Αριστοτέλειο
Πανεπιστήμιο
Θεσσαλονίκης

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΑΝΑΛΥΣΗ

ΥΛΟΠΟΙΗΣΗ ΣΤΟΙΧΕΙΩΝ ΑΝΑΛΥΣΗΣ UML ΜΕ JAVA

Ιωάννης Σταμέλος
Βάιος Κολοφωτιάς
Πληροφορική

Θεσσαλονίκη, Σεπτέμβριος 2013

Άδειες Χρήσης

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 τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

ανοικτάμαθήματα
opencourses

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.

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

Περιεχόμενα

1. Περιεχόμενο Μαθήματος

Εβδομάδα	**Περιεχόμενο**
1η	Εισαγωγή στην Αντικειμενοστρεφή Ανάλυση/UML
2η	Rational Unified Process
3η	Περιπτώσεις Χρήσης
4η	Διαγράμματα Κλάσεων
5η	Διαγράμματα Συνεργασίας
6η	Διαγράμματα Ακολουθίας
7η	Πρότυπα Σχεδίασης
8η	Διεργασία ICONIX
9η	Επιχειρηματική Μοντελοποίηση
10η	**Υλοποίηση Σχεδίασης με Java**
11η	Μετρικές Αντικειμενοστραφούς Σχεδίασης
12η	Επισκόπηση

1.1 Περιεχόμενα ενότητας

Στην ενότητα αυτή θα δούμε τις συσχετίσεις και πως μπορούν να υλοποιηθούν στη Java. Στις συσχετίσεις που θα δούμε περιλαμβάνονται η υλοποίηση πεδίων συσχέτισης, συσχέτισης διπλής κατεύθυνσης, δυαδικής συσχέτισης, πολλά προς ένα, συνάθροισης και σύνθεσης.

2. Συσχετίσεις *(Associations)*

Μία συσχέτιση έχει **όνομα** που γράφεται σαν ετικέτα στο μέσο της γραμμής σύνδεσης (ρήμα). Το όνομα φανερώνει το είδος της συσχέτισης.

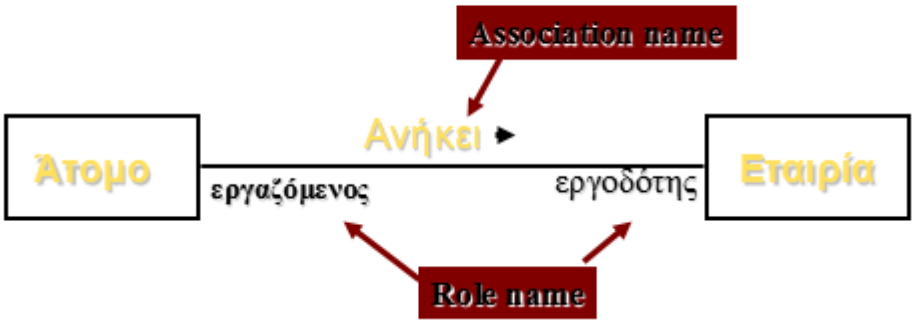
Μία κλάση που μετέχει σε μία συσχέτιση παίζει ένα **ρόλο** (role) σε αυτήν. Γράφουμε το **όνομα του ρόλου** (ουσιαστικό) στο τέλος της σχέσης, δηλαδή στο σημείο σύνδεσης με την κλάση.

3. Υλοποίηση Συσχέτισης (Association)

Δεν υποστηρίζεται άμεσα από τη Java (όπως π.χ. και η συνάθροιση)

Πρέπει όμως να υλοποιείται με τυποποιημένο τρόπο για

- Εύκολη και επαναλήψιμη υλοποίηση
- Ομοιομορφία κώδικα
- Ευκολία κατανόησης από άλλους προγραμματιστές (πρότυπο πρ/σμού!)
- Τυποποίηση των δοκιμών



Εικόνα 1 : Συσχέτιση

<u>Πολλαπλότητα *(multiplicity)*</u>

- **Καθορίζει πόσα αντικείμενα μετέχουν σε μια συσχέτιση (από 0 έως οποιοδήποτε πλήθος).**

