα φυτά ζουν πάνω σε αυτόν. Οι οργανισμοί αλληλεπιδρούν μεταξύ τους, καθώς τα ζώα τρέφονται με φυτά ή με άλλα ζώα, ανάλογα με το είδος στο οποίο ανήκουν. Αν δεν συγκεντρώσουν την απαραίτητη τροφή ή φαγωθούν από άλλον οργανισμό οι οργανισμοί πεθαίνουν. Εκτός από το χάρτη, το περιβάλλον περιλαμβάνει μία μπάρα λειτουργιών από την οποία ο χρήστης μπορεί να επιλέγει τις λειτουργίες που προαναφέραμε καθώς και την μπάρα οργανισμών η οποία περιέχει όλους τους οργανισμούς και από την οποία αυτοί τοποθετούνται στο χάρτη. Τέλος, στο κάτω μέρος βρίσκονται τρεις περιοχές όπου εμφανίζονται οι απαραίτητες πληροφορίες.

Το EcoSim μπορεί να εκτελεστεί σαν εφαρμογή (Application) και σαν Applet μέσω του appletviewer ή οποιουδήποτε web browser.

2. Περιγραφή του Γραφικού Περιβάλλοντος (GUI)

Το EcoSim περιέχει ένα βασικό JFrame που είναι το βασικό παράθυρο της εφαρμογής, μέσα στο οποίο υπάρχουν όλα αυτά που αναφέραμε παραπάνω. Συγκεκριμένα, το βασικό παράθυρο του EcoSim περιέχει 4 κυρίως μέρη: Ένα JMenuBar που περιέχει όλες τις λειτουργίες που μπορεί να επιλέξει ο χρήστης, ένα JPanel με τα 8 εικονίδια των ζώων που μπορεί να εισάγει ο χρήστης, τον χάρτη του οικοσυστήματος και ένα ακόμα JPanel που αποτελείται από 3 μέρη και εκεί φαίνονται οι απαραίτητες πληροφορίες που χρειάζεται ο χρήστης.

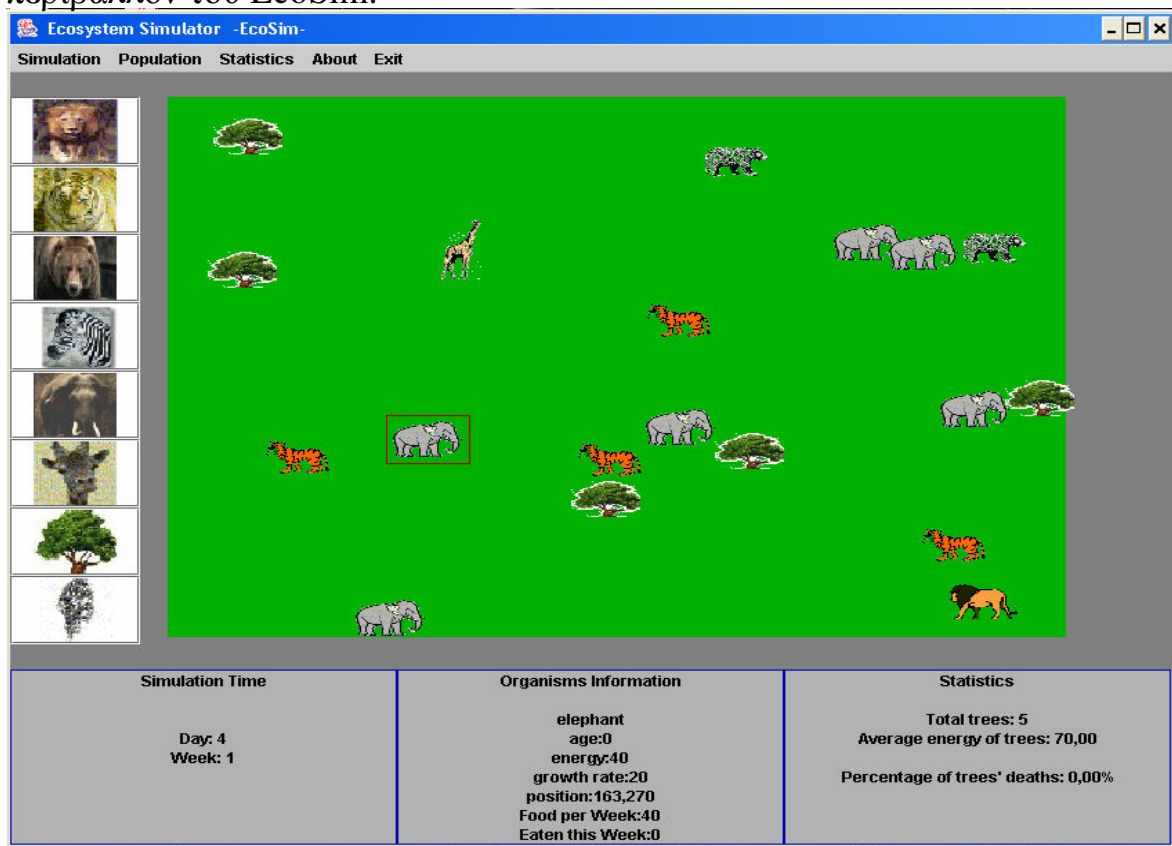
Στην κορυφή του παραθύρου βρίσκεται ένα JMenuBar το οποίο περιέχει 5 JMenu (Simulation, Population, Statistics, About, Exit). Το κάθε JMenu περιέχει κάποια JMenuItem τα οποία μπορεί να επιλέξει ο χρήστης ώστε να γίνει η κατάλληλη λειτουργία. Επιλέγοντας κάποιο JMenuItem ο ActionListener που υπάρχει στην βασική κλάση του EcoSim “πιάνει” τα αντίστοιχα events όπως τα έχουμε ονομάσει και καλεί την κατάλληλη μέθοδο από την κλάση Operations.

Στο αριστερό μέρος του παραθύρου βρίσκεται ένα JPanel με 8 JButtons που έχουν τα εικονίδια των 8 διαθέσιμων οργανισμών. Από εκεί ο χρήστης επιλέγει όποιον οργανισμό επιθυμεί για προσθήκη στο χάρτη και στο οικοσύστημα πατώντας το αντίστοιχο κουμπί και έπειτα το σημείο στον χάρτη που επιθυμεί να εισαχθεί ο οργανισμός. Επιλέγοντας ο χρήστης κάποιο από αυτά τα 8 κουμπιά ο ActionListener “πιάνει” το αντίστοιχο event και έπειτα αφήνει τον MouseListener του χάρτη να κάνει την εισαγωγή του οργανισμού στο σημείο το οποίο θα επιλέξει με πάτημα του ποντικιού ο χρήστης.

Στο κέντρο του βασικού παραθύρου του EcoSim υπάρχει ο χάρτης του οικοσυστήματος. Πάνω στον χάρτη, τα ζώα περιπλανιούνται ελεύθερα με το πέρασμα των ημερών, αλληλεπιδρούν μεταξύ τους και πεθαίνουν. Ο χάρτης υλοποιείται σε ξεχωριστή κλάση από αυτήν του υπόλοιπου περιβάλλοντος, η οποία κάνει κληρονομεί την JComponent και υλοποιεί το MouseListener interface. Ο χάρτης έχει σταθερά όρια (650x450) και αυτά παραμένουν σταθερά ακόμα και εάν το βασικό παράθυρο αλλάξει μέγεθος (resize). Έχει πράσινο φόντο και πάνω σε αυτόν ζωγραφίζονται όλα τα ζώα μέσω της μεθόδου paint (ή της repaint η οποία καλεί την paint) στην θέση την οποία βρίσκονται κάθε στιγμή. Τέλος, ο χάρτης έχει την δυνατότητα να “πιάνει” τα πατήματα του ποντικιού και το σημείο στο οποίο ο χρήστης πάτησε το ποντίκι (MouseListener). Έτσι, μπορούμε να εισάγουμε έναν οργανισμό στο σημείο που επέλεξε ο χρήστης, αν προηγουμένως είχε επιλέξει το

αντίστοιχο κουμπί από την μπάρα οργανισμών, και επίσης μπορούμε να επιλέξουμε έναν οργανισμό που υπάρχει στον χάρτη πατώντας το ποντίκι πάνω του για να βλέπουμε πληροφορίες για αυτόν καθώς αυτός κινείται και εξελίσσεται στον χάρτη.

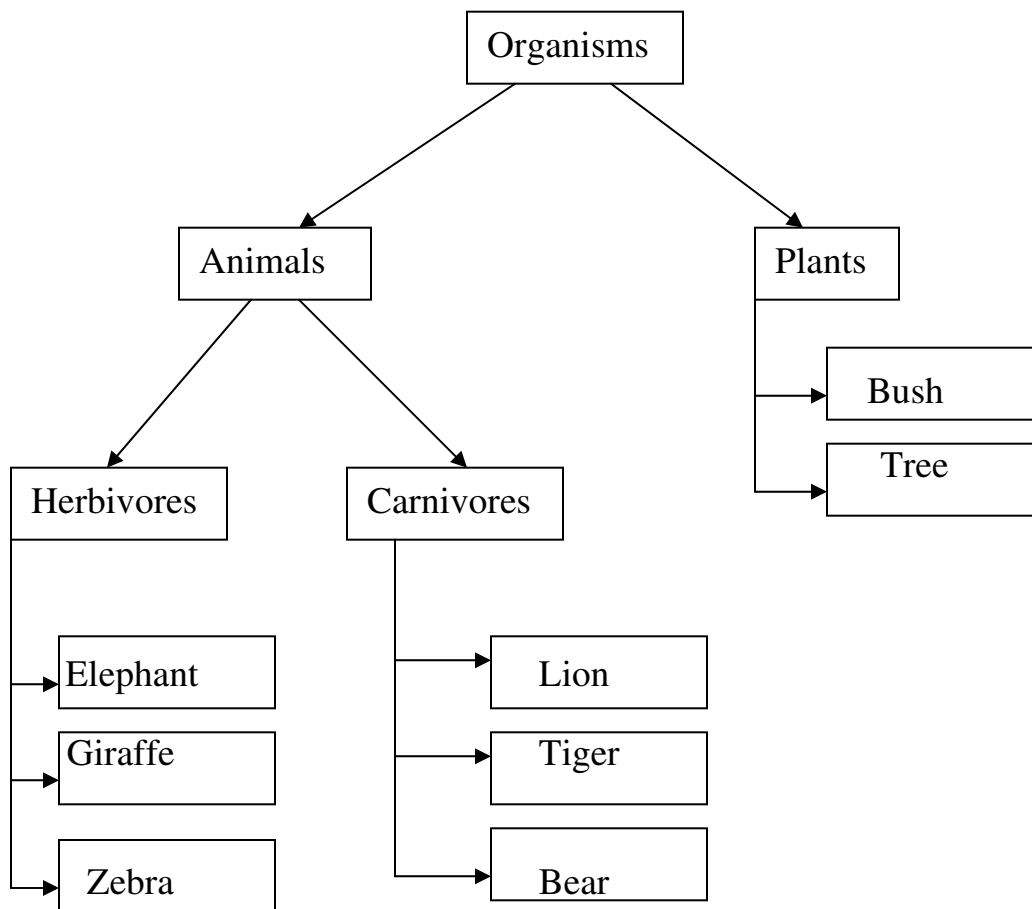
Τέλος, στο κάτω μέρος του παραθύρου βρίσκεται ένα JPanel που περιέχει τρία άλλα διαδοχικά JPanel και σε αυτά φαίνονται οι απαραίτητες πληροφορίες με την χρήση μερικών JLabel. Στο αριστερό JPanel φαίνεται ο χρόνος της προσομοίωσης σε μέρες και σε εβδομάδες, βλέπουμε δηλαδή σε ποια μέρα και ποια εβδομάδα βρισκόμαστε. Στο κεντρικό JPanel φαίνονται οι πληροφορίες για τον οργανισμό που έχουμε επιλέξει από τον χάρτη, αν έχουμε επιλέξει κάποιον. Οι πληροφορίες αυτές (ενέργεια, ηλικία, ρυθμός ανάπτυξης, θέση, τροφή ανά εβδομάδα και τροφή που έχει καταναλώσει αυτήν την εβδομάδα) ανανεώνονται κάθε μέρα, και όταν ο επιλεγμένος οργανισμός πεθάνει το JPanel αυτό αδειάζει. Τέλος, στο αριστερό JPanel βλέπουμε τα στατιστικά τα οποία έχουμε επιλέξει από το MenuBar για να δούμε (π.χ. Population, Statistics for lions, κτλ). Τα στατιστικά αυτά ανανεώνονται επίσης κάθε μέρα για να έχουν τις σωστές τιμές κάθε φορά που τα βλέπουμε. Παρακάτω βλέπουμε μια φωτογραφία με το γραφικό περιβάλλον του EcoSim.



