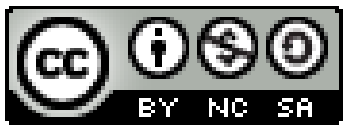
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Αντικειμενοστρεφής Προγραμματισμός

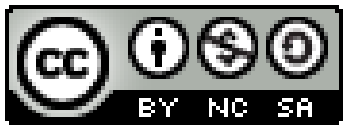
Ενότητα 8: Περαιτέρω Τεχνικές Αφαίρεσης

Γρηγόρης Τσουμάκας, Επικ. Καθηγητής
Τμήμα Πληροφορικής



Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περαιτέρω Τεχνικές Αφαίρεσης

Τα παραδείγματα κώδικα που χρησιμοποιούνται σε κάποιες από τις ακόλουθες διαφάνειες μπορούν να βρεθούν στον παρακάτω σύνδεσμο:
http://users.auth.gr/greg/oop.zip

Βασικές Έννοιες

- Αφηρημένες κλάσεις.
- Διασυνδέσεις (Interfaces).
- Πολλαπλή κληρονομικότητα.

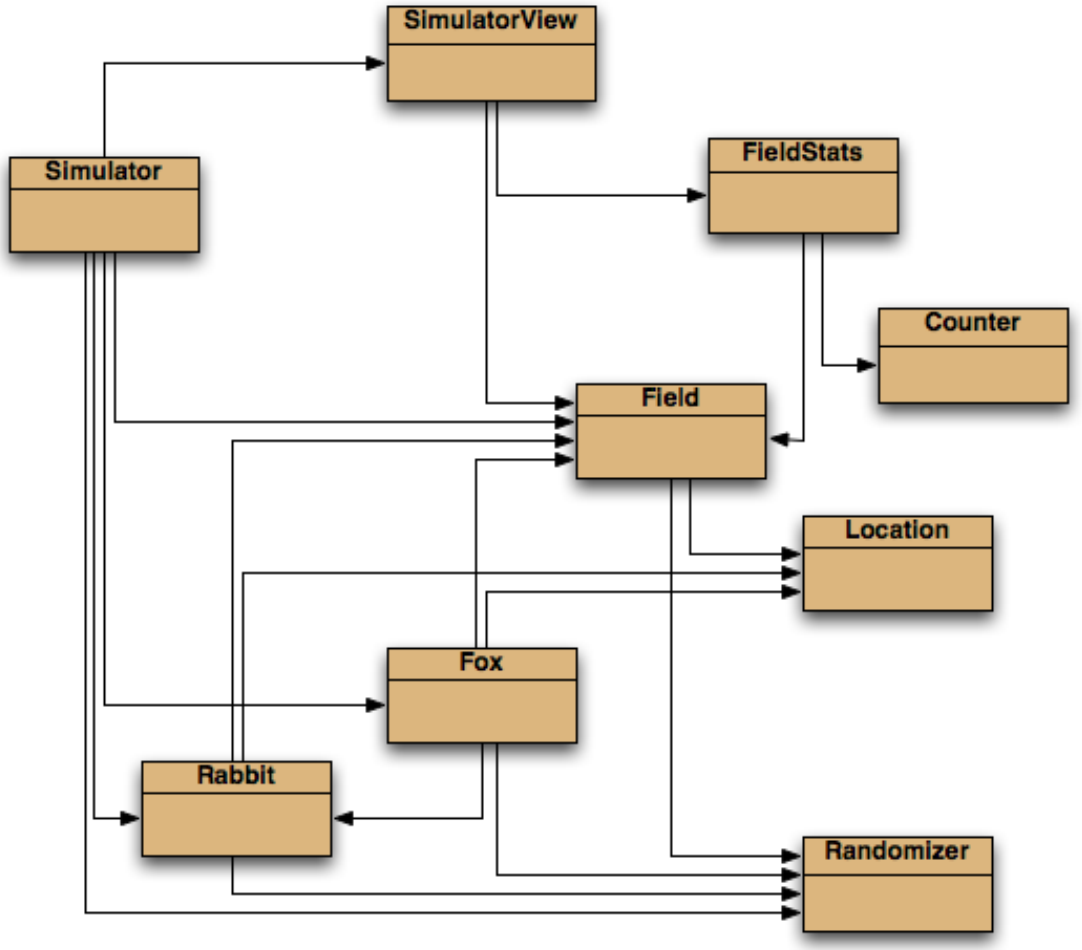
Προσομοίωση

- Χρήση λογισμικού για την προσομοίωση δραστηριοτήτων του πραγματικού κόσμου.
  - Κίνηση σε πόλεις, καιρός, πολεμικά σενάρια, …