✓ Ακριβώς ένα	1
✓ Μηδέν ή ένα	0..1
✓ Μηδέν ή περισσότερα	0..*
✓ Ένα ή περισσότερα	1..*
✓ Από..Έως	m..n
✓ Πολλαπλά διαφορετικά όρια	2, 4..6, 8

Εικόνα 2 : Πολλαπλότητα

- Μία δυαδική συσχέτιση (ένα προς ένα) συνήθως υλοποιείται με ένα πεδίο σε μία ή και στις δύο κλάσεις που συμμετέχουν στη συσχέτιση
- Το πεδίο είναι μία αναφορά στο άλλο αντικείμενο

- Μία συσχέτιση πολλά-προς-πολλά υλοποιείται με
    - ένα σύνολο αντικειμένων
- με ένα μονοδιάστατο πίνακα (array) αντικειμένων (αν τα αντικείμενα έχουν μία προκαθορισμένη σειρά): μπορεί να χρησιμοποιηθεί η κλάση Vector **σε μία ή και στις δύο κλάσεις που συμμετέχουν στη συσχέτιση**
- Αν η συσχέτιση είναι μονής κατεύθυνσης αρκεί ένα πεδίο σε μία από τις δύο κλάσεις
- Αν η συσχέτιση είναι διπλής κατεύθυνσης υπάρχουν διάφοροι τρόποι:

4. Συσχετίσεις

4.1 Υλοποίηση Πεδίων Συσχέτισης (Association Attributes)

- Αν η συσχέτιση είναι ένα-προς-ένα, τα πεδία της συσχέτισης μπορούν να ανήκουν σε μία από τις δύο κλάσεις
- Αν η συσχέτιση είναι πολλά-προς-ένα, τα πεδία είναι προτιμότερο να τοποθετηθούν στην κλάση από την πλευρά ‘πολλά’
- Αν η συσχέτιση είναι πολλά-προς-πολλά, είναι καλύτερο να δημιουργηθεί μία κλάση για τη συσχέτιση στην οποία και θα τοποθετηθούν τα πεδία της συσχέτισης

4.2 Υλοποίηση Συσχέτισης Διπλής Κατεύθυνσης

1ος τρόπος υλοποίησης: Με ένα πεδίο στη μία κλάση και εφαρμόζοντας μία αναζήτηση όταν απαιτείται η εφαρμογή της συσχέτισης από την άλλη κατεύθυνση

Συνίσταται όταν

- Η συσχέτιση είναι του τύπου πολλά-προς-ένα και διασχίζεται από τα πολλά προς το ένα και
- Νέα μέλη προστίθενται συχνά

2ος τρόπος υλοποίησης: Με ένα πεδίο και στις δύο κλάσεις

Συνίσταται όταν

- Δεν προστίθενται συχνά νέα μέλη
- Η ταχύτητα εκτέλεσης του προγράμματος πρέπει να είναι μεγάλη

3ος τρόπος υλοποίησης: Με την εισαγωγή μίας πρόσθετης κλάσης (associative class) μεταξύ των δύο κλάσεων

Συνίσταται όταν

- Απαιτείται ευελιξία για μελλοντικές επεκτάσεις και
- Η ταχύτητα εκτέλεσης του προγράμματος δεν πρέπει να είναι μεγάλη

4.3 Υλοποίηση Δυαδικής Συσχέτισης

Υλοποιείται η συσχέτιση 'σύζυγος' (spouse) για την κλάση Person

- **Private Person spouse;** **υλοποιεί τη συσχέτιση**
- **setSpouse(Person p)** **υλοποιεί τη συσχέτιση από τη μία πλευρά**
- **getSpouse()** **υλοποιεί τη 'διάσχιση' της σχέσης**

```
Public class Person
{
//Services
publicPerson(String n,  char s,  int a)  {…..code here….}
publicPerson getSpouse()  {return spouse;}
public void setSpouse(Person p)  {spouse =p ; return;  }
……..
Private String name;
Private char sex;
Private int age;……..
Private Person spouse;
}
```