Το γραφικό περιβάλλον του EcoSim

3. Περιγραφή των οργανισμών

Το οικοσύστημα που προσομοιώνουμε αποτελείται από διάφορα είδη οργανισμών. Όλοι οι οργανισμοί έχουν τέσσερα κύρια χαρακτηριστικά: την ηλικία τους, τον ρυθμό ανάπτυξής τους, την ενέργειά τους -μέγεθος- και την θέση στην οποία βρίσκονται. Η ηλικία αρχικά είναι μηδέν και μετριέται σε εβδομάδες. Ο ρυθμός ανάπτυξής είναι σταθερός και διαφορετικός για κάθε είδος οργανισμού. Η ενέργεια κάθε οργανισμού αρχικά είναι ένας αριθμός μεγαλύτερος του μηδέν που διαφέρει ανάλογα με το είδος οργανισμού και κατά την διάρκεια της προσομοίωσης αυτή μεταβάλλεται ανάλογα με τον συγκεκριμένο οργανισμό. Αυξάνεται και μειώνεται όσο είναι ο ρυθμός ανάπτυξης του οργανισμού. Αν κάποια στιγμή η ενέργεια ενός οργανισμού γίνει μικρότερη ή ίση με το μηδέν τότε ο οργανισμός αυτός πεθαίνει. Τέλος η θέση κάθε οργανισμού μας δείχνει σε ποιο μέρος του χάρτη βρίσκεται. Οι οργανισμοί αποτελούνται από δύο κύριες κατηγορίες: Τα φυτά και τα ζώα. Τα φυτά τρέφονται κάθε εβδομάδα αυτόματα με ήλιο και νερό που υπάρχουν άφθονα και έτσι αυξάνεται κάθε εβδομάδα η ενέργειά τους κατά τον ρυθμό ανάπτυξης του κάθε φυτού. Αν ένα χορτοφάγο ζώο όμως φάει από ένα φυτό, η ενέργεια αυτού του φυτού μειώνεται κατά τον ρυθμό ανάπτυξής του. Τα ζώα κινούνται μέσα στο οικοσύστημα και ψάχνουν για τροφή. Πρέπει να φάνε ένα ορισμένο ποσό τροφής για να αυξήσουν την ενέργειά τους κατά τον ρυθμό ανάπτυξης τους κάθε εβδομάδα. Αλλιώς η ενέργειά τους μειώνεται κατά το ίδιο ποσό. Η κίνηση των ζώων γίνεται με τυχαίο τρόπο. Τα ζώα χωρίζονται σε χορτοφάγα και φυτοφάγα. Τα χορτοφάγα μπορούν να τραφούν μόνο από τα φυτά του οικοσυστήματος, τρώγοντας από αυτά τροφή ίση με τον ρυθμό ανάπτυξης του κάθε φυτού, όταν τα πλησιάσουν σε μικρή απόσταση. Τα σαρκοφάγα τρέφονται μόνο με άλλα ζώα, φυτοφάγα ή σαρκοφάγα, τα οποία έχουν το μισό ή λιγότερο μέγεθος από το δικό τους και τα έχουν πλησιάσει σε μικρή απόσταση. Η τροφή που παίρνουν από αυτά ισούται και πάλι με τον ρυθμό ανάπτυξης του θύματος, του οποίου η ενέργεια μειώνεται κατά τον ίδιο αριθμό. Τα φυτά που εμείς θα έχουμε διαθέσιμα για εισαγωγή στο οικοσύστημά μας είναι θάμνοι και δέντρα. Τα φυτοφάγα ζώα θα είναι ζέβρες, ελέφαντες και καμηλοπαρδάλεις. Τα σαρκοφάγα θα είναι λιοντάρια, τίγρεις και αρκούδες.. Η ιεραρχία των οργανισμών του οικοσυστήματός μας όπως την περιγράψαμε φαίνεται παρακάτω.



Οι ρυθμοί ανάπτυξης, η αρχική ενέργεια και η ποσότητα του φαγητού που πρέπει να τρώνε ανά εβδομάδα (τα ζώα μόνο) αυτών των οργανισμών φαίνονται στον παρακάτω πίνακα:

Είδος οργανισμού	Ρυθμός ανάπτυξης	Αρχική ενέργεια	Φαγητό ανά εβδομάδα
Lion	30	70	40
Tiger	20	50	35
Bear	30	70	40
Zebra	15	20	30
Giraffe	20	25	35
Elephant	20	40	40
Tree	25	100	-
Bush	10	25	-

4. Ανάλυση των λειτουργιών

4.1 Δημιουργία Νέου Οργανισμού

Ο χρήστης μέσω αυτής της λειτουργίας μπορεί να προσθέτει στον χάρτη έναν νέο οργανισμό, που αυτός επιλέγει, σε οποιαδήποτε θέση θέλει. Τα εικονίδια των οργανισμών που μπορεί να δημιουργεί ο χρήστης φαίνονται στο αριστερό κομμάτι της οθόνης της προσομοίωσης (μπάρα οργανισμών). Εκεί υπάρχουν όλοι οι οργανισμοί που το πρόγραμμα μας προσομοιώνει.

Η επιλογή του Οργανισμού που θέλουμε να δημιουργήσουμε γίνεται πατώντας πάνω στο εικονίδιο του επιθυμητού οργανισμού. Στη συνέχεια ο χρήστης πρέπει να επιλέγει τη θέση μέσα στο χάρτη όπου θέλει να τοποθετηθεί ο οργανισμός. Αυτό γίνεται πατώντας πάνω στο επιθυμητό μέρος του χάρτη όπου και στη συνέχεια εμφανίζεται ένα νέο εικονίδιο που αφορά τον οργανισμό. Προϋπόθεση είναι ότι στη θέση που εισάγουμε το νέο οργανισμό δεν υπάρχει ήδη άλλος οργανισμός.

4.2 Λειτουργίες Προσομοιώσεις

Μέσω αυτών των λειτουργιών ο χρήστης μπορεί να παρακολουθήσει την κίνηση των οργανισμών και την αλληλεπίδραση μεταξύ τους. Κάθε οργανισμός (εκτός των φυτών) σε καθημερινή βάση (χρόνο που έχουμε καθορίσει εμείς να αντιστοιχεί σε μια μέρα) θα τρώει τον οργανισμό που βρίσκεται πιο κοντά σε αυτόν εφόσον η απόστασή τους είναι αρκετά μικρή (η μεγαλύτερη δυνατή απόσταση που ένα ζώο μπορεί να τρώει ένα άλλο έχει καθοριστεί πάλι από εμάς) και ανάλογα αν η ενέργεια του και το είδος του το επιτρέπει. Ο οργανισμός ο οποίος επιτίθεται θα αυξάνει την ποσότητα του φαγητού του την τρέχουσα εβδομάδα. Αντίστοιχα ο οργανισμός ο οποίος θα αποτελεί την τροφή του πρώτου οργανισμού θα χάνει ενέργεια. Επιπλέον, όταν κάποιος οργανισμός φτάσει να έχει μηδενική ενέργεια, τότε θα πεθαίνει και το εικονίδιο που αντιστοιχεί στο συγκεκριμένο οργανισμό θα εξαφανίζεται από τον χάρτη. Σε εβδομαδιαία βάση πλέον ο κάθε οργανισμός θα ανακτά η θα χάνει ποσά ενεργείας που θα αναλογούν στον ρυθμό ανάπτυξης του κι επιπλέον θα αυξάνεται η ηλικία του κατά μια ποσοστιαία μονάδα. Οι διαθέσιμες λειτουργίες προσομοίωσης είναι οι εξής:

4.2.1 Έναρξη προεπιλεγμένου οικοσυστήματος(Start Preconstructed Simulation)

Μέσω αυτής της λειτουργίας ο χρήστης μπορεί να αρχίζει την προσομοίωση ενός έτοιμου οικοσυστήματος από 20 τυχαίους οργανισμούς που είναι τοποθετημένοι τυχαία κάθε φορά στο χάρτη της προσομοίωσης. Η λειτουργία αυτή πραγματοποιείται μέσω της επιλογής ‘Start Preconstructed Simulation’ από την επιλογή ‘Simulation’ του μενού. Η λειτουργία αυτή μπορεί να επιλεγθεί ανά πάσα στιγμή χωρίς κανένα περιορισμό. Επιπλέον ο χρήστης από εκεί και πέρα θα μπορεί να προσθέτει τους οργανισμούς της επιλογής του σε θέσεις πάλι της επιλογής του.

4.2.2 Έναρξη (Start Simulation)

Μέσω αυτής της λειτουργίας ο χρήστης μπορεί να αρχίζει την προσομοίωση. Ουσιαστικά όμως, για να αρχίσει η προσομοίωση, θα πρέπει πρώτα ο χρήστης να έχει δημιουργήσει οργανισμούς και να τους έχει εισάγει στο χάρτη . Διαφορετικά δε θα υπάρχει ορατό αποτέλεσμα με την έναρξη της προσομοίωσης. Η λειτουργία αυτή πραγματοποιείται μέσω της επιλογής ‘Start Simulation’ που βρίσκεται στο μενού ‘Simulation’ στη πάνω μεριά της οθόνης και συγκεκριμένα στη Μπάρα Λειτουργιών. Από εκεί ο χρήστης επιλέγει αυτή τη λειτουργία και η προσομοίωση ξεκινάει. Ο χρήστης μπορεί να προσθέτει νέους οργανισμούς και μετά την έναρξη της προσομοίωσης με τον ίδιο τρόπο που περιγράφηκε. Πρέπει να δοθεί σημασία στο γεγονός ότι η αντίστοιχη επιλογή έχει νόημα μόνο την πρώτη φορά που ο χρήστης την επιλέγει. Από εκεί και πέρα δε θα υπάρχει καμία διάφορα όσες φορές κι αν γίνει η επιλογή.

4.2.3 Πάγωμα (Pause Simulation)

Επιπλέον ο χρήστης θα μπορεί να παγώνει την προσομοίωση. Αντίστοιχα αυτό θα γίνεται πατώντας το κουμπί ‘Pause Simulation’ από την επιλογή ‘Simulation’. Η προσομοίωση θα σταματάει προσωρινά και ο χρήστης θα μπορεί να εισάγει νέους οργανισμούς με τον ίδιο τρόπο που περιγράφηκε πριν. Φυσικά η λειτουργία αυτή έχει νόημα μόνο εφόσον προηγουμένως έχει επιλέγει η λειτουργία που αντιστοιχεί στην έναρξη ή στην επανεκκίνηση της προσομοίωσης. Διαφορετικά η επιλογή δεν επιφέρει καμία αλλαγή στην προσομοίωση.