- Απλουστευμένα σενάρια της πραγματικότητας.
  - Πιο πολλές λεπτομέρειες = μεγαλύτερη ακρίβεια, αλλά και μεγαλύτερες υπολογιστικές απαιτήσεις.
- Πλεονεκτήματα.
  - Χρήσιμες προβλέψεις (π.χ. καιρός).
  - Ασφαλής, φθηνός και γρήγορος πειραματισμός.

Προσομοίωση Θηρευτής-Θήραμα

- Λεπτή ισορροπία μεταξύ ειδών.
  - Πολύ θήραμα = πολύ φαγητό.
  - Πολύ φαγητό = πολλοί θηρευτές.
  - Πολλοί θηρευτές = λιγότερο φαγητό.
  - Λιγότερο φαγητό = …

Το Λογισμικό FoxesAndRabbits



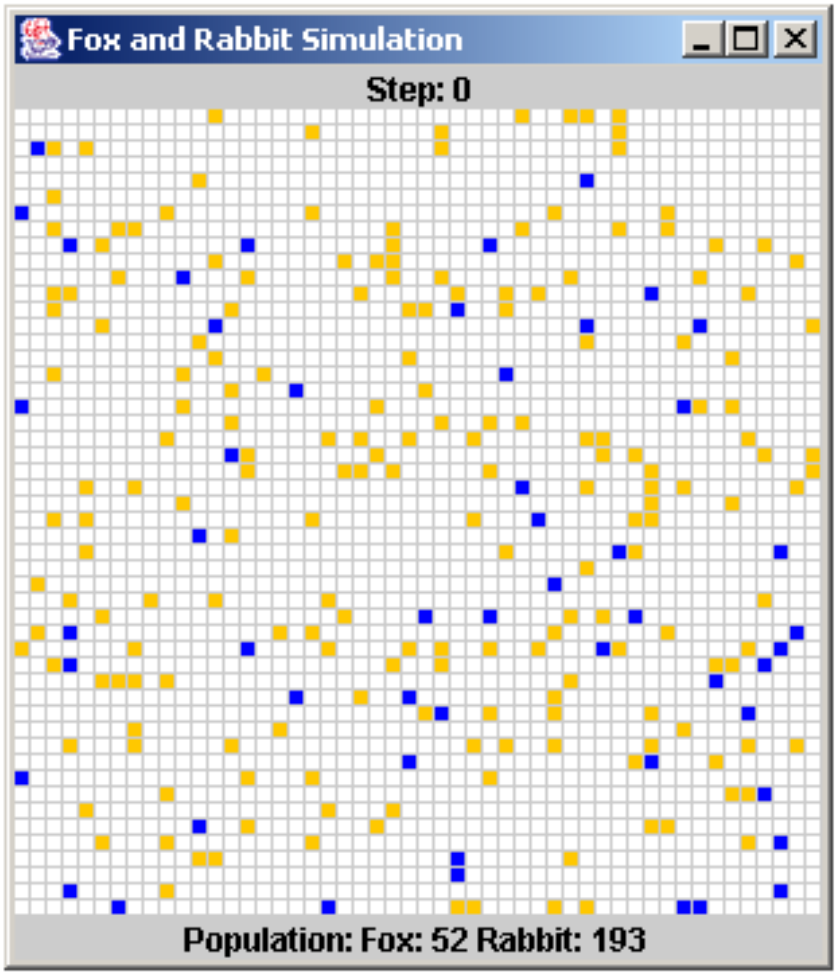
Βασικές Κλάσεις Ενδιαφέροντος

- Fox.
  - Απλό μοντέλο ενός θηρευτή.
- Rabbit.
  - Απλό μοντέλο ενός θηράματος.
- Simulator.
  - Διαχειρίζεται την προσομοίωση.
  - Διατηρεί δύο συλλογές από αντικείμενα των κλάσεων *Fox* και *Rabbit* αντίστοιχα.