4.4 Υλοποίηση Συσχέτισης πολλά-προς-ένα: δεν ενδιαφέρει η ταχύτητα εκτέλεσης

```
Public class Person
{
//services
public Person(String n, char s, int a, Person pop, Person mom)
{……code here……}
public Person getFather() {return father; }
public Person getMother() {return mother; }
public Vector  getChildren()  {…need to search for all references to yourself..}
private void setFather(Person p) {father=p; return;}
private void setMother(Person p) {mother=p; return;}
//Variables
private String name;
private char sex;
private int age;….
private Person father;
private Person mother;
}
```

4.5 Υλοποίηση Συσχέτισης πολλά-προς-ένα: ενδιαφέρει η ταχύτητα εκτέλεσης

```
Public class Person
{
//Services
public Person(String n, char s, int a, person mom, person dad)
{….code here…}
public Person getFather() {return father;}
public Person getMother() {return mother;}
public Vector getChildren()  {return children;}
public boolean removeChild(Person p)
```

```
        {Boolean x = children.remove(p); return x;}
public boolean addChild(Person p)
        {Boolean x=children.add(p);
        return x;}
private void setFather(Person p)
{ father=p; if (p.addChild(this)  )  then return; else
"error"  }
privatevoid setMother(Person p)
{ mother=p; if (p.addChild(this)  )  then return; else
"error"  }
//Variables
private String name;
private char sex;
private int age;
private Person father;
private Person mother;
private Vector children;
}
```

4.6 Υλοποίηση Συνάθροισης και Σύνθεσης (Aggregation - Composition)

- Υλοποίηση ανάλογη με αυτή της συσχέτισης
- Υπενθυμίζεται ότι
  - Η συνάθροιση είναι μία πιο 'σθεναρή' σχέση συσχέτισης και αναπαριστά μία σχέση Όλου-Μέρους: το Όλο υλοποιείται από το 'περιέχον' αντικείμενο και τα Μέρη από τα 'περιεχόμενα' αντικείμενα
  - Η σύνθεση είναι μία πολύ ισχυρή σχέση Όλου-Μέρους: τα Μέρη δεν μπορούν να υπάρχουν ανεξάρτητα από το Όλο

4.6.1 Συνάθροιση

- Μονοδιάστατος πίνακας ‘δεικτών’-αναφορών σε αντικείμενα
- Χρησιμοποιείται όταν
  - Τα περιεχόμενα αντικείμενα μπορεί να αντικαθιστώνται από άλλα αντικείμενα
  - Θέλουμε να έχουμε άμεση πρόσβαση στα περιεχόμενα αντικείμενα από το περιέχον αντικείμενο

4.6.2 Σύνθεση

- Με ξεχωριστά αντικείμενα που ανήκουν στο περιέχον
- Χρησιμοποιείται επειδή θέλουμε τα περιεχόμενα αντικείμενα να τροποποιούνται μόνο από τις μεθόδους του περιέχοντος αντικειμένου

4.7 Δημιουργία ΚΣ

- Δημιουργείται το αντικείμενο του Κεντρικού Συστήματος (ΚΣ) για δίκτυο 5 κόμβων

```
KentrikoSystima kSys = new KentrikoSystima(5);
```

4.8 Δημιουργία Κόμβων

- Δημιουργούνται ανεξάρτητα οι 5 κόμβοι που αποτελούν αυτόνομες μονάδες. Ο κάθε κόμβος κατά την δημιουργία του ενσωματώνει (**composition**) ανάλογα 2-4 διαφορετικούς αισθητήρες (Ανέμου, Θερμοκρασίας, Υγρασίας, Ηλιοφάνειας).

```
Komvos ioannina = new Komvos("Ioannina..", new AisthitirasThermo('U'), new AisthitirasHliofaneias('H'));
```