4.2.4 Συνέχιση (Resume Simulation)

Με αυτή τη λειτουργία το πρόγραμμα μας θα συνεχίζει την προσομοίωση από εκεί που σταμάτησε πριν με το πάτημα του κουμπιού 'Pause Simulation'. Η συνέχιση της προσομοίωσης θα γίνεται πατώντας πάνω στο κουμπί 'Resume Simulation' από το μενού και συγκεκριμένα από την επιλογή 'Simulation'. Ουσιαστικά αυτή η λειτουργία θα έχει νόημα μόνο εφόσον προηγουμένως έχει επιλέξει η λειτουργία που αντιστοιχεί στο πάγωμα της προσομοίωσης. Διαφορετικά πάλι δε θα υπάρχει καμία αλλαγή στην προσομοίωση.

4.2.5 Επανεκκίνηση (Restart Simulation)

Εδώ ο χρήστης θα μπορεί να σβήνει οτιδήποτε είχε πριν εισάγει στον χάρτη και να ξεκινάει μια νέα προσομοίωση. Όλοι οι οργανισμοί που βρίσκονται στο χάρτη καθώς και οι πληροφορίες που βρίσκονται στο κάτω μέρος της οθόνης της προσομοίωσης θα χάνονται και θα ξεκινάει μια εκ νέου προσομοίωση. Η λειτουργία αυτή θα γίνεται πατώντας το κουμπί 'Restart Simulation' από την επιλογή 'Simulation' του μενού. Από εκεί και πέρα θα ισχύουν τα ίδια με τη λειτουργία 'Start Simulation'.

4.3 Πληροφορίες πληθυσμού οργανισμών (Show Population)

Με αυτή τη λειτουργία ο χρήστης βλέπει στο κάτω δεξιά τμήμα των στατιστικών-πληροφοριών αναλυτικά τον πληθυσμό του οικοσυστήματος την κάθε χρονική στιγμή. Βλέπει δηλαδή πόσοι οργανισμοί από κάθε είδος υπάρχουν στο οικοσύστημα. Για να επιλεγεί αυτή η λειτουργία ο χρήστης πρέπει να επιλέξει 'Show Population' από το μενού 'Population' στο Menu Bar του κεντρικού παραθύρου.

4.4 Πληροφορίες ανά πληθυσμό οργανισμών (Statistics)

Με αυτή τη λειτουργία ο χρήστης μπορεί να δει γενικές πληροφορίες που αφορούν την κατηγορία οργανισμών που αυτός επιλέγει από το Menu Bar και συγκεκριμένα από το υπο-μενού 'Statistics'. Επιλέγει την κατηγορία οργανισμών του οποίου θέλει να δει πληροφορίες, πατώντας πάνω στην επιλογή που του αντιστοιχεί στη λίστα που προκύπτει επιλέγοντας 'Statistics' από το μενού. Οι πληροφορίες που εμφανίζονται αφορούν

στατιστικά, όπως το μέσο όρο ενέργειας των οργανισμών που είναι ζωντανοί την τρέχουσα στιγμή, τη μέση κατανάλωση τροφής την τελευταία εβδομάδα, ποσοστά θανάτων σε σχέση με το σύνολο των οργανισμών του χάρτη, αλλά και τον αριθμό των ζωντανών οργανισμών του είδους που επιλέχθηκε την τρέχουσα στιγμή. Οι πληροφορίες αυτές φαίνονται στο κάτω μέρος της οθόνης και συγκεκριμένα στο δεξιό τμήμα της και ανανεώνονται καθημερινά.

4.5 Πληροφορίες ανά οργανισμό

Επιπλέον ο χρήστης μπορεί να βλέπει πληροφορίες που αφορούν αποκλειστικά κάποιον οργανισμό και όχι την ευρύτερη κατηγορία στη οποία ανήκει. Αυτό είναι εφικτό μέσω της επιλογής κάποιου συγκεκριμένου οργανισμού από το χάρτη της προσομοίωσης. Αυτό γίνεται πατώντας πάνω στο εικονίδιο του επιθυμητού οργανισμού μέσα στο χάρτη όποτε και το αντίστοιχο εικονίδιο θα τοποθετείται σε ένα ορθογώνιο πλαίσιο, δείχνοντας ποιον οργανισμό αφορούν οι συγκεκριμένες πληροφορίες. Στη συνέχεια πληροφορίες που αφορούν χαρακτηριστικά όπως την ηλικία, το μέγεθος, τον βαθμό ανάπτυξης, την ποσότητα τροφής που χρειάζεται ανά εβδομάδα ο οργανισμός για να αναπτυχθεί την ποσότητα τροφής που έχει καταναλώσει ο οργανισμός την τρέχουσα εβδομάδα, καθώς και τη θέση του θα εμφανίζονται σε ένα πλαίσιο στο κάτω μέρος της οθόνης και συγκεκριμένα στη μέση. Οι πληροφορίες αυτές θα αλλάζουν συνεχώς καθώς ο οργανισμός θα κινείται στο χάρτη.

4.6 Έξοδος από την εφαρμογή (Exit Application)

Ο χρήστης επιλέγοντας 'Exit' από το μενού και στη συνέχεια 'Exit Application' μπορεί αν πάσα στιγμή να σταματήσει την εφαρμογή και να κλείσει το παράθυρο της προσομοίωσης.

5. Περιγραφή των κλάσεων και των packages που υλοποιούνται

5.1 To package organisms

Σε αυτό το package υπάρχουν όλες οι κλάσεις που υλοποιούν τους οργανισμούς που θα έχουμε στο οικοσύστημά μας. Το package organisms περιέχει τις αφηρημένες κλάσεις organisms, animals, plants, carnivores, herbivores και τις συγκεκριμένες κλάσεις lion, tiger, bear, elephant, giraffe, zebra, bush και tree. Η ιεραρχία αυτών των κλάσεων συμπίπτει με την ιεραρχία των οργανισμών όπως την περιγράψαμε στο 3 και φαίνεται στο σχήμα που κάναμε εκεί. Οι κλάσεις animals και plants είναι υποκλάσεις της organisms, οι κλάσεις carnivores και herbivores είναι υποκλάσεις της animals, οι τελικές κλάσεις bush και tree είναι υποκλάσεις της plants, οι τελικές κλάσεις lion, tiger και bear είναι υποκλάσεις της carnivores και τέλος οι τελικές κλάσεις elephant, giraffe και zebra είναι υποκλάσεις της herbivores. Παρακάτω περιγράφονται αναλυτικά όλες οι παραπάνω κλάσεις που αναφέραμε.

5.1.1 Η κλάση organisms

Είναι αφηρημένη κλάση και όλοι οι οργανισμοί βρίσκονται κάτω από αυτήν στην ιεραρχία των κλάσεων. Είναι δηλαδή υπερκλάση όλων των υπολοίπων κλάσεων που υπάρχουν σε αυτό το package είτε άμεσα είτε έμμεσα. Τα πεδία αυτής της κλάσης είναι:

- age: (protected int age) Η ηλικία του οργανισμού.
- energy: (protected int energy) Η ενέργεια του οργανισμού.
- growth_rate: (protected int growth_rate) Ο ρυθμός ανάπτυξης του οργανισμού.
- position: (protected Position position) Η θέση του οργανισμού στον χάρτη.
- isChosen: (protected boolean isChosen) Εάν ο οργανισμός είναι επιλεγμένος ή όχι.
- im: (protected ImageIcon im) Το εικονίδιο του οργανισμού.
- alive_organisms: (protected static int alive_organisms) Πόσοι οργανισμοί είναι ζωντανοί συνολικά – στατική μεταβλητή.
- dead_organisms: (protected static int dead_organisms) Πόσοι οργανισμοί έχουν πεθάνει συνολικά – στατική μεταβλητή.

Τα πεδία της κλάσης organisms είναι protected μεταβλητές επειδή πρέπει να είναι κρυφά από το έξω κόσμο αλλά να κληρονομούνται στις υποκλάσεις της organisms. Οι μέθοδοι της κλάσης organisms είναι:

- public organisms() : Constructor – καλείται κατά την δημιουργία ενός νέου οργανισμού.
- public abstract void draw(Graphics g) : Abstract, Accesor-Selector. Ζωγραφίζει τον οργανισμό στον χάρτη με την κατάλληλη εικόνα.
- public boolean isAlive() : Accesor-Observer. Επιστρέφει true αν ο οργανισμός είναι ζωντανός, αλλιώς false.
- public int age() : Accesor-Selector. Επιστρέφει την ηλικία του οργανισμού.
- public int energy() : Accesor-Selector. Επιστρέφει την ενέργεια του οργανισμού.
- public int growth_rate() : Accesor-Selector. Επιστρέφει τον ρυθμό ανάπτυξης του οργανισμού.
- public Position position() : Accesor-Selector. Επιστρέφει την θέση του οργανισμού.
- public int alive_organisms() : Accesor-Selector. Επιστρέφει το πλήθος των ζωντανών οργανισμών.
- public int dead_organisms() : Accesor-Selector. Επιστρέφει το πλήθος των νεκρών οργανισμών.
- public void increase_age() : Transformer-Mutative. Αυξάνει την ηλικία κατά ένα.
- public void increase_energy() : Transformer-Mutative. Αυξάνει την ενέργεια κατά growth_rate(ρυθμός ανάπτυξης).
- public void decrease_energy() : Transformer-Mutative. Μειώνει την ενέργεια κατά growth_rate(ρυθμός ανάπτυξης).
- public abstract void restoreEnergy() : Abstract, Transformer-Mutative. Ανανεώνει την ενέργεια ανάλογα με το είδος του οργανισμού.
- public abstract void die() : Abstract, Transformer-Mutative. Καλείται όταν πεθαίνει ένας οργανισμός και αυξομειώνει τις στατικές μεταβλητές για το πλήθος των ζωντανών και νεκρών οργανισμών.
- public void choose(boolean c) : Transformer-Mutative. Κανονίζει αν ο οργανισμός είναι επιλεγμένος ή όχι.
- public static void initStats() : Transformer-Mutative. Αρχικοποιεί τα στατιστικά.

5.1.2 Η κλάση **animals**

Είναι αφηρημένη και κάθε ζώο βρίσκεται κάτω από αυτήν στην ιεραρχία των κλάσεων. Είναι υποκλάση της κλάσης **organisms**. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- **foodPerWeek** : (protected final int foodPerWeek) Το φαγητό που πρέπει να φάει το ζώο ανά εβδομάδα.
- **eatenThisWeek** : (protected int eatenThisWeek) Το φαγητό που έχει φάει το ζώο μέχρι στιγμής αυτήν την εβδομάδα.
- **alive_animals** : (protected static int alive_animals) Πόσα ζώα είναι ζωντανά συνολικά – στατική μεταβλητή.
- **dead_animals** : (protected static int dead_animals) Πόσα ζώα είναι νεκρά συνολικά – στατική μεταβλητή.

Τα πεδία της κλάσης **animals** είναι **protected** μεταβλητές επειδή πρέπει να είναι κρυφά από το έξω κόσμο αλλά να κληρονομούνται στις υποκλάσεις της **animals**. Οι μέθοδοι της κλάσης **animals** είναι:

- **public animals()** : Constructor – καλείται κατά την δημιουργία ενός νέου ζώου.
- **public int foodPerWeek()** : Accesor-Selector. Επιστρέφει το φαγητό που πρέπει να φάει το ζώο ανά εβδομάδα.
- **public int eatenThisWeek()** : Accesor-Selector. Επιστρέφει το φαγητό που έχει φάει το ζώο μέχρι στιγμής.
- **public int alive_animals()** : Accesor-Selector. Επιστρέφει το πλήθος των ζωντανών ζώων.
- **public int dead_animals()** : Accesor-Selector. Επιστρέφει το πλήθος των νεκρών ζώων.
- **public void reset_eatenThisWeek()** : Transformer-Mutative. Μηδενίζει το φαγητό που έχει φάει το ζώο.
- **public void increase_eatenThisWeek(int amount)** : Transformer-Mutative. Αυξάνει το φαγητό που έχει φάει το ζώο κατά amount.
- **public void restoreEnergy()** : Transformer-Mutative. Υλοποιεί την **restoreEnergy()** της **organisms**. Ανανεώνει κατάλληλα την ενέργεια του ζώου.
- **public void move(int x, int y)** : Transformer-Mutative. Αλλάζει την θέση του ζώου.
- **public abstract void eat(organisms o)** : Abstract, Transformer-Mutative. Ελέγχει αν το ζώο τρώει τον οργανισμό **o** και αν τον τρώει αυξάνει κατάλληλα το **eatenThisWeek** του ζώου και μειώνει το **energy** του οργανισμού **o**.

- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.1.3 Η κλάση `plants`

Είναι αφηρημένη και κάθε φυτό βρίσκεται κάτω από αυτήν στην ιεραρχία των κλάσεων. Είναι υποκλάση της κλάσης `organisms`. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- `alive_plants : (protected static int alive_plants)` Πόσα φυτά είναι ζωντανά συνολικά – στατική μεταβλητή.
- `dead_plants : (protected static int dead_plants)` Πόσα φυτά είναι νεκρά συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης `plants` είναι:

- `public plants () : Constructor` – καλείται κατά την δημιουργία ενός νέου φυτού.
- `public int alive_plants() : Accesor-Selector`. Επιστρέφει το πλήθος των ζωντανών φυτών.
- `public int dead_plants() : Accesor-Selector`. Επιστρέφει το πλήθος των νεκρών φυτών.
- `public void restoreEnergy() : Transformer-Mutative`. Υλοποιεί την `restoreEnergy()` της `organisms`. Ανανεώνει κατάλληλα την ενέργεια του φυτού.
- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.1.4 Η κλάση `carnivores`

Είναι αφηρημένη και κάθε σαρκοφάγο ζώο βρίσκεται κάτω από αυτήν στην ιεραρχία των κλάσεων. Είναι υποκλάση της κλάσης `animals`. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- `alive_carnivores : (protected static int alive_carnivores)` Πόσα σαρκοφάγα ζώα είναι ζωντανά συνολικά – στατική μεταβλητή.
- `dead_carnivores: (protected static int dead_carnivores)` Πόσα σαρκοφάγα ζώα είναι νεκρά συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης `carnivores` είναι:

- `public carnivores() : Constructor` – καλείται κατά την δημιουργία ενός νέου σαρκοφάγου ζώου.

- `public int alive_carnivores() : Accesor-Selector`. Επιστρέφει το πλήθος των ζωντανών σαρκοφάγων ζώων.
- `public int dead_carnivores() : Accesor-Selector`. Επιστρέφει το πλήθος των νεκρών σαρκοφάγων ζώων.
- `public boolean eat(organisms o) : Transformer-Mutative`. Υλοποιεί την `eat()` της `animals`. Ελέγχει αν το σαρκοφάγο ζώο τρώει τον οργανισμό `o` και αν τον τρώει αυξάνει κατάλληλα το `eatenThisWeek` του ζώου και μειώνει το `energy` του οργανισμού `o`.
- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.1.5 Η κλάση `herbivores`

Είναι αφηρημένη και κάθε φυτοφάγο ζώο βρίσκεται κάτω από αυτήν στην ιεραρχία των κλάσεων. Είναι υποκλάση της κλάσης `animals`. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- `alive_herbivores : (protected static int alive_herbivores)` Πόσα φυτοφάγα ζώα είναι ζωντανά συνολικά – στατική μεταβλητή.
- `dead_herbivores: (protected static int dead_herbivores)` Πόσα φυτοφάγα ζώα είναι νεκρά συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης `herbivores` είναι:

- `public herbivores () : Constructor` – καλείται κατά την δημιουργία ενός νέου φυτοφάγου ζώου.
- `public int alive_herbivores() : Accesor-Selector`. Επιστρέφει το πλήθος των ζωντανών φυτοφάγων ζώων.
- `public int dead_herbivores() : Accesor-Selector`. Επιστρέφει το πλήθος των νεκρών φυτοφάγων ζώων.
- `public boolean eat(organisms o) : Transformer-Mutative`. Υλοποιεί την `eat()` της `animals`. Ελέγχει αν το φυτοφάγο ζώο τρώει τον οργανισμό `o` και αν τον τρώει αυξάνει κατάλληλα το `eatenThisWeek` του ζώου και μειώνει το `energy` του οργανισμού `o`.
- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.1.6 Η κλάση lion

Είναι τελική κλάση από την οποία δημιουργούνται όλα τα λιοντάρια. Είναι υποκλάση της κλάσης carnivores. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- alive_lions : (protected static int alive_lions) Πόσα λιοντάρια είναι ζωντανά συνολικά – στατική μεταβλητή.
- dead_lions : (protected static int dead_lions) Πόσα λιοντάρια είναι νεκρά συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης lions είναι:

- public lion(int x, int y, ImageIcon im) : Constructor – καλείται κατά την δημιουργία ενός νέου λιονταριού.
- public int alive_lions(): Accesor-Selector. Επιστρέφει το πλήθος των ζωντανών λιονταριών.
- public int dead_lions(): Accesor-Selector. Επιστρέφει το πλήθος των νεκρών λιονταριών.
- public void draw(Graphics g) : Accesor-Selector. Υλοποιεί την draw(Graphics g) της organisms. Ζωγραφίζει μια εικόνα ενός λιονταριού στην θέση που αυτό βρίσκεται πάνω στον χάρτη.
- public void die() : Transformer-Mutative. Υλοποιεί την die() της organisms. Καλείται όταν πεθαίνει ένα λιοντάρι και μειώνει τις στατικές μεταβλητές για τους ζωντανούς οργανισμούς ενώ αυξάνει τις στατικές μεταβλητές για τους νεκρούς οργανισμούς.
- public static void initStats() : Transformer-Mutative. Αρχικοποιεί τα στατιστικά.

5.1.7 Η κλάση tiger

Είναι τελική κλάση από την οποία δημιουργούνται όλες οι τίγρεις. Είναι υποκλάση της κλάσης carnivores. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- alive_tigers : (protected static int alive_tigers) Πόσες τίγρεις είναι ζωντανές συνολικά – στατική μεταβλητή.
- dead_tigers : (protected static int dead_tigers) Πόσες τίγρεις είναι νεκρές συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης tigers είναι:

- public tiger(int x, int y, ImageIcon im) : Constructor – καλείται κατά την δημιουργία μιας νέας τίγρης.

- `public int alive_ tigers():` Accesor-Selector. Επιστρέφει το πλήθος των ζωντανών τίγρεων.
- `public int dead_ tigers():` Accesor-Selector. Επιστρέφει το πλήθος των νεκρών τίγρεων.
- `public void draw(Graphics g) :` Accesor-Selector. Υλοποιεί την `draw(Graphics g)` της `organisms`. Ζωγραφίζει μια εικόνα μιας τίγρης στην θέση που αυτή βρίσκεται πάνω στον χάρτη.
- `public void die() :` Transformer-Mutative. Υλοποιεί την `die()` της `organisms`. Καλείται όταν πεθαίνει μια τίγρη και μειώνει τις στατικές μεταβλητές για τους ζωντανούς οργανισμούς ενώ αυξάνει τις στατικές μεταβλητές για τους νεκρούς οργανισμούς.
- `public static void initStats() :` Transformer-Mutative. Αρχικοποιεί τα στατιστικά.

5.1.8 Η κλάση `bear`

Είναι τελική κλάση από την οποία δημιουργούνται όλες οι αρκούδες. Είναι υποκλάση της κλάσης `carnivores`. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- `alive_bears :` (`protected static int alive_ bears`) Πόσες αρκούδες είναι ζωντανές συνολικά – στατική μεταβλητή.
- `dead_bears :` (`protected static int dead_ bears`) Πόσες αρκούδες είναι νεκρές συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης `bears` είναι:

- `public bear (int x, int y, ImageIcon im) :` Constructor – καλείται κατά την δημιουργία μιας νέας αρκούδας.
- `public int alive_ bears():` Accesor-Selector. Επιστρέφει το πλήθος των ζωντανών αρκούδων.
- `public int dead_ bears():` Accesor-Selector. Επιστρέφει το πλήθος των νεκρών αρκούδων.
- `public void draw(Graphics g) :` Accesor-Selector. Υλοποιεί την `draw(Graphics g)` της `organisms`. Ζωγραφίζει μια εικόνα μιας αρκούδας στην θέση που αυτή βρίσκεται πάνω στον χάρτη.
- `public void die() :` Transformer-Mutative. Υλοποιεί την `die()` της `organisms`. Καλείται όταν πεθαίνει μια αρκούδα και μειώνει τις στατικές μεταβλητές για τους ζωντανούς οργανισμούς ενώ αυξάνει τις στατικές μεταβλητές για τους νεκρούς οργανισμούς.

- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.1.9 Η κλάση **zebra**

Είναι τελική κλάση από την οποία δημιουργούνται όλες οι ζέβρες. Είναι υποκλάση της κλάσης `herbivores`. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- `alive_zebra : (protected static int alive_ zebras)` Πόσες ζέβρες είναι ζωντανές συνολικά – στατική μεταβλητή.
- `dead_ zebras: (protected static int dead_ zebras)` Πόσες ζέβρες είναι νεκρές συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης `zebras` είναι:

- `public zebras(int x, int y, ImageIcon im) : Constructor` – καλείται κατά την δημιουργία μιας νέας ζέβρας.
- `public int alive_ zebras(): Accesor-Selector`. Επιστρέφει το πλήθος των ζωντανών ζέβρων.
- `public int dead_ zebras(): Accesor-Selector`. Επιστρέφει το πλήθος των νεκρών ζέβρων.
- `public void draw(Graphics g) : Accesor-Selector`. Υλοποιεί την `draw(Graphics g)` της `organisms`. Ζωγραφίζει μια εικόνα μιας ζέβρας στην θέση που αυτή βρίσκεται πάνω στον χάρτη.
- `public void die() : Transformer-Mutative`. Υλοποιεί την `die()` της `organisms`. Καλείται όταν πεθαίνει μια ζέβρα και μειώνει τις στατικές μεταβλητές για τους ζωντανούς οργανισμούς ενώ αυξάνει τις στατικές μεταβλητές για τους νεκρούς οργανισμούς.
- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.1.10 Η κλάση **elephant**

Είναι τελική κλάση από την οποία δημιουργούνται όλες οι ελέφαντες. Είναι υποκλάση της κλάσης `herbivores`. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- `alive_elephants : (protected static int alive_ elephants)` Πόσοι ελέφαντες είναι ζωντανοί συνολικά – στατική μεταβλητή.
- `dead_ elephants: (protected static int dead_ elephants)` Πόσοι ελέφαντες είναι νεκροί συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης zebras είναι:

- `public elephant(int x, int y, ImageIcon im) : Constructor` – καλείται κατά την δημιουργία ενός νέου ελέφαντα.
- `public int alive_ elephants(): Accesor-Selector`. Επιστρέφει το πλήθος των ζωντανών ελεφάντων.
- `public int dead_ elephants(): Accesor-Selector`. Επιστρέφει το πλήθος των νεκρών ελεφάντων.
- `public void draw(Graphics g) : Accesor-Selector`. Υλοποιεί την `draw(Graphics g)` της `organisms`. Ζωγραφίζει μια εικόνα ενός ελέφαντα στην θέση που αυτός βρίσκεται πάνω στον χάρτη.
- `public void die() : Transformer-Mutative`. Υλοποιεί την `die()` της `organisms`. Καλείται όταν πεθαίνει ένας ελέφαντας και μειώνει τις στατικές μεταβλητές για τους ζωντανούς οργανισμούς ενώ αυξάνει τις στατικές μεταβλητές για τους νεκρούς οργανισμούς.
- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.1.11 Η κλάση giraffe

Είναι τελική κλάση από την οποία δημιουργούνται όλες οι καμηλοπαρδάλεις. Είναι υποκλάση της κλάσης `herbivores`. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- `alive_ giraffes : (protected static int alive_ giraffes)` Πόσες καμηλοπαρδάλεις είναι ζωντανές συνολικά – στατική μεταβλητή.
- `dead_ giraffes: (protected static int dead_ giraffes)` Πόσες καμηλοπαρδάλεις είναι νεκρές συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης `giraffes` είναι:

- `public giraffe(int x, int y, ImageIcon im) : Constructor` – καλείται κατά την δημιουργία μιας νέας καμηλοπάρδαλης.
- `public int alive_ giraffes(): Accesor-Selector`. Επιστρέφει το πλήθος των ζωντανών καμηλοπαρδάλεων.
- `public int dead_ giraffes(): Accesor-Selector`. Επιστρέφει το πλήθος των νεκρών καμηλοπαρδάλεων.
- `public void draw(Graphics g) : Accesor-Selector`. Υλοποιεί την `draw(Graphics g)` της `organisms`. Ζωγραφίζει μια εικόνα μιας καμηλοπάρδαλης στην θέση που αυτή βρίσκεται πάνω στον χάρτη.
- `public void die() : Transformer-Mutative`. Υλοποιεί την `die()` της `organisms`. Καλείται όταν πεθαίνει μια καμηλοπάρδαλη και μειώνει

τις στατικές μεταβλητές για τους ζωντανούς οργανισμούς ενώ αυξάνει τις στατικές μεταβλητές για τους νεκρούς οργανισμούς.

- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.1.12 Η κλάση `bush`

Είναι τελική κλάση από την οποία δημιουργούνται όλες οι θάμνοι. Είναι υποκλάση της κλάσης `plants`. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- `alive_bushes : (protected static int alive_bushes)` Πόσοι θάμνοι είναι ζωντανοί συνολικά – στατική μεταβλητή.
- `dead_bushes: (protected static int dead_bushes)` Πόσοι θάμνοι είναι νεκροί συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης `bushes` είναι:

- `public bush(int x, int y, ImageIcon im) : Constructor` – καλείται κατά την δημιουργία ενός νέου θάμνου.
- `public int alive_bushes(): Accesor-Selector`. Επιστρέφει το πλήθος των ζωντανών θάμνων.
- `public int dead_bushes(): Accesor-Selector`. Επιστρέφει το πλήθος των νεκρών θάμνων.
- `public void draw(Graphics g) : Accesor-Selector`. Υλοποιεί την `draw(Graphics g)` της `organisms`. Ζωγραφίζει μια εικόνα ενός θάμνου στην θέση που αυτός βρίσκεται πάνω στον χάρτη.
- `public void die() : Transformer-Mutative`. Υλοποιεί την `die()` της `organisms`. Καλείται όταν πεθαίνει ένας ελέφαντας και μειώνει τις στατικές μεταβλητές για τους ζωντανούς οργανισμούς ενώ αυξάνει τις στατικές μεταβλητές για τους νεκρούς οργανισμούς.
- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.1.13 Η κλάση `tree`

Είναι τελική κλάση από την οποία δημιουργούνται όλα τα δέντρα. Είναι υποκλάση της κλάσης `plants`. Τα πεδία αυτής της κλάσης, εκτός από τα κληρονομημένα, είναι:

- `alive_trees : (protected static int alive_trees)` Πόσα δέντρα είναι ζωντανά συνολικά – στατική μεταβλητή.

- `dead_ trees: (protected static int dead_ trees)` Πόσα δέντρα είναι νεκρά συνολικά – στατική μεταβλητή.

Οι μέθοδοι της κλάσης `trees` είναι:

- `public tree(int x, int y, ImageIcon im) : Constructor` – καλείται κατά την δημιουργία ενός νέου δέντρου.
- `public int alive_ trees(): Accesor-Selector`. Επιστρέφει το πλήθος των ζωντανών δέντρων.
- `public int dead_ trees(): Accesor-Selector`. Επιστρέφει το πλήθος των νεκρών δέντρων.
- `public void draw(Graphics g) : Accesor-Selector`. Υλοποιεί την `draw(Graphics g)` της `organisms`. Ζωγραφίζει μια εικόνα ενός δέντρου στην θέση που αυτό βρίσκεται πάνω στον χάρτη.
- `public void die() : Transformer-Mutative`. Υλοποιεί την `die()` της `organisms`. Καλείται όταν πεθαίνει ένα δέντρο και μειώνει τις στατικές μεταβλητές για τους ζωντανούς οργανισμούς ενώ αυξάνει τις στατικές μεταβλητές για τους νεκρούς οργανισμούς.
- `public static void initStats() : Transformer-Mutative`. Αρχικοποιεί τα στατιστικά.

5.2 To package Position

Το package `Position` περιέχει μια μόνο κλάση, την `Position`. Η κλάση `Position` χρησιμοποιείται για να προσδιορίζουμε την θέση ενός αντικειμένου πάνω στον χάρτη. Τα πεδία αυτής της κλάσης είναι:

- `x : (private int x)` Η συντεταγμένη `x` της θέσης.
- `y : (private int y)` Η συντεταγμένη `y` της θέσης.
- `max_distance : (private final int max_distance)` Η μέγιστη απόσταση από την οποία ένα ζώο μπορεί να φάει έναν άλλον οργανισμό.

Τα πεδία της κλάσης `Position` είναι `private` μεταβλητές επειδή πρέπει να είναι κρυφά από το έξω κόσμο. Οι μέθοδοι της κλάσης `Position` είναι:

- `public Position(int x, int y) : Constructor`. Δημιουργεί μια νέα θέση με συντεταγμένες `x,y`.
- `public int get_x() : Accesor-Selector`. Επιστρέφει την συντεταγμένη `x`.
- `public int get_y() : Accesor-Selector`. Επιστρέφει την συντεταγμένη `y`.
- `public void set_x(int x) : Transformer-Mutative`. Θέτουμε το `x`.
- `public void set_y(int y) : Transformer-Mutative`. Θέτουμε το `y`.

- `public boolean checkIfClose(Position p) : Accesor-Observer`. Ελέγχει αν οι δυο θέσεις των οργανισμών είναι τόσο κοντά ώστε να μπορεί ο ένας να φάει τον άλλο.
- `public boolean isEqual(Position p) : Accesor-Observer`. Ελέγχει αν δυο θέσεις είναι ίδιες.
- `public double distance(Position p) : Accesor-Selector`. Επιστρέφει την απόσταση δύο θέσεων.
- `public boolean isInside(Position p, int width, int height): Accesor-Observer`. Ελέγχει αν ένα αντικείμενο είναι μέσα σε ένα άλλο.

5.3 Οι κλάσεις με τις λειτουργίες

Για τις λειτουργίες του EcoSim φτιάξαμε δυο κλάσεις, την Operations και την Task.

5.3.1 Η κλάση Task

Είναι η κλάση η οποία αφορά τη δημιουργία μιας ανάθεσης εργασίας, η οποία αργότερα θα μπαίνει σαν όρισμα σε ένα instance μιας κλάσης Timer. Έτσι θα καθορίζεται τι θα κάνει ένα Object αυτής της κλάσης (Timer) σε τακτά χρονικά διαστήματα. Η κλάση αυτή κληρονομεί την abstract κλάση TimerTask έτσι ώστε να κάνει override την abstract μέθοδο `run()` και υλοποιεί το interface `Runnable` εν μέσω της κλάσης TimerTask που κληρονομεί. Ουσιαστικά θα καθορίζει την κίνηση και την αλληλεπίδραση ενός οργανισμού με τους υπόλοιπους οργανισμούς του χάρτη προσομοίωσης σε καθημερινή βάση.

Τα πεδία, εκτός των πεδίων που αυτή η κλάση κληρονομεί (εννοείται όχι `private` ή `static`), είναι :

- `Vector orgs (private)`: Η δομή δεδομένων μέσα στην οποία κρατούνται όλοι οι οργανισμοί που υπάρχουν στο χάρτη της προσομοίωσης.
- `Map map (private)`: instance της κλάσης Map που αφορά το χάρτη της προσομοίωσης μας.
- `static int week (private)`: Αυτή η μεταβλητή αφορά τον αριθμό της εβδομάδας της προσομοίωσης μας.
- `static int day (private)`: Αυτή η μεταβλητή αφορά τον αριθμό της ημέρας μέσα στη εβδομάδα της προσομοίωσης μας.
- `Operations op (private)`: Ένα instance της κλάσης Operations.

Οι μέθοδοι αυτής της κλάσης είναι:

- `public Task(Operations op)`: Εδώ ο constructor αρχικοποιεί το πεδίο `op` να είναι ίσο με το όρισμα.
- `public void loadVector(Vector v)`: Αυτή η μέθοδος φορτώνει στο πεδίο `orgs` το `v`.
- `public void loadMap(Map map)` : Αυτή η μέθοδος φορτώνει στο πεδίο `map` το όρισμα `map`.
- `public void setDay(int day)` : Αυτή η μέθοδος κάνει το πεδίο `day` ίσο με το όρισμα `day`.
- `public void setWeek(int week)`: Αυτή η μέθοδος κάνει το πεδίο `week` ίσο με το όρισμα `week`.
- `public int getDay()` : Αυτή η μέθοδος επιστρέφει το πεδίο `day`.
- `public int getWeek()`: Αυτή η μέθοδος επιστρέφει το πεδίο `week`.
- `Public void run ()`: Κάνει override την abstract μέθοδο της super class. Ουσιαστικά αλλάζει τα περιεχόμενα του `orgs` αφού ελέγχει επί καθημερινής βάσης(όσον αφορά την προσομοίωση) τις αλληλεπιδράσεις μεταξύ των οργανισμών και ανάλογα ρυθμίζει τις συντεταγμένες τις θέσης τους καθώς και τα στατιστικά τους . Για κάθε κλήση της αυξάνει το πεδίο `day` και κάθε 8 κλήσεις της αυξάνει το πεδίο `week`. Πιο συγκεκριμένα την πρώτη μέρα κάθε εβδομάδας διασχίζουμε τον `orgs` και καλούμε την μέθοδο `increase_age()` και `restore Energy()` για κάθε οργανισμό και αν είναι και instances της κλάσης `animals` καλούμε την `reset_EatenThisWeek()`. Τώρα όσον αφορά κάθε μέρα, χρησιμοποιούμε την static μέθοδο `Math.random()` για να βρούμε τυχαία τις συντεταγμένες της θέσης κάθε αντικειμένου της κλάσης `animals` στο χάρτη(φροντίζοντας να είναι εντός των ορίων του χάρτη και με διάφορα μέχρι 26 pixels σε σχέση με την προηγούμενη θέση του). Επίσης καλούμε την μέθοδο `findCloserToEat(animals o)` ώστε κάθε ζώο να τρώει τον κοντινότερο για αυτόν οργανισμό που πληροί τις απαραίτητες προϋποθέσεις. Επιπλέον καλώντας τη μέθοδο `isAlive()` από την κλάση `organisms` ελέγχουμε, διατρέχοντας πάλι τον `orgs`, αν κάποιος οργανισμός πέθανε και αν ναι καλούμε την μέθοδο `die()` της κλάσης `organisms` και τη `remove(index i)` της κλάσης `Vector` για αυτό τον οργανισμό. Τέλος καλούμε τη μέθοδο `repaint()` της κλάσης `Map` , για να ανανεωθούν οι θέσεις των

οργανισμών πάνω στο χάρτη και τη reloadLabels() για το πεδίο op ώστε να ανανεωθούν και οι στατιστικές πληροφορίες.

- private void findCloserToEat(animals o) : Εντοπίζει πιο είναι το κοντινότερο animal που μπορεί να φάει ο οργανισμός o.