Οι Υπόλοιπες Κλάσεις

- Field.
  - Διδιάστατο πλέγμα.
- Location.
  - Μια τοποθεσία σε διδιάστατο πλέγμα.
- SimulatorView.
  - Οπτικοποίηση του πλέγματος.
- FieldStats, Counter
  - Καταγραφή στατιστικών.
- Randomizer
  - Κεντρική διαχείριση τυχαιότητας.

Παράδειγμα Οπτικοποίησης



Η Κατάσταση ενός Λαγού

```
public class Rabbit {
    // Static fields omitted.

    private int age;
    private boolean alive;
    private Location location;
    private Field field;
}
```

Η Συμπεριφορά των Λαγών

- Υλοποιείται εντός της μεθόδου *run.*
- Αυξάνεται η ηλικία τους.
  - Ένας λαγός πεθαίνει όταν ξεπεράσει ένα άνω όριο ηλικίας (βήματα προσομοίωσης), MAX_AGE.
- Ενδέχεται να αναπαραχθούν .
  - Με πιθανότητα (BREEDING_PROB), εφόσον έχουν ξεπεράσει ένα κάτω όριο ηλικίας (BREEDING_AGE).
  - Ένας αριθμός νέων λαγών (MAX_LITTER_SIZE) τοποθετούνται σε διπλανά κελιά.
- Μετακινούνται σε γειτονικά κελιά.

Απλουστεύσεις Μοντέλου Λαγών

- Δεν υπάρχουν διαφορετικά φύλα.
  - Όλοι είναι θηλυκοί.
- Ο ίδιο λαγός ενδέχεται να αναπαραχθεί σε συνεχόμενα βήματα της προσομοίωσης.
- Όλοι οι λαγοί πεθαίνουν στην ίδια ηλικία.

Η Κατάσταση μιας Αλεπούς

```
public class Fox {
    // Static fields omitted.

    private int age;
    private boolean alive;
    private Location location;
    private Field field;
    private int foodLevel;
}
```

Η Συμπεριφορά των Αλεπούδων

- Υλοποιείται στη μέθοδο *hunt.*
- Μεγαλώνουν και αναπαράγονται με τον ίδιο τρόπο όπως και οι λαγοί.
- Αυξάνεται η πείνα τους (RABBIT_FOOD_VALUE).
- Ψάχνουν στα γειτονικά κελιά για φαγητό.

Απλουστεύσεις στις Αλεπούδες

- Παρόμοιες απλουστεύσεις με τους λαγούς.
- Διαφορετικός τρόπος μοντελοποίησης της πείνας και του κυνηγιού θηραμάτων.
  - Το επίπεδο φαγητού καλώς είναι αυξητικό;
  - Είναι ίδια η πιθανότητα για κυνήγι μιας πεινασμένης και μιας χορτάτης αλεπούς;
- Είναι αποδεκτές αυτές οι απλουστεύσεις;
  - Μόνο εφόσον βγαίνουν χρήσιμα συμπεράσματα.

Η Κλάση Simulator

- Κατασκευαστής και μέθοδος *reset.*
  - Αρχικοποιούν μια προσομοίωση.
- Η μέθοδος *populate.*
  - Γεμίζει τυχαία το πλέγμα με ζώα τυχαίας ηλικίας.
- Η μέθοδος *simulateOneStep.*
  - Διατρέχει τις δύο διαφορετικές συλλογές των ζώων και καλεί τις συμπεριφορές τους.