4.9 Προσθήκη κόμβων στο ΚΣ

- Οι κόμβοι στη συνέχεια προστίθενται (εντάσσονται) ένας-ένας στο δίκτυο του ΚΣ. Η μορφή της συσχέτισης είναι η **aggregation**
- Οι κόμβοι δεν ανήκουν αποκλειστικά στο ΚΣ. Θα μπορούσαν να ενταχθούν και σε άλλα συστήματα (πχ. Γεωργικής εκμετ/σης)

kSys.addKomvos(ioannina);

4.10 Αμφίδρομη επικοινωνία

- Περνώντας το ΚΣ την αναφορά του **this** (του εαυτού του) στον Κόμβο, δίνει τη δυνατότητα στον κάθε κόμβο να του στείλει τις μετρήσεις του. Ετσι, επιτυγχάνεται **αμφίδρομη επικοινωνία** μέσω ΚΣ - κόμβων

komvoi[k].dwseMetriseis(this, xronos);
(εντολή που περιέχεται στο kSys)

4.11 **Συσχετίσεις του διαγράμματος κλάσεων**

- Κεντρικό σύστημα (ΚΣ) <>----- Κόμβος, **Aggregation**, διότι το ΚΣ «συγκροτεί» έναν αριθμό Κόμβων (δίκτυο). Η επικοινωνία μεταξύ των αντικειμένων των δύο κλάσεων είναι ΑΜΦΙΔΡΟΜΗ ώστε το ΚΣ να ζητάει και οι κόμβοι να του απαντούν με την επιστροφή των μετρήσεών τους. Οι κόμβοι προστίθενται στο ΚΣ.

private Komvos komvoi[];
kSys.addKomvos(ioannina);

- Κόμβος ◆----- Αισθητήρας, **Σύνθεση (Composition)**, διότι μόνο ο κόμβος γνωρίζει τους αισθητήρες του. Η υλοποίηση της *σύνθεσης* επιτυγχάνεται με την **ταυτόχρονη δημιουργία** Κόμβου και Αισθητήρων

```
private Aisthitiras aisthitires [];
Komvos ioannina = new Komvos("Ioannina..", new AisthitirasThermo('U'), new AisthitirasHliofaneias('H'));
```

- ΚΣ → Αρχείο, **Σύνδεση (association)**, διότι υπάρχει μόνιμη επικοινωνία που αφορά την καταχώρηση και ανάκληση των εγγραφών-κινήσεων.

- Αρχείο ----> ΚίνησηΜέτρησης **Εξάρτηση (Dependency)**, διότι ουσιαστικά οι εγγραφές που καταχωρούνται στο αρχείο είναι «αντίγραφα» υπό μορφήν κειμένου της 'ΚίνησηΜέτρησης'.

- Αισθητήρας **<|---** ΑισθητήραςΥγρασίας, κλπ, **Κληρονομικότητα**, διότι «**είναι είδος**» Αισθητήρα.

```
public class AisthitirasThermo extends Aisthitiras  {
        protected double min = -10;   // Υπέρβαση
        protected double max = 40;
```

4.12 Υλοποίηση πολυμορφισμού

- Ο Κόμβος μέσα από την μέθοδό του dwseMetriseis(….) στέλνει το ίδιο μήνυμα στους διάφορους αισθητήρες που διαθέτει, για λήψη μέτρησης, και ο κάθε ένας απαντάει με τον δικό του τρόπο που γνωρίζει. Ετσι, ο 'ΑισθητήραςΘερμοκρασίας' απαντάει με βαθμούς Κελσίου, ενώ ο 'ΑισθητήραςΑνέμων' με μποφόρ, κλπ.
-

```
aisthitires[a].getMetrisi();
```

- Για να υλοποιηθεί αυτό, στην κλάση Κόμβος ορίζεται ένας πίνακας αντικειμένων της **υπερκλάσης Αισθητήρας!!.** Ετσι, σε κάθε στοιχείο του πίνακα μπορεί να αποθηκευτεί ένα αντικείμενο των υποκλάσεων του Αισθητήρα (ΑισθητήραςΥγρασίας, ΑισθητήραςΑνέμου, κλπ), και αυτό επειδή και αυτά είναι αντικείμενα τύπου Αισθητήρα.
-

```
private Aisthitiras aisthitires [];
```