5.3.2 Η κλάση Operations

Η κλάση αυτή υλοποιεί όλες τις λειτουργίες που χρησιμοποιούνται στην προσομοίωση του οικοσυστήματός μας. Τέτοιες λειτουργίες είναι η δημιουργία ενός νέου οργανισμού, η έναρξη, η παύση, η συνεχεία και η επανεκκίνηση της προσομοίωσης. Επίσης προσφέρεται η δυνατότητα προεπιλογής ενός έτοιμου, τυχαίου κάθε φορά οικοσυστήματος. Επιπλέον η κλάση υλοποιεί την απεικόνιση στατιστικών πληροφοριών που αφορούν τόσο μεμονωμένους οργανισμούς όσο και γενικότερα είδη οργανισμών. Τα πεδία αυτής της κλάσης είναι:

- Vector orgs (private): Σε αυτή τη δομή δεδομένων αποθηκεύεται το σύνολο των οργανισμών που είναι ζωντανοί στο χάρτη του οικοσυστήματος.
- java.util.Timer t(private): Ένα instance της κλάσης Timer από το package java.util όπου θα καλούνται σε μεθόδους της, instances της Task.
- Task task (private): Ένα instance της κλάσης Task, το οποίο μπαίνει σαν όρισμα σε μεθόδους της κλάσης java.util.Timer
- int delay (private): Η χρονική καθυστέρηση που θέλουμε να έχει η προσομοίωση κατά την πρώτη εκκίνηση. Δηλαδή ο χρόνος σε milliseconds που θέλουμε να καθυστερήσει μια λειτουργία(εκκίνηση, επανεκκίνηση, εκκίνηση προσομοιώσεις προεπιλεγμένου οικοσυστήματος) από τη στιγμή που θα την επιλέξουμε μέχρι να μπει σε εφαρμογή.
- int period (private): Η χρονική καθυστέρηση που θέλουμε να έχει η προσομοίωση μεταξύ κλήσεων του ίδιου task. Δηλαδή ο χρόνος σε milliseconds που θέλουμε να αντιστοιχεί σε μια νοητή ημέρα, οπότε και θα κινούνται οι οργανισμοί του οικοσυστήματος και θα αλληλεπιδρούν μεταξύ τους.
- boolean start (private): Αυτή η μεταβλητή «βλέπει» αν έχει γίνει εκκίνηση της προσομοίωσης (π.χ. για να μη γίνεται πάγωμα αν προηγουμένως δεν έχει γίνει εκκίνηση).

- `boolean resume (private)` : Αυτή η μεταβλητή «βλέπει» αν υπάρχει δυνατότητα συνέχισης της προσομοίωσης (π.χ για να μη γίνεται `resume` αν προηγουμένως δεν έχει γίνει `pause`).
- `Map c (private)`: ένα instance της κλάσης `Map` που αφορά το χάρτη της εφαρμογής μας.
- `JLabel l1[],l2[],l3[] (private)`: Πίνακες από instances της κλάσης `JLabel` από το package `javax.swing`. που αφορούν την απεικόνιση των στατιστικών πληροφοριών μέσα στο παράθυρο της εφαρμογής (l1:κάτω αριστερά,l2:κάτω μέση,l3:κάτω δεξιά.)
- `ImageIcon lions, tigers, bears, zebras, elephants, giraffes, trees, bushes (private)`: Instances της κλάσης `ImageIcon` από το package `javax.swing`. που αφορούν τις εικόνες που αναπαριστούν τη μορφή οργανισμών από instances αντίστοιχης κλάσης με το όνομα του πεδίου.

Οι μέθοδοι της κλάσης είναι:

- `Operations()`: Είναι ο constructor της κλάσης. Αρχικοποιεί τον `t`, κάνει το `delay` 0 και το `period` 1000(1 δευτερόλεπτο). Επιπλέον κάνει το `day` και το `week` 1. Το `start` παίρνει την τιμή `true`(ώστε να μπορεί να γίνει εκκίνηση και το `resume` `false`(ώστε να μη μπορεί να γίνει συνέχιση αφού δεν έχει γίνει ακόμα εκκίνηση).
- `public void loadMap (Map c)`: Κάνει το πεδίο `map` ίσο με `c`.
- `public void loadLabels (JLabel[] l1, JLabel[] l2, JLabel[] l3)`: Κάνει τα πεδία `l1[], l2[], l3[]` ίσα με τα αντίστοιχα ορίσματα.
- `public void setTask (Task task)`: Κάνει το πεδίο `task` ίσο με το όρισμα.
- `public boolean create (int x, int y, String className, ImageIcon im)`: Δημιουργεί ένα νέο οργανισμό που έχει συντεταγμένες θέσεις πάνω στο χάρτη της προσομοίωσης τις (x,y). Αν στις συντεταγμένες αυτές υπάρχει ήδη κάποιος οργανισμός (διατρέχει το `orgs` και συγκρίνει το `position()`) ή είναι εκτός των ορίων του χάρτη επιστρέφει `false` και δε δημιουργείται ο οργανισμός. Σε αντίθετη περίπτωση θα δημιουργηθεί οργανισμός που ανήκει στην κλάση με όνομα `className` (π.χ. `lion`, `bush` κτλ) και θα επιστραφεί η τιμή `true`.
- `public void start ()`: Ξεκινάει την προσομοίωση και καθορίζει τις ενέργειες πάνω στους οργανισμούς που πρέπει να γίνονται τόσο σε καθημερινή όσο και σε εβδομαδιαία βάση. Αρχικά δημιουργούμε ένα νέο instance `Task` και το φορτώνουμε μέσω

της μεθόδου `setTask(Task task)`. Στη συνέχεια φορτώνουμε τον `orgs` (για να είναι ορατός και στα αντικείμενα τύπου `Task`) μέσω της μεθόδου `loadVector(Vector v)` και καλούμε τη μέθοδο `schedule(TimerTask t, int d, int p)` της κλάσης `Timer`. Μέσα στο σώμα της `start()` μπαίνουμε μόνο αν `start=true`, για αυτό μέσα στο κύριο σώμα κάνουμε `start=false` (ώστε να μπορούμε να κάνουμε εκκίνηση μόνο μια φορά) κι επιπλέον `resume=false` (ώστε να μη μπορεί να γίνει `resume`, χρειάζεται πρώτα να έχει κληθεί η μέθοδος `pause()`).

- `public void preconstruct()`: Ξεκινάει την προσομοίωση με 20 τυχαίους οργανισμούς σε τυχαίες κάθε φορά θέσεις. Η μέθοδος αυτή μπορεί να κληθεί οποιαδήποτε στιγμή. Εδώ μέσω της static μεθόδου `Math.random()` γίνονται τυχαία οι επιλογές των οργανισμών και των θέσεων και καλείται η μέθοδος `create()`. Αν είναι η πρώτη φορά που καλείται η μέθοδος καλούμε την `start()`, διαφορετικά `restart()` για να ξεκινήσει η προσομοίωση με τους τυχαίους οργανισμούς.
- `public void pause ()`: Παγώνει την προσομοίωση και αφήνει τα χαρακτηριστικά των οργανισμών όπως ήταν και πριν το κάλεσμα της μεθόδου. Στο σώμα αυτής της μεθόδου μπαίνουμε μόνο αν `start=false` (άρα έχει ήδη κληθεί η `start()`). Εδώ καλείται η μέθοδος `cancel()` από την κλάση `TimerTask` για το τρέχον `Task` όποτε και σταματάει ο `Timer t`. Επιπλέον κάνουμε `resume=true` (ώστε να μπορούμε πλέον να μπούμε στο σώμα της `resume()`).
- `public void resume ()`: Επαναφέρει την προσομοίωση στο σημείο που ήταν πριν το κάλεσμα της `pause ()` και συνεχίζει από εκεί. Στο κύριο σώμα της μεθόδου μπαίνουμε μόνο αν `resume=true` (άρα έχει πριν κληθεί η `pause()`). Εδώ φτάνουμε ένα νέο instance της κλάσης `Task` και το φορτώνουμε μέσω της μεθόδου `setTask(Task t)`. Στη συνέχεια καλούμε τη `start()` αφού πρώτα έχουμε κάνει `start=true` (για να μπορεί να μπει μέσα στο σώμα της `start()`). Επιπλέον κάνουμε `resume=false` για να μη μπορεί να ξαναμπει στη `resume()` αν πρώτα δεν έχει κληθεί η `pause()`.
- `public void restart ()`: Σβήνει από τον `orgs` όλους τους οργανισμούς και από εκεί και πέρα ξεκινά μια νέα προσομοίωση όπως ορίζει και η `start ()`. Εδώ μπαίνουμε μόνο αν `start=false` δηλ. έχει ήδη κληθεί η `start()`. Επιπλέον εδώ καλούμε πάλι τη μέθοδο `cancel()` της κλάσης `Task` και τη

μέθοδο `die()` της κλάσης `organism` για κάθε οργανισμό που βρίσκεται στο `orgs` (πριν τον σβήσουμε). Μέσω τον μεθόδων `setWeek()`, `setDay()` της κλάσης `Task` αρχικοποιούμε το `day` και `week`. Αφού φορτώσουμε ένα νέο `Task` (με τον ίδιο τρόπο όπως και πριν) κάνουμε `repaint()` (από την κλάση `Map`) και καλούμε τη μέθοδο `start()` (αφού κάνουμε `start=true` και `resume=false`).

- `public void population()` : Προβάλλει μέσα στο παράθυρο της εφαρμογής (κάτω δεξιά) πληροφορίες που αφορούν τον αριθμό όλων των διαφορετικών ειδών οργανισμών κάθε στιγμή. Πιο συγκεκριμένα χρησιμοποιούμε τη μέθοδο `setText(String s)` της κλάσης `JLabel` για να «γράψουμε» πάνω στα `JLabel` του πίνακα `l3`. Επίσης καλούμε τις μεθόδους `alive_lions`, `alive_tigers` κτλ. για να βρούμε τους ζωντανούς οργανισμούς κάθε είδους.
- `public void populationInfo (String className)`: Προβάλλει στην οθόνη (κάτω αριστερά) πληροφορίες που αφορούν στατιστικά του είδους των οργανισμών που καθορίζει η `className`. Εδώ μέσω `if`, `else if` και της μεθόδου `equals(String s)` της κλάσης `String` επιλέγουμε την κλάση που αντιστοιχεί στο `className`. Από εκεί και πέρα κάνουμε κλήσεις των `static` μεθόδων που αντιστοιχούν στην εκάστοτε κλάση (π.χ `bear` `alive_bears()`, `dead_bears()`) καθώς και μεθόδων που αφορούν την κλάση `organism` (όπως `dead_organisms()`) και από την κλάση `animals` (`eatenThisWeek()`). Έτσι παίρνουμε τις στατιστικές πληροφορίες για κάθε ομάδα οργανισμών και μέσω της μεθόδου `setText()` της κλάσης `JLabel` «γράφουμε» στα `JLabel` του πίνακα `l3`.
- `public void OrganismInfo (int x, int y)`: Προβάλλει στην οθόνη πληροφορίες που αφορούν στατιστικά του μεμονωμένου οργανισμού που έχει θέση στο χάρτη, που καθορίζεται από τις συντεταγμένες `(x,y)`. Εδώ διατρέχουμε όλο τον `orgs` μέχρι να βρούμε ένα οργανισμό που έχει συντεταγμένες θέσης `(x,y)`. Στη συνέχεια μέσω της μεθόδου `setText()` της κλάσης `JLabel` «γράφουμε» τις πληροφορίες στα `JLabel` του πίνακα `l2`.
- `public void drawAll (Graphics g)`: Διασχίζει το `orgs` και καλεί για κάθε οργανισμό τη μέθοδο `draw(g)` από την κλάση `organism`.
- `public void clearLabels(JLabel[] l)`: Σβήνουμε τα περιεχόμενα του `l`. Αυτό γίνεται πάλι καλώντας τη μέθοδο `setText(String s)` όπου `s` ένα κενό `String`.