Ενημέρωση Προσομοίωσης (1/2)

```
step++;
ArrayList<Rabbit> newRabs = new ArrayList<>();
Iterator<Rabbit> it1 = rabbits.iterator();
while (it1.hasNext()) {
    Rabbit rabbit = it1.next();
    rabbit.run(newRabs);
    if (!rabbit.isAlive()) { it1.remove(); }
}
rabbits.addAll(newRabs);
```

Ενημέρωση Προσομοίωσης (2/2)

```
ArrayList<Rabbit> newFoxs = new ArrayList<>();
Iterator<Fox> it2 = foxes.iterator();
while (it2.hasNext()) {
    Fox fox = it2.next();
    fox.hunt(newFoxs);
    if(! fox.isAlive()) { it2.remove(); }
}
foxes.addAll(newFoxs);
view.showStatus(step, field);
```

Προβλήματα και Βελτιώσεις

- Προβλήματα.
  - Οι κλάσεις *Fox*, *Rabbit* έχουν πολλες ομοιότητες.
  - Η κλάση Simulator έχει υψηλή σύζευξη με τις κλάσεις *Fox* και *Rabbit,* και επανάληψη κώδικα.
- Βελτιώσεις;

Η Υπερκλάση Animal (1/2)

- Κοινά *private* πεδία στην *Animal.*
  - *age*, *alive*, *location*, *field.*
  - Το *age* σχετίζεται με *static final* πεδία (σταθερές), οπότε θα το εξετάσουμε αργότερα.
- Αρχικοποίηση.
  - *alive* = true, *location* και *field* από κατασκευαστή.
- Νέα μέθοδος πρόσβασης.
  - Νέα *protected* μέθοδος *getField* για πρόβαση στο πεδίο *field* από τις *run*, *hunt*, *giveBirth*, *findFood.*

Η Υπερκλάση Animal (2/2)

- Αλλαγές σε υπάρχουσες μεθόδους πρόσβασης και μετάλλαξης.

	Rabbit	Fox	Animal
getLocation	public	public	
setLocation	private	private	
isAlive	public	public	
setDead	public	private	

Αλλαγές στη Simulator (1/2)

```
ArrayList<Animal> newAnims = new ArrayList<>();

Iterator<Animal> it = animals.iterator();

while (it.hasNext()) {
   Animal animal = it.next();

   if (animal instanceof Rabbit)
     Rabbit rabbit = (Rabbit) animal;
     rabbit.run(newAnims);
   } else if (animal instanceof Fox){
     Fox fox = (Fox) animal;
     fox.hunt(newAnims);
   }

   if (!animal.isAlive()) {
      it.remove();
   }
}

animals.addAll(newAnims);
```

Αλλαγές στη Simulator (2/2)

- Μείωση σύξευξης *Simulator* και *Fox, Rabbit.*
  - Χρήση πολυμορφικής συλλογής με *Animal.*
  - Μετονομασία μεθόδων *run* και *hunt* σε *act.*

```
Iterator<Animal> it = animals.iterator();
while (it.hasNext()) {
    Animal animal = it.next();
    animal.act(newAnimals);
    if (!animal.isAlive()) {
        it.remove();
    }
}
```

Η Μέθοδος *act*

- Ο στατικός έλεγχος τύπου απαιτεί μέθοδο *act* στην κλάση *Animal.*
- Δεν υπάρχει προφανής κοινός κώδικας.
  - Η αύξηση της ηλικίας εξαρτάται από στατικές σταθερές των επί μέρους κλάσεων!
- Δήλωση μεθόδου *act* ως αφηρημένης.

```
public abstract void
act(ArrayList<Animal> newAnimals);
```

Αφηρημένες Κλάσεις και Μέθοδοι

- Αφηρημένες μέθοδοι.
  - Περιλαμβάνουν στην αρχή τη λέξη κλειδί *abstract.*
  - Δεν έχουν σώμα.
  - Εφόσον υπάρχουν σε μια κλάση, τότε και η κλάση πρέπει να δηλωθεί ως αφηρημένη.
- Αφηρημένες κλάσεις.
  - Στη δήλωση τους μπαίνει η λέξη κλειδί *abstract.*
  - Δεν μπορούμε να έχουμε αντικείμενα τους.
  - Δεν είναι υποχρεώτικό να έχουν αφηρημένες μεθόδους.