- `public void reloadLabels()` : Ανανεώνει τα στατιστικά. Καλεί κάθε φορά όποια από τις μεθόδους `population()` και `populationInfo()` είχε καλεστεί τελευταία για το κάτω αριστερά πεδίο των στατιστικών-πληροφοριών, την `OrganismInfo()` με τον επιλεγμένο οργανισμό για το μεσαίο κάτω πεδίο, και τις `task.getDay()` και `task.getWeek()` για το κάτω αριστερό πεδίο έτσι ώστε να ανανεώνονται τα στατιστικά. Καλείται κάθε μέρα κατά την διάρκεια της προσομοίωσης.
- `private void initializeStatistics()` : Μηδενίζει τα στατιστικά όλων των οργανισμών. Αυτό γίνεται καλώντας τη μέθοδο `initStats()` που υπάρχει υλοποιημένη σε όλες τις κλάσεις του `package organisms`.

5.4 Οι κλάσεις του γραφικού περιβάλλοντος

Για το γραφικό περιβάλλον υλοποιήσαμε δύο κλάσεις, την `EcoSim` και την `Map`.

5.3.1 Η κλάση `EcoSim`

Η κλάση `EcoSim` είναι η βασική κλάση της εφαρμογής. Περιέχει την `main` που καλείται όταν το `EcoSim` εκτελείται σαν εφαρμογή και την `init` που καλείται όταν το `EcoSim` εκτελείται σαν `Applet`. Διαβάζει από τα αρχεία όλες τις εικόνες, δημιουργεί όλο το γραφικό περιβάλλον της εφαρμογής και χειρίζεται τα `events` που προκύπτουν από τον χρήστη. Κληρονομεί την `JApplet` για να έχει την δυνατότητα το `EcoSim` να τρέχει και σαν `Applet`, και κάνει `override` την μέθοδο `init` όπως είπαμε, ενώ υλοποιεί το `interface ActionListener` έτσι ώστε να κάνει `override` την μέθοδο `actionPerformed(ActionEvent e)` και να χειρίζεται μέσω αυτής της μεθόδου τα `events`. Αναλυτικά, τα πεδία αυτής της κλάσης είναι:

- `private Operations op` : Ένα instance της κλάσης `Operations` από το οποίο θα εκτελούνται όλες οι λειτουργίες που χρειάζονται.
- `private JFrame frame` : Το βασικό παράθυρο της εφαρμογής.
- `private JMenuBar menu` : Το `Menu Bar` του παραπάνω παραθύρου.
- `private JPanel Icons` : Το `JPanel` με τα εικονίδια των οργανισμών που μπορούν να εισαχθούν στον χάρτη.

- private JPanel Statistics : Το JPanel που περιέχει τρία άλλα JPanel με τα 3 πεδία πληροφοριών-στατιστικών.
- private JPanel Statistics1 : Το πρώτο JPanel με τις πληροφορίες για τον χρόνο της προσομοίωσης (κάτω αριστερά)
- private JPanel Statistics2 : Το δεύτερο JPanel με τα στατιστικά για τον επιλεγμένο οργανισμό (κάτω στο κέντρο)
- private JPanel Statistics3 : Το τρίτο JPanel με τα στατιστικά για κατηγορίες οργανισμών και πληθυσμό (κάτω δεξιά)
- private Map map : Ένα instance της κλάσης Map που χρησιμοποιείται για τον χάρτη του οικοσυστήματος.
- private JLabel l1[] : Τα JLabel με τις πληροφορίες για το πρώτο JPanel (Statistics1)
- private JLabel l2[] : Τα JLabel με τις πληροφορίες για το δεύτερο JPanel (Statistics2)
- private JLabel l3[] : Τα JLabel με τις πληροφορίες για το τρίτο JPanel (Statistics3)
- private JFrame about : Ένα επιπλέον μικρό JFrame που εμφανίζεται όταν επιλέξουμε την επιλογή 'Project Team' από το μενού 'About' του Menu Bar και γράφει τα ονόματα των μελών της ομάδας που έφτιαξε τον EcoSim.
- ImageIcon lion, lion_big, tiger, tiger_big, bear, bear_big, zebra, zebra_big, giraffe, giraffe_big, elephant, elephant_big, tree, tree_big, bush, bush_big : Τα εικονίδια που χρησιμοποιούμε στον EcoSim.

Οι μέθοδοι της EcoSim είναι οι εξής:

- public void Initialize() : Transformer-Mutative. Καλείται από την main και από την init και κάνει τις κατάλληλες αρχικοποιήσεις ενώ φτιάχνει και όλο το γραφικό περιβάλλον όπως είναι στην αρχή.
- public void clearLabels(JLabel[] l) : Transformer-Mutative. Σβήνει το περιεχόμενο των JLabel του πίνακα l.
- public void init() : Καλείται όταν τρέχουμε τον EcoSim σαν Applet. Διαβάζει τις εικόνες σαν URL και τις γράφει στα ImageIcon που υπάρχουν εσωτερικά, και έπειτα καλεί την Initialize().
- public void paint(Graphics g) : Καλείται όταν τρέχουμε τον EcoSim σαν Applet και ζωγραφίζει στην σελίδα απλώς ένα String.
- public static void main(String s[]) : Καλείται όταν τρέχουμε τον EcoSim σαν Application. Διαβάζει τις εικόνες από τα αρχεία και τις γράφει στα ImageIcon που υπάρχουν εσωτερικά, και έπειτα καλεί την Initialize().

- `public void actionPerformed(ActionEvent e) : Transformer-Mutative.` Για κάθε `ActionEvent e` που μπορεί να γίνει στον `EcoSim`, όπως το πάτημα ενός κουμπιού ή ενός `Menu Item`, καλεί την κατάλληλη μέθοδο για να κάνει την κατάλληλη λειτουργία.

5.3.1 Η κλάση `Map`

Η κλάση `Map` είναι η κλάση που υλοποιεί τον χάρτη του οικοσυστήματος. Πάνω στον χάρτη του οικοσυστήματος εισάγονται, κινούνται και εξελίσσονται οι οργανισμοί όπως έχουμε πει. Ο χάρτης έχει πράσινο `background`, πάνω σε αυτόν ζωγραφίζονται τα εικονίδια των οργανισμών στις κατάλληλες θέσεις και “πιάνει” τα `MouseEvent`s (πάτημα του ποντικιού σε κάποιο σημείο). Έχει καθορισμένες διαστάσεις (650x450) και όλα τα ζώα πρέπει να κινούνται σε αυτές τις διαστάσεις. Πρέπει να διευκρινίσουμε ότι θέση ενός οργανισμού θεωρούμε την πάνω αριστερή γωνία του εικονιδίου του. Η κλάση `Map` κληρονομεί την κλάση `JComponent` και κάνει `override` την μέθοδο `paint`, ενώ υλοποιεί και το `interface ActionListener` για να “πιάνει” τα `events` του πατήματος του ποντικιού. Έτσι υλοποιεί όλες τις μεθόδους του `interface ActionListener`. Τα πεδία αυτής της κλάσης είναι:

- `Operations op` : Το `instance` της κλάσης `Operations` από το οποίο εκτελούμε τις λειτουργίες.
- `boolean creation` : Εάν περιμένουμε να γίνει εισαγωγή οργανισμού ή όχι.
- `ImageIcon im` : Το εικονίδιο του οργανισμού που περιμένουμε να εισαχθεί.
- `String className` : Το όνομα της κλάσης του οργανισμού που περιμένουμε να εισαχθεί.

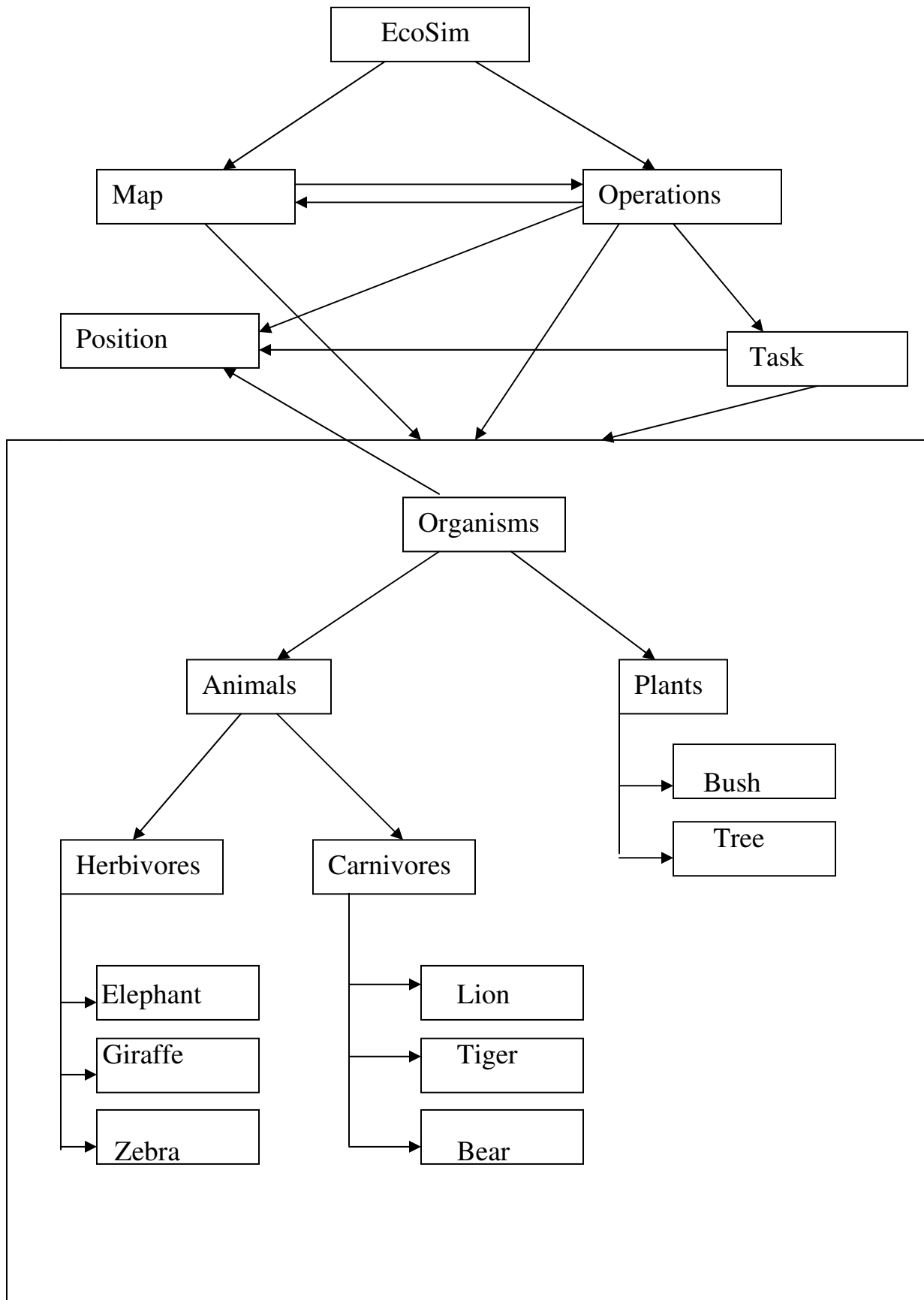
Οι μέθοδοι της κλάσης `Map` είναι:

- `public Map(Operations op) : Constructor.` Δημιουργεί ένα καινούργιο στιγμιότυπο της κλάσης `Map` και φορτώνει το `instance` της κλάσης `Operations op` στην εσωτερική μεταβλητή `op`.
- `public void creation(String n, ImageIcon im) : Transformer-Mutative.` Θέτει την `creation=true` και τα `className=n` και `this.im=im`. Καλείται από την `actionPerformed` της `EcoSim` όταν αρχίζει η εισαγωγή ενός οργανισμού πατώντας στο κουμπί με το εικονίδιο του οργανισμού από αριστερά και πριν επιλεγεί η θέση του οργανισμού στον χάρτη.
- `public void paint(Graphics g) : Ζωγραφίζει όλους τους οργανισμούς πάνω στον χάρτη καλώντας την μέθοδο drawAll από το instance op της Operations.`

- `public void mouseClicked(MouseEvent e) : Transformer-Mutative.` Υλοποιεί την αντίστοιχη μέθοδο από το `MouseListener` interface. Όταν περιμένουμε εισαγωγή καλεί την `create` για να δημιουργήσει νέο οργανισμό ενώ όταν δεν κάνουμε εισαγωγή ψάχνει να βρει αν επιλέξαμε κάποιον οργανισμό και ποιόν για να δείχνουμε τις πληροφορίες του.
- `public void mousePressed(MouseEvent e) :` Υλοποιεί την αντίστοιχη μέθοδο από το `MouseListener` interface.
- `public void mouseReleased(MouseEvent e) :` Υλοποιεί την αντίστοιχη μέθοδο από το `MouseListener` interface.
- `public void mouseEntered(MouseEvent e) :` Υλοποιεί την αντίστοιχη μέθοδο από το `MouseListener` interface.
- `public void mouseExited(MouseEvent e) :` Υλοποιεί την αντίστοιχη μέθοδο από το `MouseListener` interface.