Χειρισμός του πεδίου age

```
private int age;
private static final int BREEDING_AGE = 5; // 15 fox
public boolean canBreed() {
  return age >= BREEDING_AGE;
}
```

Πριν (Rabbit)

```
protected int age;
protected abstract int getBreedingAge();
public boolean canBreed() {
  return age >= getBreedingAge();
}
```

Μετά (Animal)

```
private static final int BREEDING_AGE = 5; // 15 fox
public int getBreedingAge() {
  return BREEDING_AGE;
}
```

Μετά (Rabbit)

Περαιτέρω Αφαίρεση (1/3)

- Προσθήκη πρακτόρων που δεν είναι ζώα.
  - Άνθρωποι που κυνηγούν τις αλεπούδες.
  - Φυτά που αποτελούν τροφή για τους λαγούς.
  - Καιρός που επηρεάζει την ανάπτυξη των φυτών.
- Πως θα γενικεύαμε την προσομοίωση;

```
public abstract class Actor {
  public abstract void act(ArrayList<Actor> l);
  public abstract boolean isActive();
}
```

Περαιτέρω Αφαίρεση (2/3)

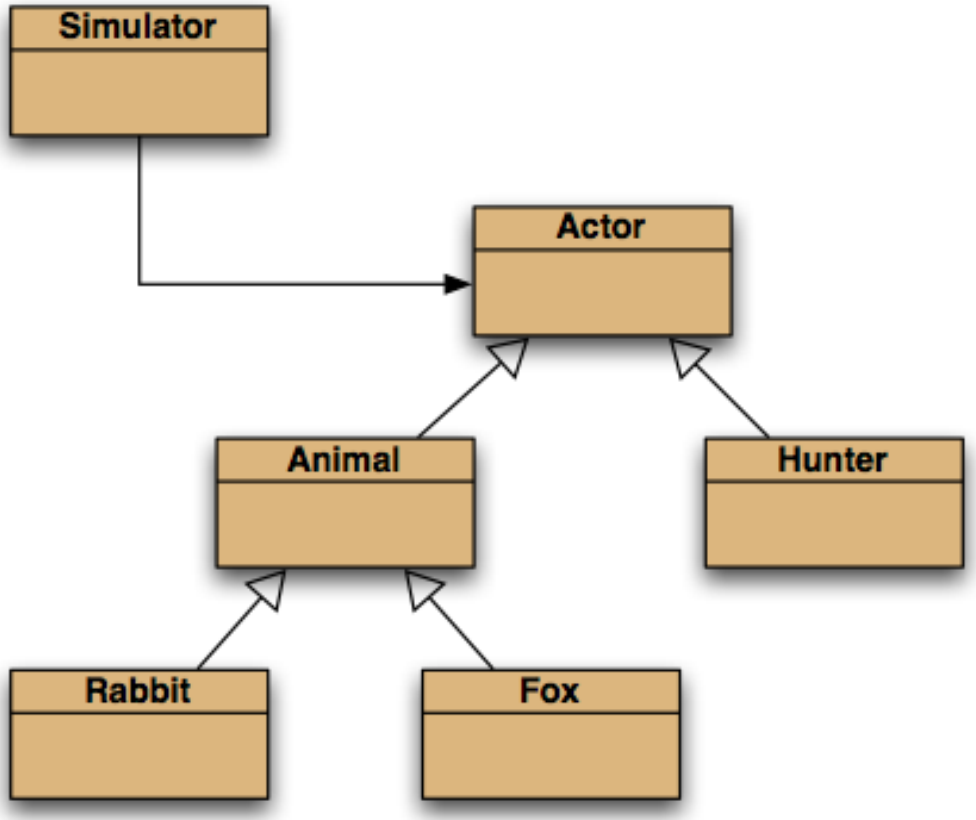
```
Iterator<Animal> it = animals.iterator();
while (it.hasNext()) {
    Animal animal = it.next();
    animal.act(newAnimals);
    if (!animal.isAlive()) {
        it.remove();
    }
}
```

Πριν

```
Iterator<Actor> it = actors.iterator();
while (it.hasNext()) {
    Actor actor = it.next();
    actor.act(newActors);
    if (!actor.isActive()) {
        it.remove();
   }
}
```

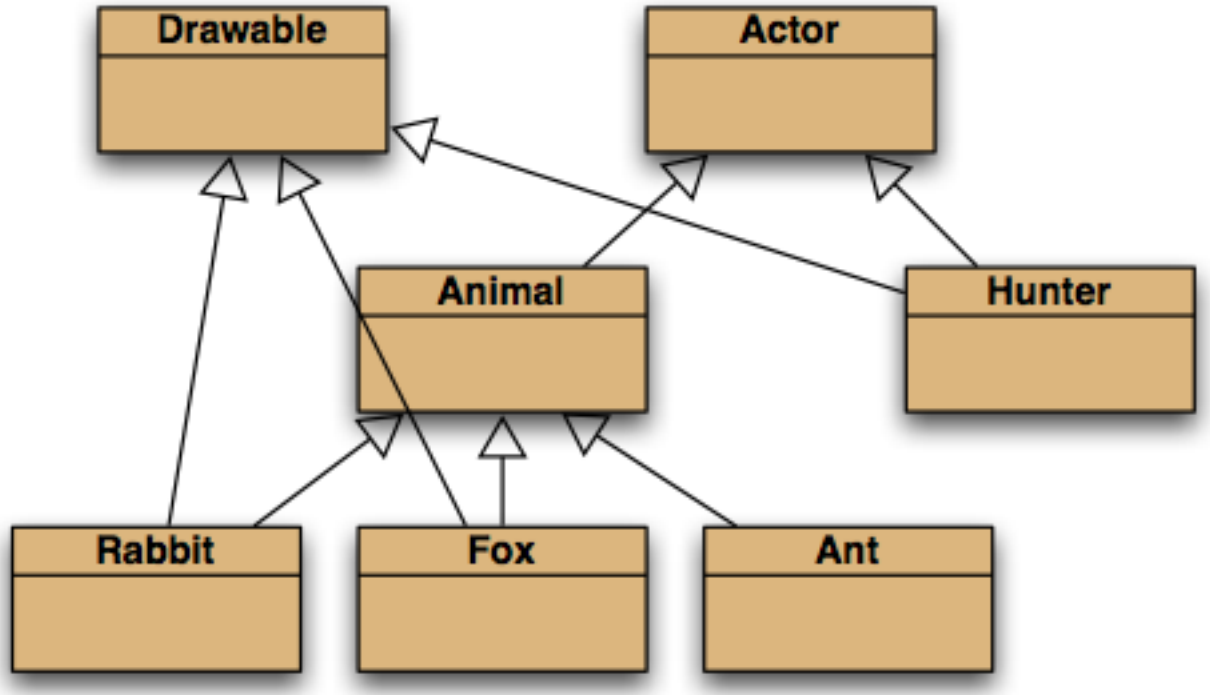
Μετά

Περαιτέρω Αφαίρεση (3/3)



Πολλαπλή Κληρονομικότητα (1/3)

- Επιλεκτική εμφάνιση των πρακτόρων της προσομοίωσης.



Πολλαπλή Κληρονομικότητα (2/3)

```
for (Actor actor : actors) {
  actor.act(newActors);
}

view.showStatus(step, field);
```

Τρέχουσα υλοποίηση

```
for (Actor actor : actors) {
  actor.act(newActors);
}

for (Drawable item : drawables) {
  item.draw();
}
```

Εναλλακτική υλοποίηση

Πολλαπλή Κληρονομικότητα (3/3)

- Πολλαπλή κληρονομικότητα.
  - Μια κλάση επεκτείνει περισσότερες από μία.
- Κάθε γλώσσα έχει τους δικούς της κανόνες.
  - Πως επιλύονται συγκρούσεις μεταξύ μεθόδων με ίδιες υπογραφές και διαφορετικές υλοποιήσεις;
- Η Java.
  - Δεν το επιτρέπει για κλάσεις.
  - Το επιτρέπει για διασυνδέσεις (interfaces), όπου δεν υπάρχουν συγκρουόμενες υλοποιήσεις.