6. Σχέσεις και επικοινωνία των κλάσεων

Οι παραπάνω κλάσεις επικοινωνούν ως εξής: Πρώτα, η κεντρική κλάση `EcoSim` φτιάχνει το γραφικό περιβάλλον της εφαρμογής, δηλαδή το `MenuBar`, τον χάρτη, τα κουμπιά με τα εικονίδια των ζώων για εισαγωγή και τα πεδία που φαίνονται οι πληροφορίες. Για να φτιάξει τον χάρτη η `EcoSim` χρησιμοποιεί ένα στιγμιότυπο της κλάσης `Map`. Έπειτα, ανάλογα με το event που “πίάνει” κάθε φορά ο `ActionListener` της κλάσης `EcoSim` εκτελεί την κατάλληλη λειτουργία καλώντας την κατάλληλη μέθοδο της κλάσης `Operations`. Η κλάση `Operations` χρησιμοποιεί την κλάση `Task` για τις λειτουργίες της προσομοίωσης. Η κλάση `Operations` και η κλάση `Task` χρησιμοποιούν την δομή δεδομένων `Vector` για να αποθηκεύουν τους ζωντανούς οργανισμούς που υπάρχουν στο οικοσύστημα. Για να υλοποιήσει τις λειτουργίες της η κλάση `Operation` χρησιμοποιεί τις μεθόδους των κλάσεων που ορίζονται στο `package Organisms`. Το ίδιο και η κλάση `Task`. Σχεδόν όλες οι κλάσεις χρησιμοποιούν την κλάση `Position` από το `package Position` για την θέση των αντικειμένων πάνω στον χάρτη. Οι κλάσεις στο `package Organisms` συνδέονται με σχέσεις κληρονομικότητας όπως ακριβώς περιγράφηκε στις παραγράφους 3 και 5.1. Σχηματικά, οι σχέσεις και η επικοινωνία των κλάσεων που υλοποιήθηκαν φαίνεται στο παρακάτω σχήμα.



7. Αλλαγές σε σχέση με την σχεδίαση της 1^{ης} φάσης

Στην 2^η φάση -υλοποίηση- του project δεν έγιναν πολλές αλλαγές σε σχέση με την σχεδίαση που είχε γίνει στην 1^η φάση. Οι αλλαγές που έγιναν ήταν ελάχιστες και όχι ουσιαστικές αλλαγές. Μερικές από αυτές είναι ότι χρησιμοποιήσαμε μεθόδους για την αρχικοποίηση των στατιστικών στις κλάσεις του package organisms (initStats()) καθώς διαπιστώθηκε ότι χρειαζόταν, προστέθηκε μια νέα μέθοδος στην κλάση Position του package Position που μας δίνει την απόσταση δύο θέσεων σε double (distance(Position p)) και μερικές αλλαγές έγιναν στο γραφικό περιβάλλον για να γίνει πιο όμορφο και πιο εύχρηστο. Συγκεκριμένα υλοποιήσαμε τον χάρτη του οικοσυστήματος σε ξεχωριστή κλάση, φτιάξαμε στο κάτω μέρος του βασικού παραθύρου τρία μέρη για να φαίνονται τα στατιστικά και προσθέσαμε δυο λειτουργίες στο Menu Bar, την Exit (βγαίνει από την εφαρμογή) και την About (βγάζει σε ξεχωριστό μικρό παράθυρο τα ονόματα των μελών της ομάδας του project).

8. Προβλήματα που αντιμετωπίστηκαν

Κατά την διάρκεια της υλοποίησης του EcoSim αντιμετωπίστηκαν λίγα προβλήματα. Ένα από αυτά ήταν η υλοποίηση του χάρτη και συγκεκριμένα η εμφάνιση του πράσινου background μαζί με τους οργανισμούς. Αρχικά για την υλοποίηση του χάρτη είχε χρησιμοποιηθεί η κλάση Canvas από το java.awt package. Όμως ο Canvas είχε διάφορα προβλήματα καθώς επικάλυπτε το Menu Bar. Έτσι χρησιμοποιήθηκε η λύση του JComponent όπου κάναμε override την μέθοδο paint και για να εμφανίζουμε το πράσινο background ζωγραφίζουμε ένα τετράγωνο με συντεταγμένες αυτές του χάρτη (650x450). Έτσι το πρόβλημα αυτό λύθηκε χωρίς επιπτώσεις για τον χρήστη του EcoSim. Ένα άλλο πρόβλημα ήταν η συμπεριφορά της εφαρμογής, και κυρίως του χάρτη δηλαδή, σε περίπτωση αλλαγής μεγέθους του κεντρικού παραθύρου (resize). Όμως ο χάρτης έχει σταθερά όρια και άρα μια ενδεχόμενη αλλαγή μεγέθους του παραθύρου (resize) θα αφήσει ανεπηρέαστο τον χάρτη και την εφαρμογή. Ένα ακόμα πρόβλημα ήταν οι επικαλύψεις των οργανισμών στην κίνησή τους πάνω στον χάρτη (αν θα μπορεί να πέφτει το ένα πάνω στο άλλο). Η λύση που δόθηκε σε αυτήν την περίπτωση είναι να επιτρέπονται οι επικαλύψεις των οργανισμών, καθώς αυτό συμβαίνει και στον πραγματικό κόσμο όπου 2 ή περισσότεροι οργανισμοί έρχονται αρκετά κοντά, με μόνο περιορισμό το ότι δεν μπορούν

2 οργανισμοί να βρίσκονται ακριβώς στην ίδια θέση. Ένα άλλο πρόβλημα που λύθηκε εύκολα ήταν το πόσους και ποιους οργανισμούς από αυτούς που είναι κοντά σε ένα ζώο μπορεί να τρώει. Η λύση δόθηκε με μια συνάρτηση που βρίσκει τον πιο κοντινό οργανισμό που μπορεί να φάει ένα ζώο και έτσι το ζώο τρώει μόνο αυτόν, ένα οργανισμό το πολύ ανά ημέρα. Τέλος, ένα άλλο πρόβλημα ήταν ότι οι οργανισμοί δεν μπορούν να βγαίνουν εκτός ορίων. Έτσι ελέγχουμε τόσο την κίνηση των οργανισμών όσο και την εισαγωγή των οργανισμών να μη γίνεται κοντά στα όρια του χάρτη. Ελέγχουμε τις συντεταγμένες ενός οργανισμού έτσι ώστε το κέντρο της εικόνας να μην βγαίνει εκτός του χάρτη. Οι εικόνες των οργανισμών είναι 50x30 pixels εκτός της καμηλοπάρδαλης που είναι 30x50. Έτσι, τα επιτρεπτά όρια πάνω στον χάρτη έγιναν 620x420.

9. Πιθανές μελλοντικές επεκτάσεις

Το EcoSim είναι ένα πρόγραμμα που μπορεί να επεκταθεί πολύ εύκολα. Μπορούν να προστεθούν όσοι οργανισμοί ακόμα θέλουμε για να μπορούν και αυτοί να εισάγονται στο οικοσύστημα. Αρκεί να φτιάξουμε για αυτούς μια κλάση όπως αυτές που υπάρχουν για τους ήδη υπάρχοντες οργανισμούς, να βρούμε 2 εικονίδια για αυτούς, να προσθέσουμε στο γραφικό περιβάλλον κουμπιά και Menu Items όπως έχουμε κάνει και για τους άλλους και τέλος να αλλάξουμε ελάχιστα τον κώδικα στις περιπτώσεις που παίρνουμε. Επίσης, μπορούμε να προσθέσουμε στο EcoSim όσες και ότι λειτουργίες ακόμα θέλουμε εκτός από αυτές που υπάρχουν ήδη. Αρκεί να προσθέσουμε τα κατάλληλα Menu Items στο Menu Bar, να “πιάνουμε” τα αντίστοιχα events και να προσθέσουμε τις συναρτήσεις που υλοποιούν αυτές τις λειτουργίες στην κλάση Operations, όπως κάνουμε και με τις υπάρχουσες λειτουργίες. Τέλος, το περιβάλλον της εφαρμογής θα μπορούσε να γίνει πιο interactive με την χρήση ήχων ή εικόνων με animation κτλ. Και αυτό είναι κάτι το οποίο μπορεί να γίνει χωρίς κόστος, αφού όλα αυτά που έχουμε υλοποιήσει παραμένουν χρήσιμα, και το μόνο το οποίο θα άλλαζε θα ήταν κομμάτια του γραφικού περιβάλλοντος ή οι φωτογραφίες κτλ. Αξίζει επίσης να σημειώσουμε ότι ορισμένες παράμετροι του EcoSim μπορούν να αλλάξουν αμέσως και χωρίς καμία περαιτέρω αλλαγή. Τέτοιες παράμετροι είναι ο ρυθμός ανάπτυξης, η αρχική ενέργεια και το φαγητό ανά εβδομάδα (για ζώα) για τους οργανισμούς, η μέγιστη απόσταση από την οποία ένα ζώο μπορεί να φάει έναν οργανισμό, τα όρια του χάρτη, το μήκος και ύψος των εικονιδίων των οργανισμών και ο χρόνος τον οποίο κρατάει μια ‘νοητή’ μέρα προσομοίωσης.

10. Χρήση του EcoSim

Η χρήση του EcoSim είναι πολύ απλή. Μπορείτε να εισάγετε έναν από τους 8 διαθέσιμους οργανισμούς στον χάρτη πατώντας πρώτα το αντίστοιχο κουμπί από αριστερά και έπειτα πάνω στον χάρτη στο σημείο που θέλετε να εισαχθεί ο οργανισμός. Μπορείτε να κάνετε εισαγωγές πριν και κατά την διάρκεια της προσομοίωσης. Επίσης μπορείτε να επιλέξετε μια από τις διαθέσιμες λειτουργίες από το Menu Bar που βρίσκεται στην κορυφή του παραθύρου. Επιλέξτε 'Start Simulation' για να αρχίσει η προσομοίωση με τα ζώα που εσείς έχετε εισάγει στον χάρτη είτε 'Start Preconstructed Simulation' για να αρχίσει η προσομοίωση με 20 τυχαίους οργανισμούς σε τυχαίες θέσεις. Επίσης μπορείτε να σταματήσετε (Pause), να συνεχίσετε (Resume) και να ξεκινήσετε από την αρχή (Restart) την προσομοίωση. Τέλος, έχετε στην διάθεσή σας αρκετές λειτουργίες για στατιστικά που θα φαίνονται κάτω δεξιά. Επιλέγοντας κάποιον οργανισμό από τον χάρτη θα βλέπετε κάθε στιγμή, κάτω στο κέντρο, τα στοιχεία αυτού του οργανισμού όσο αυτός είναι ζωντανός και μέχρι να επιλέξετε κάποιον άλλον. Για να βγείτε από τον EcoSim απλά κλείστε το παράθυρο είτε πατήστε Exit από το Menu Bar.

Καλή Διασκέδαση.....