Χαρακτηριστικά Διασυνδέσεων

- Όλες οι μέθοδοι είναι public και abstract.
  - Δεν χρειάζεται να το καθορίσουμε.
- Δεν υπάρχουν κατασκευαστές.
- Τα πεδία είναι δημόσιες στατικές σταθερές.

```
public interface Actor {
  void act(ArrayList<Actor> newActors);
  boolean isActive();
}

public interface Drawable {
  void draw();
}
```

Οι Κλάσεις Υλοποιούν Διασυνδέσεις

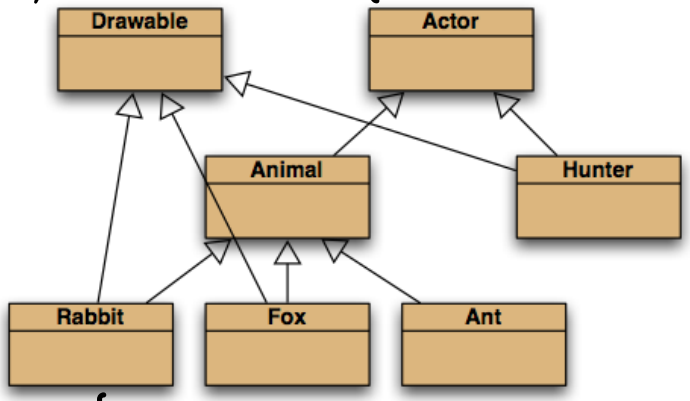
```
public class Animal implements Actor {
    ...
}

public class Fox extends Animal implements Drawable {
    ...
}

public class Hunter implements Actor, Drawable {
    ...
}

public class Ant extends Animal {
    ...
}

public class Weather implements Actor {
    ...
}
```



Διασυνδέσεις ως Τύποι

- Οι κλάσεις που υλοποιούν διασυνδέσεις.
  - Δεν κληρονομούν κώδικά, αλλά…
  - Αποτελούν υποτύπο του τύπου της διασύνδεσης, άρα επιτρέπουν την αξιοποίηση πολυμορφισμού.
- Παράδειγμα.

```
for (Actor actor : actors) {
  actor.act(newActors);
}
```

Διασυνδέσεις ως Προδιαγραφές

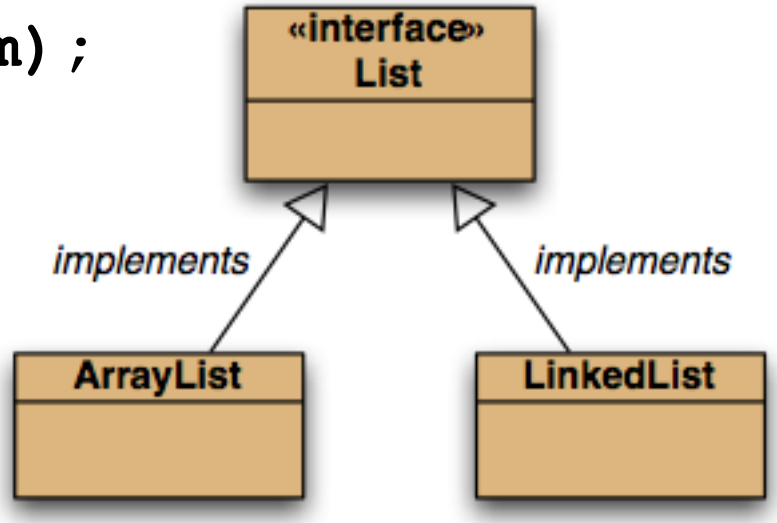
- Ισχυρός διαχωρισμός προσφερόμενων λειτουργιών και υλοποίησής τους.
  - Καθορισμός τύπου παραμέτρων και επιστρεφόμενων τιμών.
- Αλληλεπίδραση ανεξάρτητη της υλοποίησης.
  - Επιλογή ανάμεσα από εναλλακτικές υλοποιήσεις.
  - Εύκολη αλλαγή από υλοποίηση σε υλοποίηση.
  - Εύκολη υιοθέτηση νέων υλοποιήσεων.

Εναλλακτικές Υλοποιήσεις

```
List<String> list = new ArrayList<>();

list.add("Java");

for (String item : list) {
   System.out.println(item);
}
```



TestLists.java

Η Διασύνδεση Comparable<T>

- Ορίζει μέθοδο int compareTo(T obj);
  - Επιστρέφει αρνητικό, 0, ή θετικό ανάλογα αν το obj προηγείται, ισούται ή έπεται του τρέχοντος.
- Χρησιμοποιείται στο πλαίσιο συλλογών.
  - Collections.sort()
  - TreeSet, TreeMap

TestComparable.java

Σύνοψη (1/2)

- Η κληρονομικότητα προσφέρει
  - Επαναχρησιμοποίηση κώδικα μέσω αφηρημένων και μη αφηρημένων κλάσεων.
  - Κοινούς τύπους για διαφορετικά αντικείμενα μέσω αφηρημένων και μη αφηρημένων κλάσεων, αλλλά και μέσω διασυνδέσεων.

Σύνοψη (2/2)

- Οι αφηρημένες κλάσεις
  - Λειτουργούν ως ημιτελείς υπερκλάσεις.
  - Οι αφηρημένες μέθοδοι επιτρέπουν στατικό έλεγχο τύπου χωρίς να απαιτούν υλοποίηση.
  - Υποστηρίζουν τον πολυμορφισμό.
- Οι διασυνδέσεις
  - Παρέχουν προδιαγραφές δίχως υλοποίηση.
  - Υποστηρίζουν τον πολυμορφισμό.
  - Υποστηρίζουν την πολλαπλή κληρονομικότητα.

Αφηρημένη Κλάση ή Διασύνδεση;

- Αν θα περιλαμβάνει υλοποιήσεις.
  - Αφηρημένη κλάση.
- Αν δεν θα περιλαμβάνει υλοποιήσεις.
  - Οι διασυνδέσεις είναι λιγότερο περιοριστικές, καθώς επιτρέπουν πολλαπλή κληρονομικότητα.

Άσκηση

- Τι λάθη εντοπίζετε στον ορισμό της ακόλουθης διασύνδεσης;

```
public interface Monitor {
   private static final int THRESHOLD=50;
   public Monitor(int initial);
   public int getThreshold() {
      return THRESHOLD;
   }
}
```

Άσκηση

- Έστω 4 μεταβλητές αντίστοιχων κλάσεων:
  - `O o; X x; T t; M m;`
- Οι παρακάτω αναθέσεις είναι αποδεκτές:
  - `m = t; m = x; o = t;`
- Οι παρακάτω αναθέσεις ΔΕΝ είναι αποδεκτές:
  - `o = m; o = x; x = o;`
- Τι συμπεράσματα βγάζετε για τις σχέσεις των κλάσεων;

Άσκηση

- Έστω 5 μεταβλητές αντίστοιχων κλάσεων ή διασυνδέσεων:
  - `U u; G g; B b; Z z; X x;`
- Οι παρακάτω αναθέσεις είναι αποδεκτές:
  - `u = z; x = b; g = u; x = u;`
- Οι παρακάτω αναθέσεις ΔΕΝ είναι αποδεκτές:
  - `u = b; x = g; b = u; z = u; g = x;`
- Τι συμπεράσματα βγάζετε για τους τύπους και τις μεταξύ τους σχέσεις;

Άσκηση

- Έστω ότι υλοποιείτε ένα πληροφοριακό σύστημα διαχείρισης τμημάτων ΑΕΙ.
- Ορίστε ιεραρχία τύπων (διασυνδέσεις, κλάσεις, αφηρημένες κλάσεις) για τα άτομα των ΑΕΙ:
  - Προσωπικό, φοιτητές, διδακτικό προσωπικό, υποστηρικτικό προσωπικό, βοηθοί μαθήματος (φοιτητές που διδάσκουν), τεχνικό υποστηρικτικό προσωπικό, φοιτητές τεχνικοί (φοιτητές που κάνουν τεχνική υποστήριξη).

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Τέλος Ενότητας

Επεξεργασία: Εμμανουήλ Ρήγας

Θεσσαλονίκη, Εαρινό Εξάμηνο 2013-2014

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης