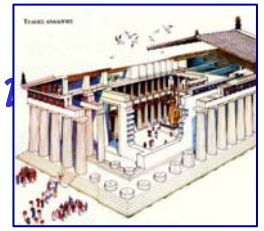
ΗΥ **351:**
Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
Information Systems Analysis and Design



# Φροντιστήριο 3

Ημερομηνία: 22/11/2006

Θεματική Ενότητα: **Δομική Μοντελοποίηση**

Θέμα: **Διαγράμματα Κλάσεων**

# Περιεχόμενα και Διάρθρωση

- Εισαγωγή
- Παραδείγματα Δομικής Μοντελοποίησης
- Συχνά λάθη
- Τρόπος σύνταξης διαγραμμάτων κλάσεων με CASE tools (demo)
- Υλοποίηση Διαγραμμάτων Κλάσεων και Java

# Τι είναι μοντέλο και γιατί μοντελοποιούμε

- Μοντέλο: Μια αφαίρεση (απλούστευση) της πραγματικότητας
  - εστιάζει στα σημαντικά, κρύβει τις άσχετες πλευρές και τις δευτερεύουσας σημασίας λεπτομέρειες
- Γιατί μοντελοποιούμε;
  - Ένα μοντέλο μας επιτρέπει την καλύτερη κατανόηση ενός συστήματος
  - Συνήθως φτιάχνουμε μοντέλα σύνθετων συστημάτων τα οποία δεν μπορούμε να κατανοήσουμε στην πληρότητα τους (ένεκα των περιορισμένων μας αντιληπτικών και διανοητικών ικανοτήτων)
  - Μοντελοποιώντας περιορίζουμε το πρόβλημα εστιάζοντας σε επιμέρους πλευρές του συστήματος (διαίρει και βασίλευε) και κλίμακες αφαίρεσης.
- Καλά μοντέλα είναι εκείνα που συνδέονται με την πραγματικότητα

# Μοντελοποίηση στην Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων

- Λειτουργική Μοντελοποίηση
  - Περιπτώσεων Χρήσης (Use Case), Δραστηριοτήτων (Activity)
- Δομική Μοντελοποίηση
  - Κλάσεων (Class), Αντικειμένων (Object), Πακέτων (Package), Παράταξης (Deployment) , Εξαρτημάτων (Component) , Σύνθετης Δομής (Composite Structure)
- Συμπεριφοράς Μοντελοποίηση
  - Sequence (Αλληλουχίας), Επικοινωνίας (Communication), Χρονισμού (Timing), Καταστάσεων (State), Interaction Overview, Protocol State Machine

# Παραδείγματα Δομικής Μοντελοποίησης


# Παράδειγμα 1: Μοντελοποίηση Κειμενογράφου

- Κάθε έγγραφο αποτελείται από ένα αριθμό σελίδων
- Κάθε σελίδα αποτελείται από μία επικεφαλίδα, το κυρίως κείμενο (σώμα) και κάποιο υποσέλιδο (footer)
- Στην επικεφαλίδα και στο υποσέλιδο μπορεί να προστεθούν στοιχεία όπως η ημέρα, η ώρα, ο αριθμός της σελίδας
- Το σώμα της σελίδας αποτελείται από προτάσεις που περιέχουν λέξεις και σημεία στίξης
- Κάθε λέξη αποτελείται από γράμματα ,αριθμούς και ειδικούς χαρακτήρες
- Το κείμενο μπορεί να περιέχει επίσης εικόνες και πίνακες
- Κάθε πίνακας έχει γραμμές και στήλες
- Κάθε κελί ενός πίνακα μπορεί να περιέχει εικόνες ή κείμενο
- Ο χρήστης μπορεί να δημιουργήσει ένα νέο έγγραφο ή να ανοίξει και να τροποποιήσει ένα υπάρχον
- Ο χρήστης μπορεί να επιλέξει να αποθηκεύσει ή να εκτυπώσει το έγγραφο

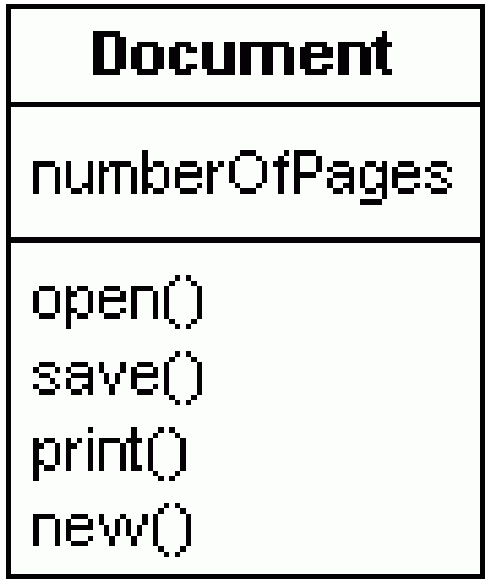
# Παράδειγμα 1: Μοντελοποίηση Κειμενογράφου

- Αν εξάγουμε τη λίστα με τα ουσιαστικά από την προηγούμενη περιγραφή θα έχουμε τα εξής:
  - έγγραφο, κείμενο, επικεφαλίδα, υποσέλιδο, ημέρα, ώρα, χρόνος, αριθμός σελίδας, σελίδα, πρόταση, λέξη, σημεία στίξης, γράμμα, αριθμός, ειδικός χαρακτήρας, εικόνα, πίνακας, γραμμή, στήλη, κελί, χρήστης

- Όλα τα ουσιαστικά αυτά αποτελούν υποψήφιες κλάσεις και χαρακτηριστικά κλάσεων του class diagram

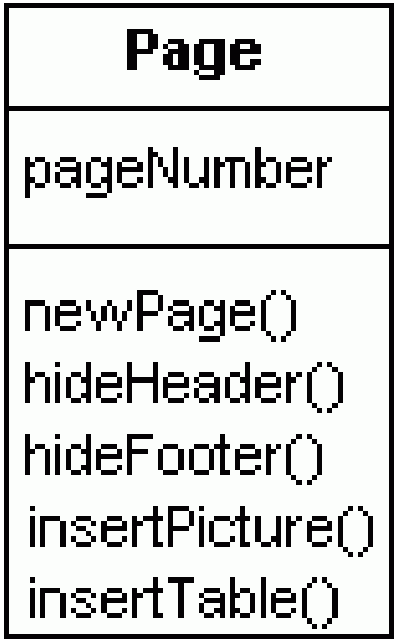
# Παράδειγμα 1: Έγγραφο (Document)

- Αποτελεί τη βασική έννοια του πεδίου εφαρμογής
- Ένα έγγραφο περιέχει ένα αριθμό σελίδων (*numberOfPages*)
- Οι λειτουργίες που επιτελούνται σε ένα έγγραφο είναι άνοιγμα (*open()*), αποθήκευση (*save()*), εκτύπωση (print()), δημιουργία (*new()*)



# Παράδειγμα 1: Σελίδα

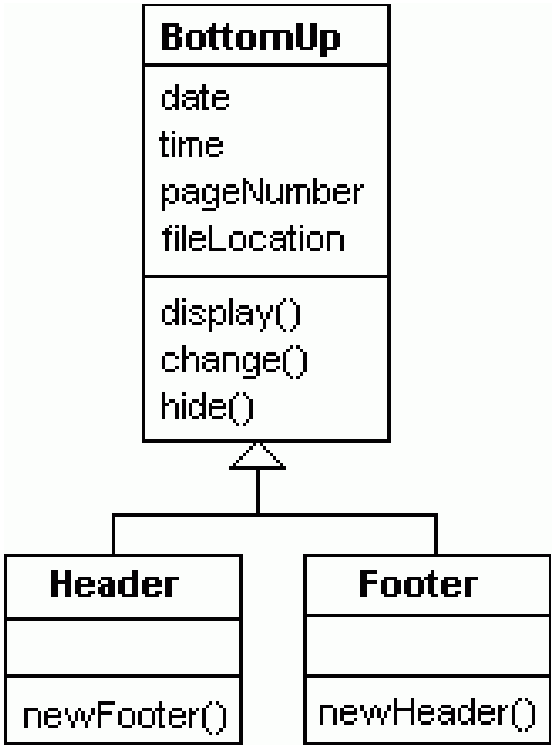
- Μία σελίδα έχει κάποιον αριθμό σελίδας
- Οι λειτουργίες πάνω σε μία σελίδα είναι οι εξής:
  - ***Δημιουργία σελίδας*** (*newPage()*)
  - ***Απόκρυψη επικεφαλίδας*** (*hideHeader()*)
  - ***Απόκρυψη υποσέλιδου*** (*hideFooter()*)
  - **Εισαγωγή εικόνας** (insertPicture())
  - **Εισαγωγή Πίνακα** (insertTable())



# Παράδειγμα 1: Επικεφαλίδα και Υποσέλιδο

- Αυτές οι κλάσεις έχουν κοινά χαρακτηριστικά την ημέρα (*date*), ώρα (*time*), αριθμό σελίδας (*pageNumber*), τοποθεσία αρχείου (*fileLocation*) και μπορούν να εμφανίζονται (*display* ()), να τροποποιούνται (*change* ()) ή να αποκρύπτονται (*hide* ())

*•Μπορεί να έχουμε δημιουργία νέου υποσέλιδου (newFooter())*

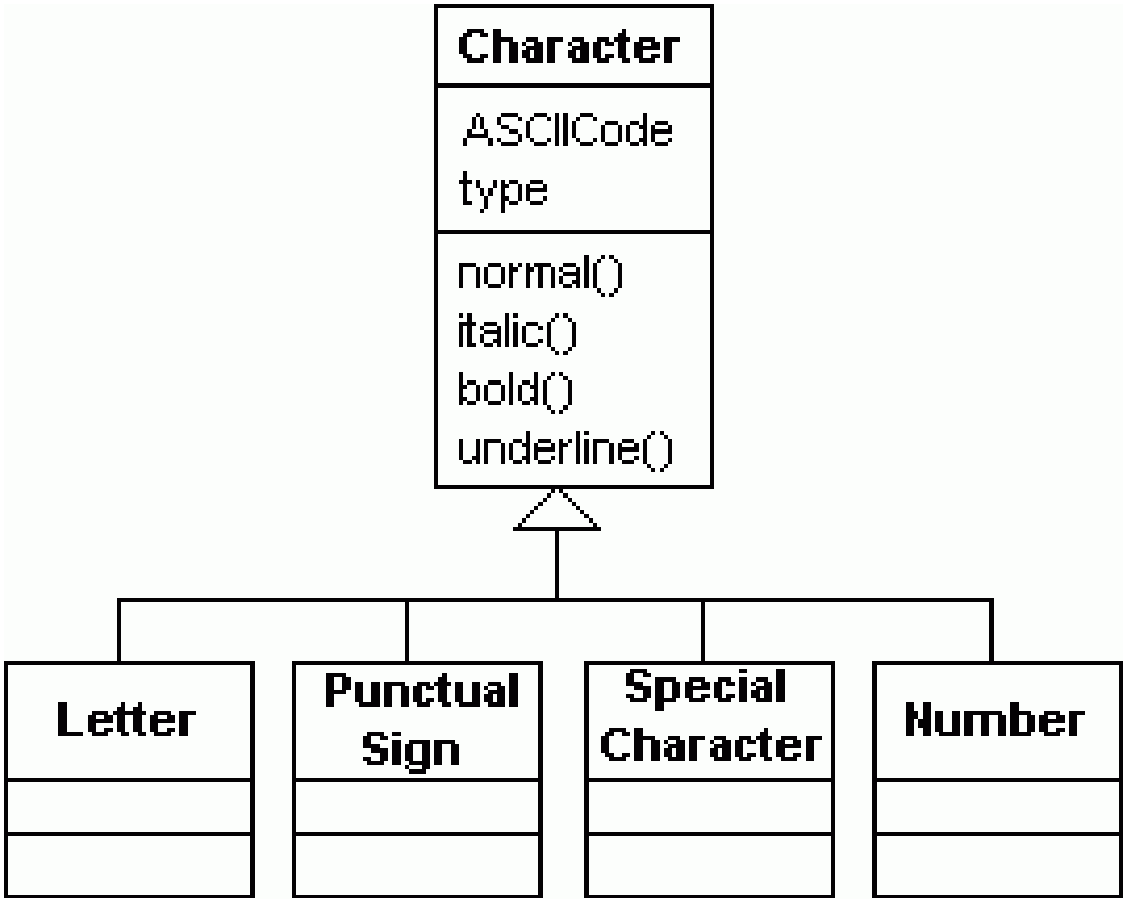



*•Μπορεί να έχουμε δημιουργία νέας επικεφαλίδας (newHeader())*


# Παράδειγμα 1: Προτάσεις, λέξεις, χαρακτήρες

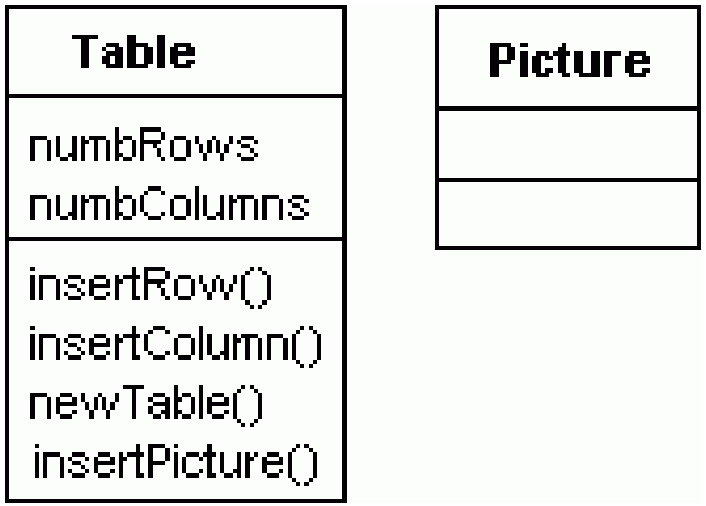
- Το σώμα αποτελείται από προτάσεις, οι προτάσεις από λέξεις και οι λέξεις από χαρακτήρες (Character)
- Άρα το σώμα αποτελείται από χαρακτήρες οι οποίοι μπορεί να είναι γράμματα (Letter), αριθμοί(Number), ειδικοί χαρακτήρες (Special Character) ή (σημεία στίξης) punctual signs
- Κάθε χαρατήρας έχει έναν τύπο (type) που καθορίζει αν είναι normal, italic, bold ή underline
- Σε κάθε χαρακτήρα μπορούν να επιτελεστούν λειτουργίες αλλαγής του τύπου : *normal()*, *bold()*, *italic()* και *underline()*

# Παράδειγμα 1: Προτάσεις, λέξεις, χαρακτήρες

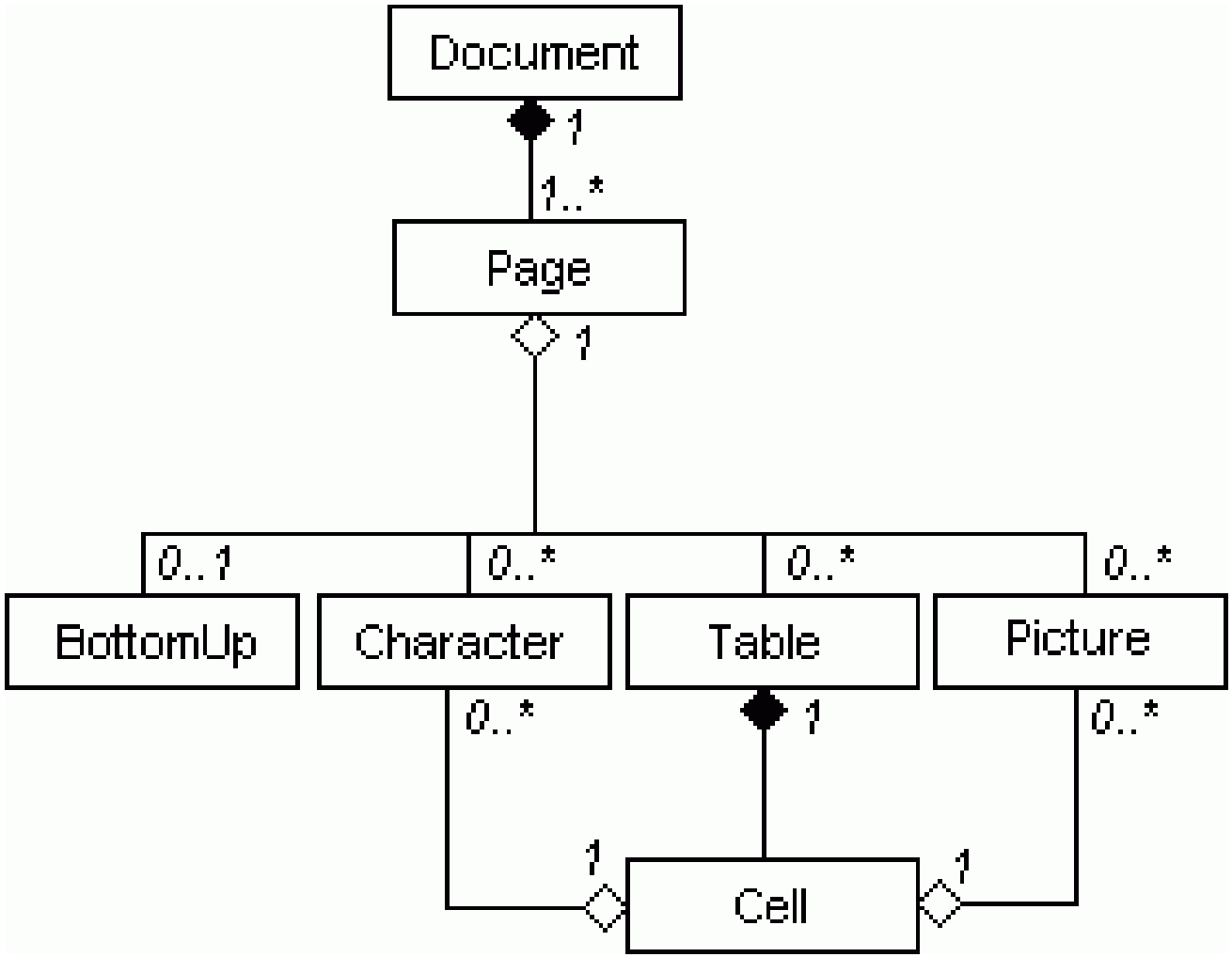


# Παράδειγμα 1: Πίνακες

- Το κείμενο μπορεί να περιέχει επίσης και πίνακες (Table)
- Κάθε πίνακας έχει γραμμές (*numbRows*) και στήλες (*numbColumns*)
- Οι λειτουργίες πάνω σε ένα πίνακα είναι: δημιουργία γραμμής (*insertRow()*),δημιουργία στήλης (*insertColumn()*) η δημιουργία πίνακα (*newTable()*)
- Σε κάθε κελί αποθηκεύεται κείμενο ή εικόνα (insertPicture())



# Παράδειγμα 1: Το ολικό διάγραμμα



# Παράδειγμα 2: Αττικό Μετρό

Θέλουμε να φτιάξουμε ένα ΠΣ για την εταιρεία «Αττικό Μετρό». Σκοπός του συστήματος είναι ο έλεγχος και η επίβλεψη της λειτουργίας και κυκλοφορίας του μετρό. Σχεδιάστε ένα διάγραμμα κλάσεων το οποίο να μοντελοποιεί:

- Το δίκτυο των σταθμών:

  Το δίκτυο αποτελείται από γραμμές. Αυτή τη στιγμή το δίκτυο έχει τρεις γραμμές: την κόκκινη, την μπλε και την πράσινη. Κάθε γραμμή αποτελείται από ένα σύνολο σταθμών (κανονικών, μετεπιβίβασης ή τερματικών). Μας ενδιαφέρει η διασύνδεση των σταθμών και οι χιλιομετρικές αποστάσεις που τους χωρίζουν.

- Τους συρμούς:

  Ένας συρμός αποτελείται από μια τουλάχιστον μηχανή και ένα σύνολο από επιβατικά βαγόνια. Μας ενδιαφέρουν επίσης τα τεχνικά στοιχεία του κάθε τμήματος του συρμού (πχ. ιπποδύναμη, χωρητικότητα, βάρος κλπ)

# Παράδειγμα 2: Αττικό Μετρό

- Τα στοιχεία των εργαζομένων:

  Μας ενδιαφέρει η ειδικότητα των εργαζομένων (οδηγοί, μηχανικοί συρμών/ δικτύου, ελεγκτές κυκλοφορίας, ταμίες) και τα συναφή τους στοιχεία: π.χ ποιος οδηγεί τον συρμό Νο3.

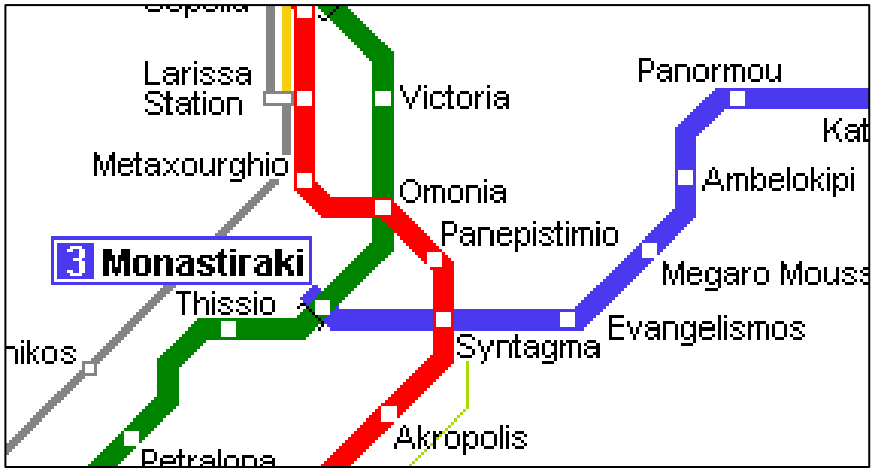
- Τα δρομολόγια

  Κάθε δρομολόγιο έχει αφετηρία, προορισμό και εκτελείται από έναν συγκεκριμένο συρμό.

# Παράδειγμα 2: Δίκτυο Σταθμών

- Καλά μοντέλα είναι εκείνα που σχετίζονται με την πραγματικότητα. Το πρώτο πράγμα που πρέπει να κάνετε είναι να βρείτε τον χάρτη του μετρό (π.χ από τον Ιστό). Ένα απόσπασμα του χάρτη παρατίθεται.

# Παράδειγμα 2: Δίκτυο Σταθμών

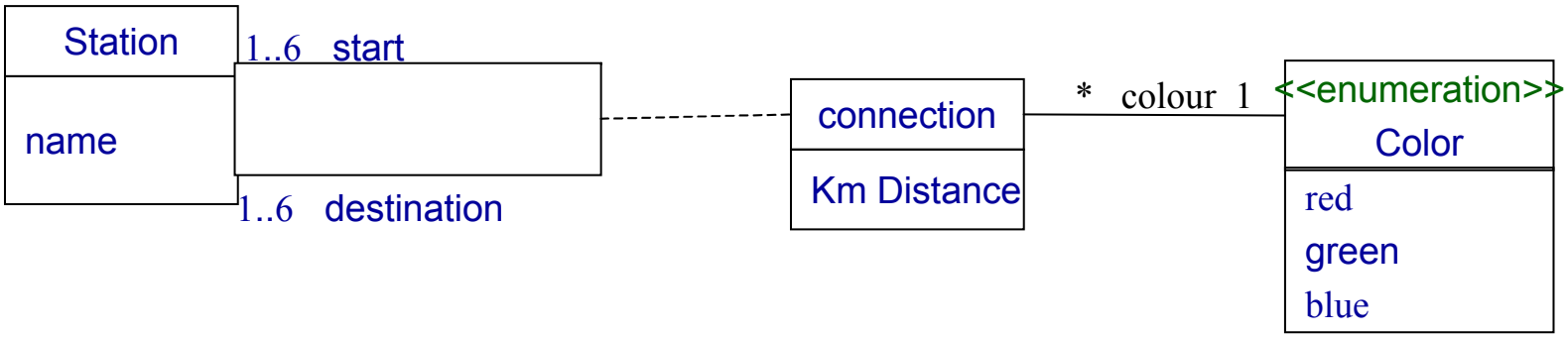



Από τον χάρτη αυτό παρατηρούμε ότι κάποιοι σταθμοί ανήκουν σε δύο γραμμές (π.χ. η Ομόνοια ανήκει στην πράσινη και στην κόκκινη, ενώ το Σύνταγμα στην μπλέ και στην κόκκινη). Άρα δεν πρέπει να διαμερίσουμε τους σταθμούς σε γραμμές αλλά τις διασυνδέσεις των σταθμών.

Εκ τούτου ένας σταθμός είναι μετεπιβίβασης αν είναι άκρο τουλάχιστον δύο συνδέσεων διαφορετικού χρώματος. Ένας σταθμός είναι τερματικός μίας γραμμής αν είναι άκρο μίας μόνο σύνδεσης αυτής της γραμμής. Αν δεν ισχύει τίποτα από τα παραπάνω δύο, τότε μπορούμε να θεωρήσουμε τον σταθμό κανονικό. Ένας σταθμός μπορεί ταυτόχρονα να είναι και τερματικός και μετεπιβίβασης. Για παράδειγμα το Μοναστηράκι είναι τερματικός σταθμός (της μπλε γραμμής) αλλά συνάμα μετεπιβίβασης αφού η πράσινη γραμμή διέρχεται από αυτόν.

# Παράδειγμα 2: Δίκτυο Σταθμών

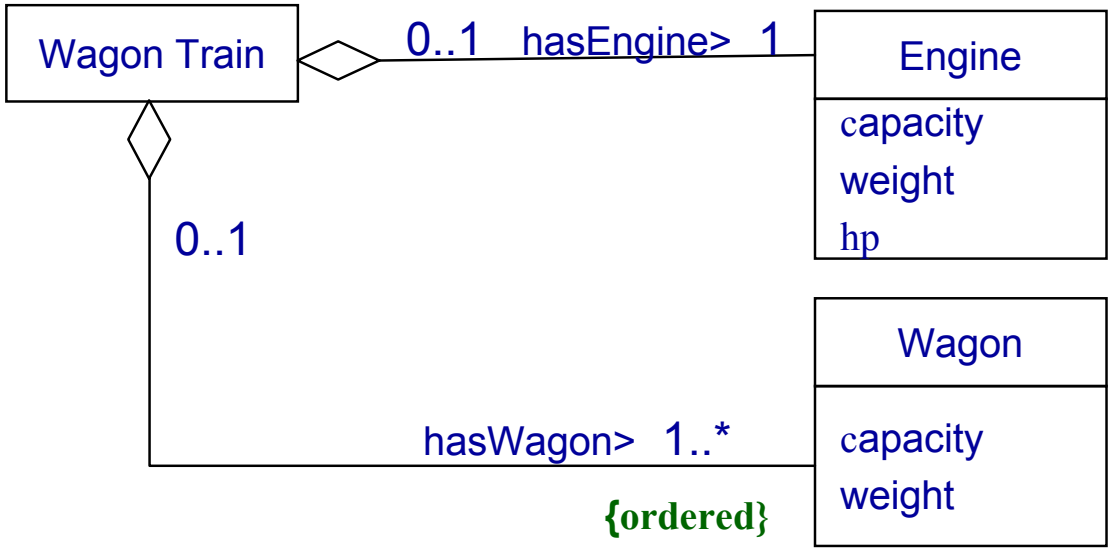
- Ένας τρόπος μοντελοποίησης του δικτύου των σταθμών φαίνεται παρακάτω. Ο περιορισμός των κλάσεων συσχέτισης (ένα ζεύγος σταθμών μπορεί να μετέχει το πολύ μια φορά) μας εξυπηρετεί αφού θα ήταν παράλογο να συνδέονται δυο σταθμοί με παραπάνω από μία (άμεση) διασύνδεση.




- Παρατηρείστε την πολλαπλότητα 1..6. Προφανώς το 0 θα ήταν παράλογο. Το 1 συνεπάγεται σίγουρα τερματικό σταθμό (που δεν είναι μετεπιβίβασης) ενώ το μέγιστο 6 μπορεί να υπάρξει μόνο στην περίπτωση ενός σταθμού από τον οποίο διέρχονται και οι 3 γραμμές και άρα έχει 2 γείτονες ανά γραμμή.

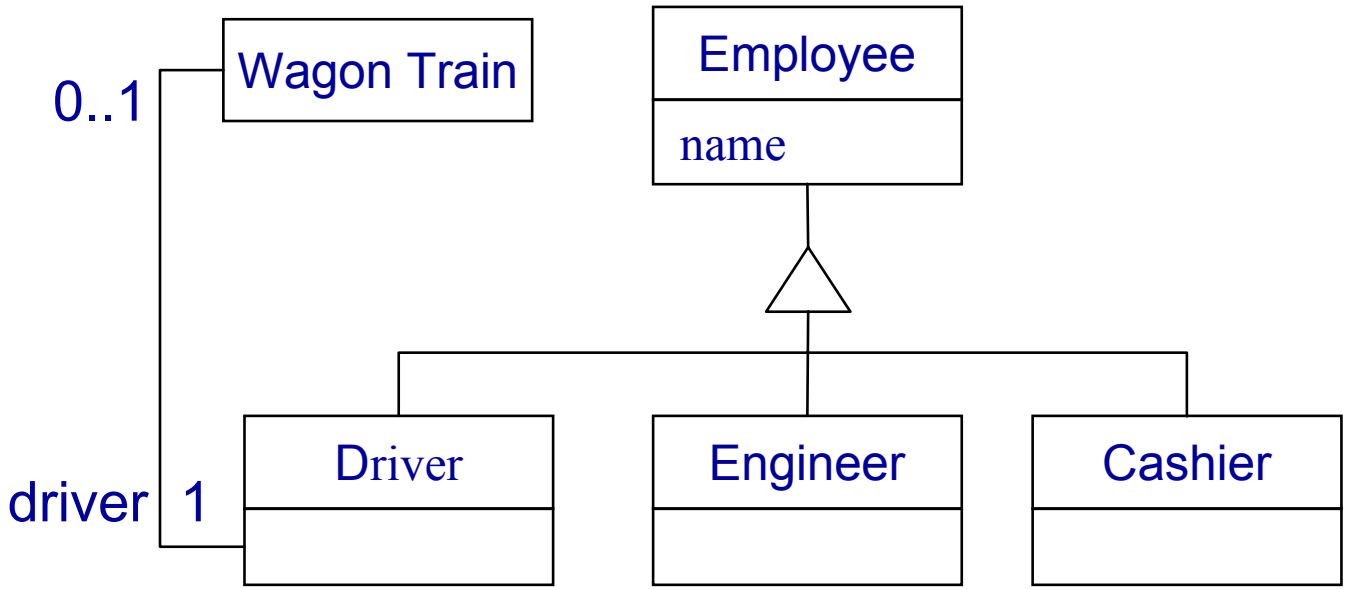
# Παράδειγμα 2: Συρμοί

- Ένας συρμός αποτελείται από μια τουλάχιστον μηχανή και ένα σύνολο από επιβατικά βαγόνια. Μας ενδιαφέρουν επίσης τα τεχνικά στοιχεία του κάθε τμήματος του συρμού (πχ. ιπποδύναμη, χωρητικότητα, βάρος κλπ)
- Μια μοντελοποίηση των συρμών είναι η εξής. Η χρήση του {ordered} μας επιτρέπει να γνωρίζουμε τη σειρά διασύνδεσης των βαγονιών.



# Παράδειγμα 2: Στοιχεία εργαζομένων

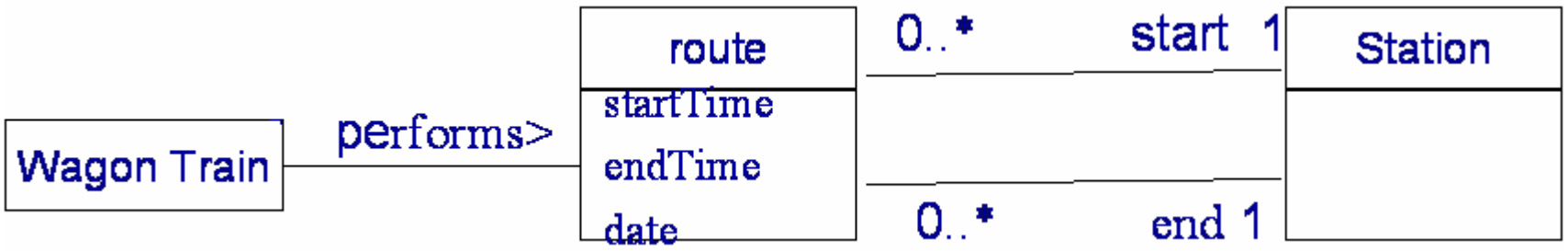
- Μας ενδιαφέρει η ειδικότητα των εργαζομένων (οδηγοί, μηχανικοί συρμών/ δικτύου, ελεγκτές κυκλοφορίας, ταμίες). Για τους οδηγούς μας ενδιαφέρει να καταγράψουμε και ποιο συρμό οδηγούν.
- Μία μοντελοποίηση:



# Παράδειγμα 2: Δρομολόγια

- Κάθε δρομολόγιο έχει αφετηρία, προορισμό και εκτελείται από έναν συγκεκριμένο  συρμό.

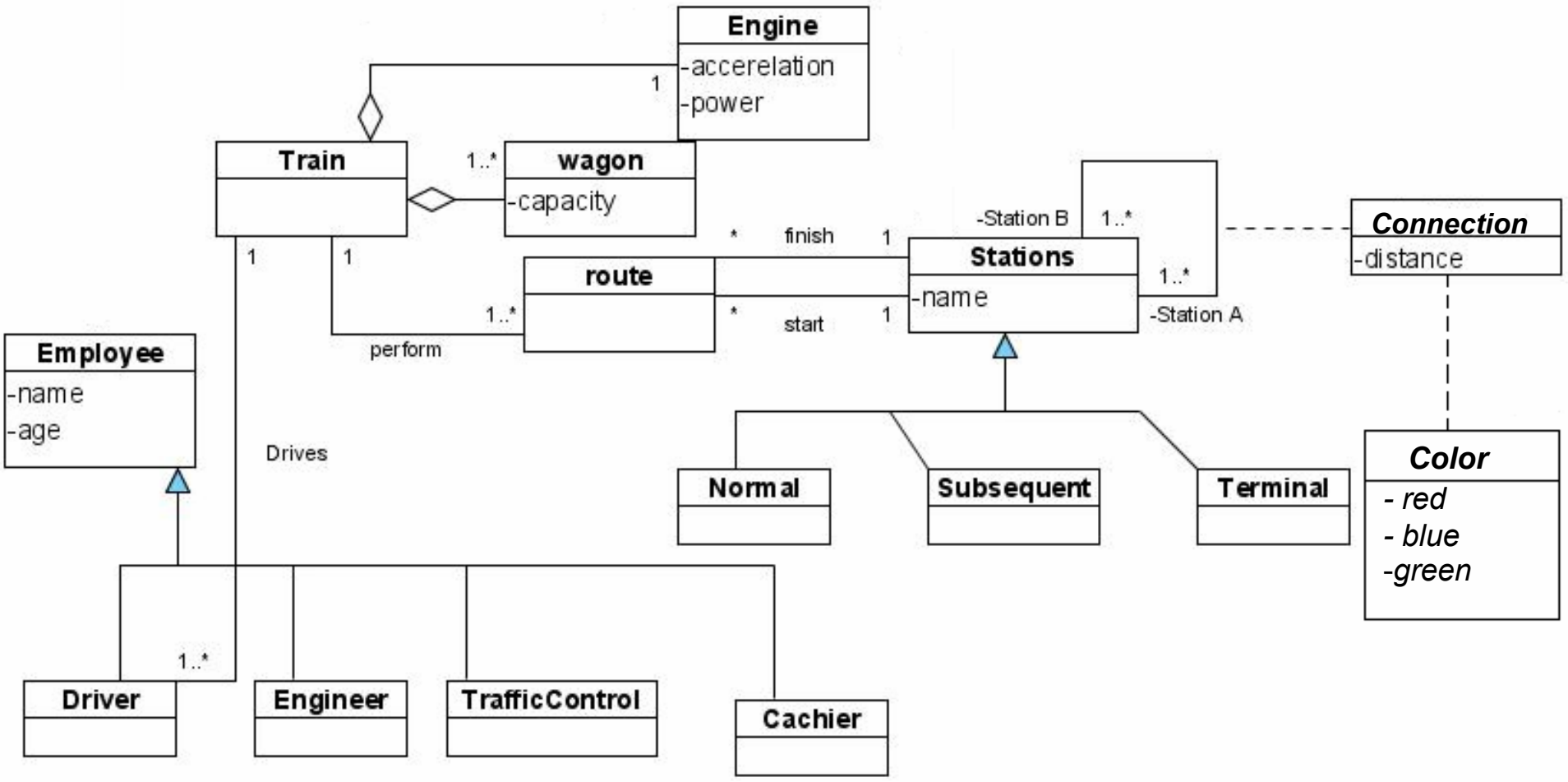
- Η αφετηρία και ο προορισμός ενός δρομολογίου πρέπει να είναι σταθμοί της ίδιας γραμμής. Ο τελευταίος περιορισμός δεν εκφράζεται στο παραπάνω διάγραμμα.



# Παράδειγμα 2: Το πλήρες μοντέλο

- Το πλήρες μοντέλο προκύπτει συναρμολογώντας όλα τα παραπάνω.
- Παρατήρηση: Είναι προτιμότερο ο Οδηγός να συσχετιστεί με το δρομολόγιο και όχι με το τραίνο. Θα μπορούσαμε να ορίσουμε μια παράγωγη συσχέτιση (derived association) μεταξύ οδηγού και τραίνου.

# Παράδειγμα 2: Το πλήρες μοντέλο



# Παράδειγμα 3: Βιβλιοθήκης

- Σε μία βιβλιοθήκη, μπορεί να υπάρχουν διαφορετικά αντίγραφα του ίδιου βιβλίου
- Οι δανειζόμενοι μπορούν κάθε στιγμή να έχουν δανειστεί μέχρι και οχτώ βιβλία
- Τα δανειζόμενα βιβλία επιστρέφονται το πολύ σε τρεις εβδομάδες μετά το δανεισμό, εκτός αν ανανεωθεί ο δανεισμός τους και η ημερομηνία επιστροφής επεκτείνεται για άλλες τρεις εβδομάδες.
- Τα βιβλία αναγνωρίζονται από τον αριθμό τους, δηλαδή το ISBN; ο τίτλος, ο συγγραφέας και η κατηγορία είναι επίσης σημαντικά.
- Κάθε αντίγραφο του βιβλίου έχει ένα ξεχωριστό αριθμό καταλόγου, και η βιβλιοθήκη καταγράφει την ημερομηνία αγοράς και το κόστος όταν η νέα αγορά καταχωρείται.

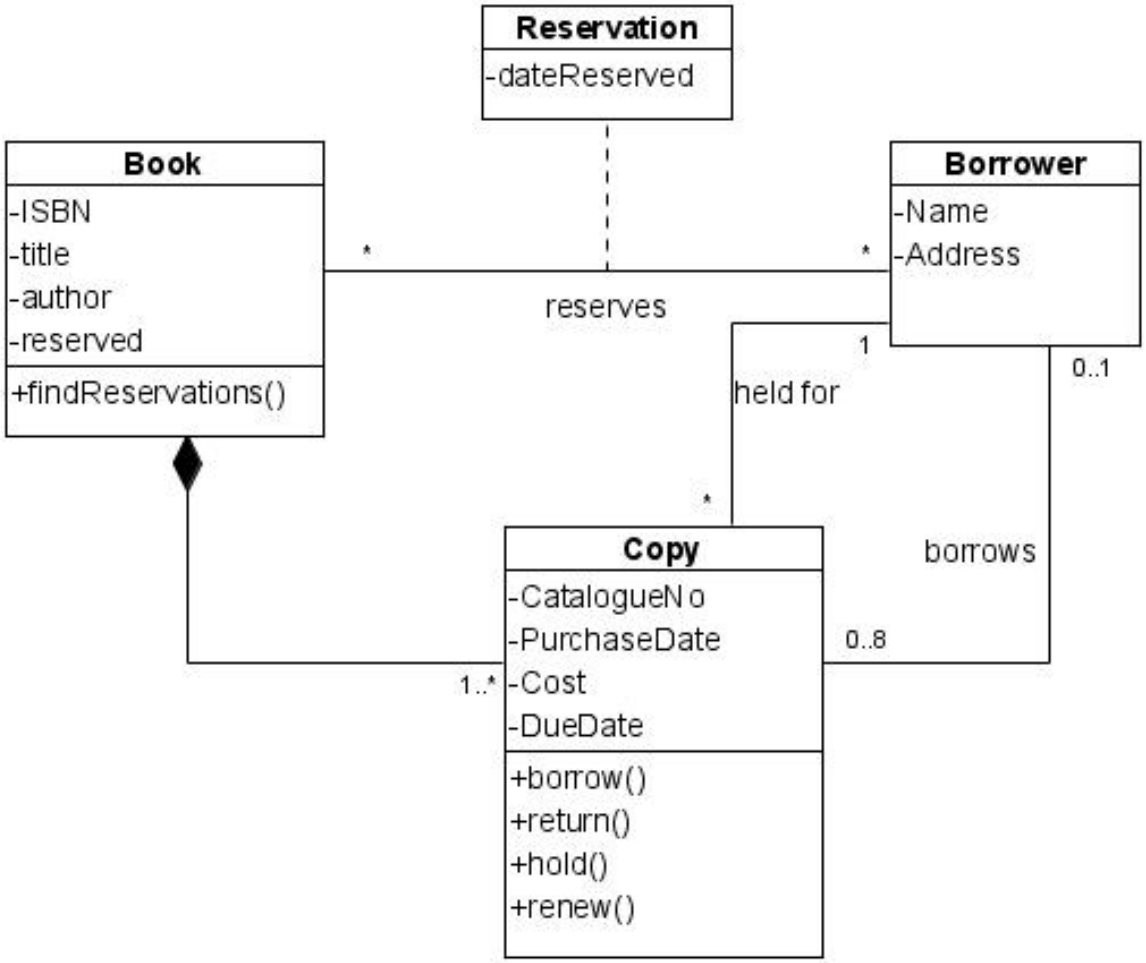
# Παράδειγμα 3: Βιβλιοθήκης

- Οι δανειζόμενοι μπορούν να κάνουν κράτηση σε βιβλία, οπότε σε αυτή τη περίπτωση το πρώτο αντίγραφο του τίτλου που θα επιστραφεί θα κρατηθεί για αυτούς, εκτός αν κάποιος άλλος έχει κάνει κράτηση στο βιβλίο πρώτος.
- Σχεδιάστε ένα διάγραμμα κλάσεων, περιλαμβάνοντας οτιδήποτε γνωρίσματα στο σενάριο, καθώς και τις εξής λειτουργίες : δανεισμός, κράτηση, επιστροφή, ανανέωση βιβλίου, καθώς και εύρεση όλων των κρατήσεων που έχουν γίνει για ένα συγκεκριμένο βιβλίο.

# Παράδειγμα 3: Γενικά Σχόλια

- Σχόλιο από διδασκαλία μαθήματος 2005
  - Αρκετοί φοιτητές επέλεξαν να μοντελοποιήσουν τις λειτουργίες όχι ως operations των κλάσεων, αλλά ως ξεχωριστές κλάσεις που κάθε τους instance αντιστοιχεί σε μία συγκεκριμένη λειτουργία που γίνεται μεταξύ δανειζόμενου και βιβλίου. Μια προτιμότερη μοντελοποίηση είναι η ακόλουθη:

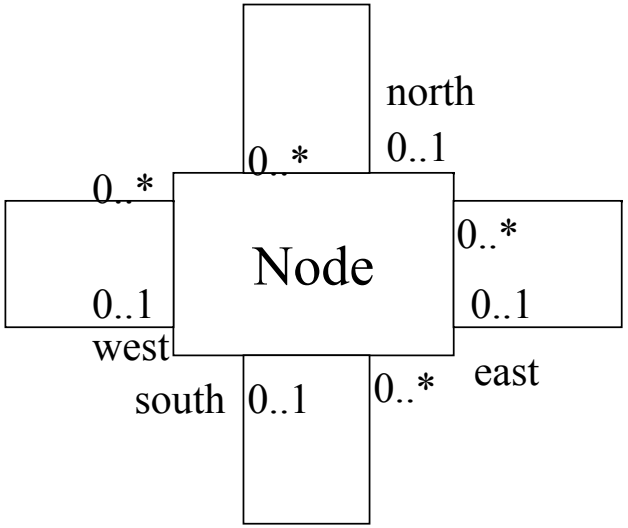
# Παράδειγμα 3: Μια μοντελοποίηση



# Παράδειγμα 4: Adventure Games Graph

- Τα adventure games συχνά βασίζονται σε γράφους. Οι κόμβοι του γράφου παριστάνουν θέσεις, ενώ υπάρχει και ένα σύνολο κατευθύνσεων (πχ Βοράς, Νότος, Ανατολή, Δύση). Για κάθε κόμβο n και κατεύθυνση d, υπάρχει το πολύ ένας κόμβος στον οποίο μπορούμε να φθάσουμε αν ακολουθήσουμε την κατεύθυνση d από τον n
- (α) Σχεδιάστε ένα διάγραμμα κλάσεων για την καταχώρηση τέτοιων γράφων. Σημειώστε επίσης και τους περιορισμούς αντιστοίχισης (multiplicity constraints) σε όλες τις συσχετίσεις που θα ορίσετε.

# Παράδειγμα 4: Μια λύση

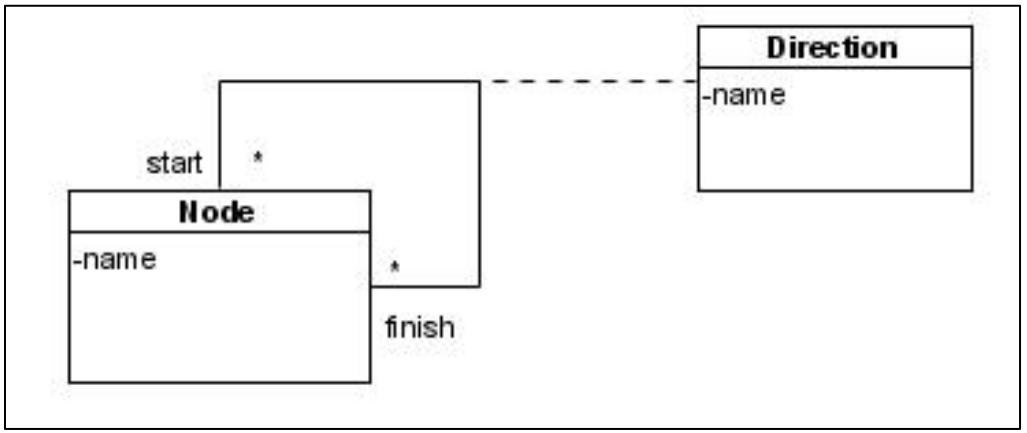


# Παράδειγμα 4: Παραλλαγή Εκφώνησης

- Σχεδιάστε μια άλλη έκδοση του σχήματος αυτού που να επιτρέπει την καταχώριση γράφων με απεριόριστο πλήθος δυνατών κατευθύνσεων. Σχολιάστε σύντομα τις σχεδιαστικές επιλογές που κάνατε σε κάθε περίπτωση.

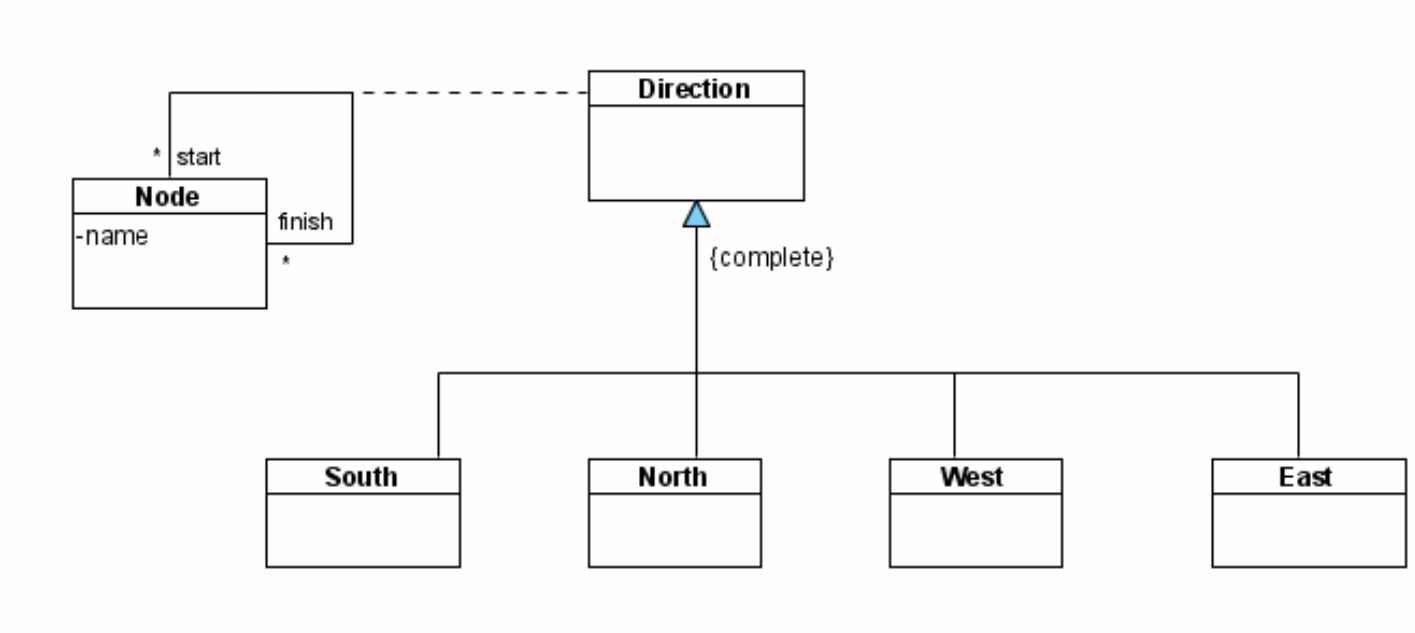
# Παράδειγμα 4: Λύση (A)

- Το διπλανό διάγραμμα επιτρέπει απεριόριστο αριθμό κατευθύνσεων. Παρά ταύτα, η κλάση συσχέτισης θέτει έναν επιπλέον περιορισμό: μεταξύ δύο κόμβων (στιγμιότυπων της κλάσης Node) μπορεί να υπάρχει το πολύ μία συσχέτιση. Άρα το παραπάνω διάγραμμα αποκλείει μεταβάσεις της μορφής (U1.North= U2) και (U1.South=U2).



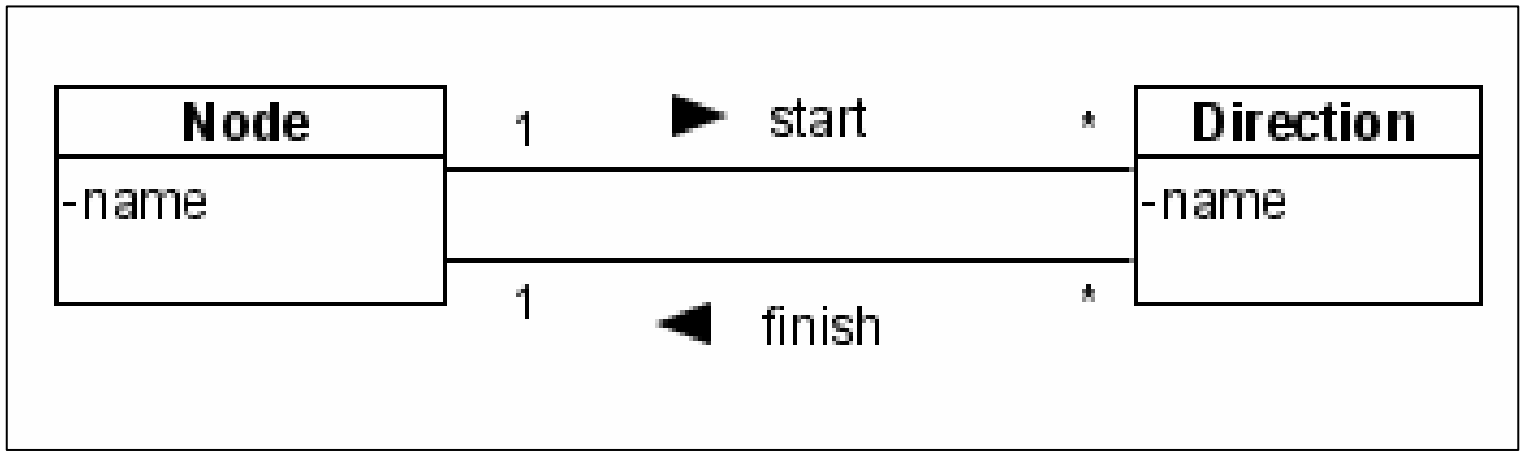
# Παράδειγμα 4: Παραλλαγή λύσης (A)

- Η εκφώνηση της άσκησης δεν έθεσε έναν τέτοιο περιορισμό. Αν είχε τεθεί αυτός ο περιορισμός, τότε το διάγραμμα για το (α) σκέλος της άσκησης θα μπορούσε να έχει την εξής μορφή:



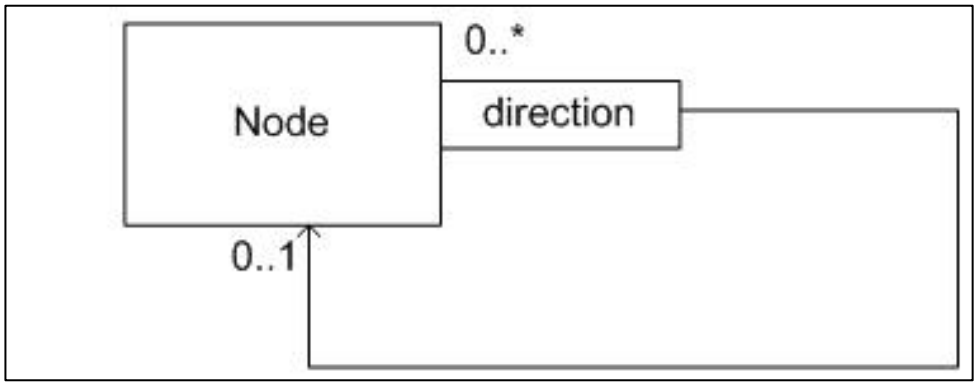
# Παράδειγμα 4: Λύση (B)

- Μια διαφορετική μοντελοποίηση που επιτρέπει απεριόριστο αριθμό κατευθύνσεων αλλά δεν θέτει τον παραπάνω περιορισμό είναι η ακόλουθη. Αυτή όμως η λύση έχει το μειονέκτημα ότι δεν ικανοποιεί τον περιορισμό μοναδικού προορισμού: «Για κάθε κόμβο n και κατεύθυνση d, υπάρχει το πολύ ένας κόμβος στον οποίο μπορούμε να φθάσουμε αν ακολουθήσουμε την κατεύθυνση d από τον n». Με άλλα λόγια επιτρέπει (U1.North=U2) και (U1.North=U3).



# Παράδειγμα 4: Λύση (Γ)

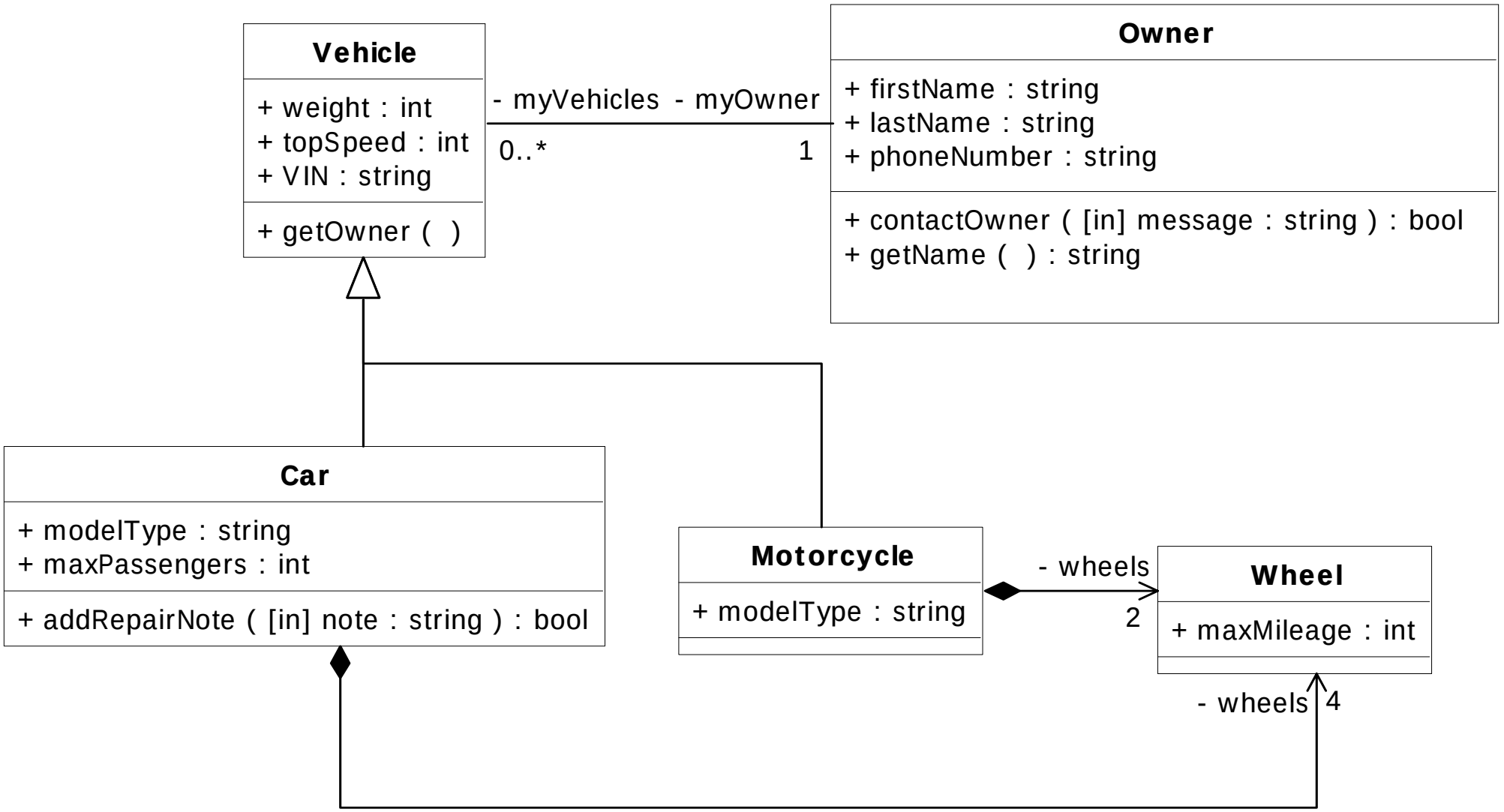
- Μία λύση που δεν έχει τα παραπάνω προβλήματα είναι η εξής:
- Χρησιμοποιούμε το γνώρισμα *direction* ως qualifier της αναδρομικής συσχέτισης και την πολλαπλότητα 0..1 στον προορισμό. Με αυτόν τον τρόπο από ένα κόμβο και για συγκεκριμένη κατεύθυνση μπορούμε να καταλήξουμε σε έναν το πολύ κόμβο. Επίσης η μοντελοποίηση αυτή δεν απαγορεύει μεταβάσεις της μορφής: (U1.North=U2) και (U1. South=U2).




# Παράδειγμα 5 : Μοντελοποίηση Οχημάτων - Ιδιοκτητών

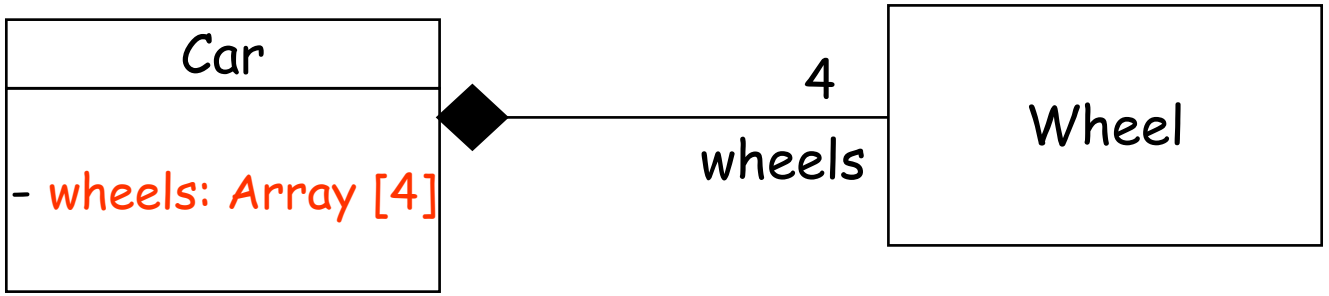
- Ένα όχημα μπορεί να είναι είτε αυτοκίνητο είτε μοτοσικλέτα
- Όλα τα οχήματα έχουν κάποια κοινά χαρακτηριστικά όπως η μέγιστη ταχύτητα και το βάρος
- Το αυτοκίνητο και οι μοτοσικλέτες διαφέρουν σε κάποια χαρακτηριστικά όπως ο αριθμός από ρόδες που διαθέτουν
- Ένα αυτοκίνητο έχει επιπλέον χαρακτηριστικά όπως ο αριθμός των επιβατών που χωράει
- Κάθε όχημα ανήκει σε έναν ιδιοκτήτη ενώ ένας ιδιοκτήτης μπορεί να έχει πολλά οχήματα
- Κάθε ιδιοκτήτης έχει κάποια χαρακτηριστικά όπως όνομα και αριθμό τηλεφώνου

# Παράδειγμα 5 : Μοντελοποίηση Οχημάτων - Ιδιοκτητών

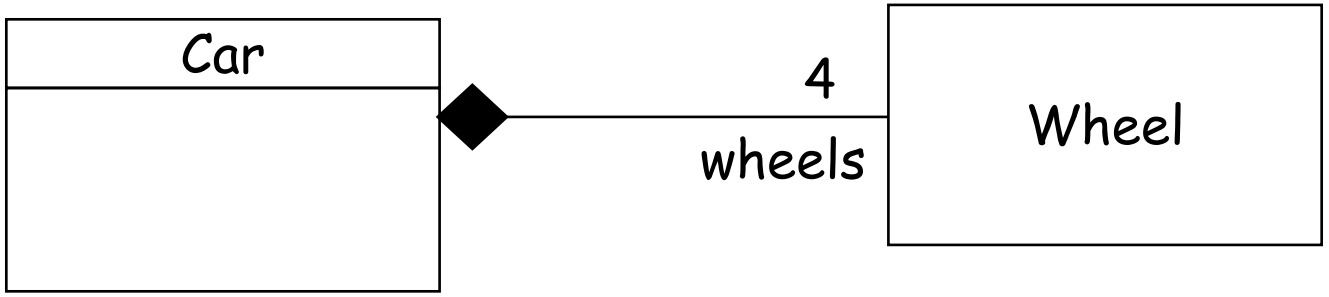


# Συχνά Λάθη

# (1) Επανάληψη Πληροφορίας




: Λάθος

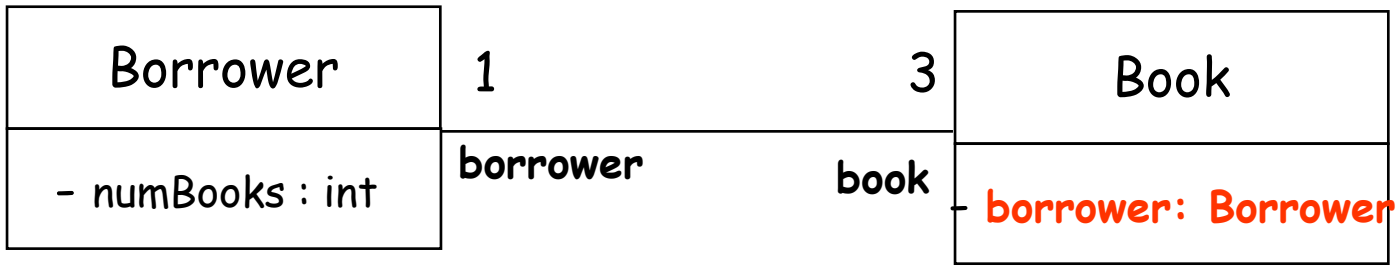



: Σωστό


: Σωστό

*(Αλλά περιγράφει την υλοποίηση)*

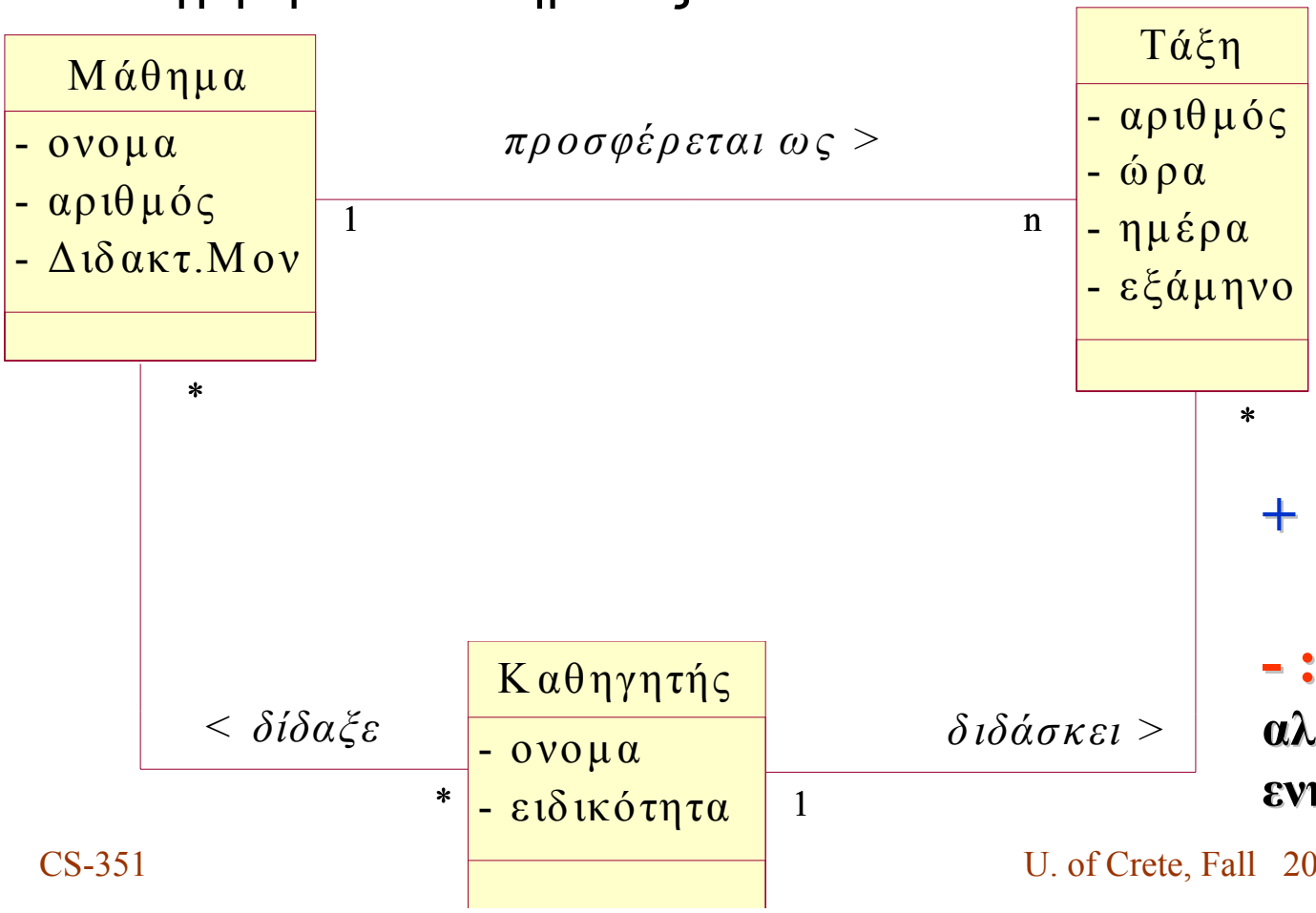
# (2) Επανάληψη Πληροφορίας




: Λάθος

# (3) Πλεονάζουσες Συσχετίσεις

Σε περίπτωση που από την ανάλυση προέκυπτε ότι συχνά απαιτούνταν να γνωρίζουμε ποιοι καθηγητές δίδαξαν ένα μάθημα, θα μπορούσαμε να επιλέξουμε να προσθέσουμε μια πλεονάζουσα συσχέτιση μεταξύ Καθηγητή και Μαθήματος



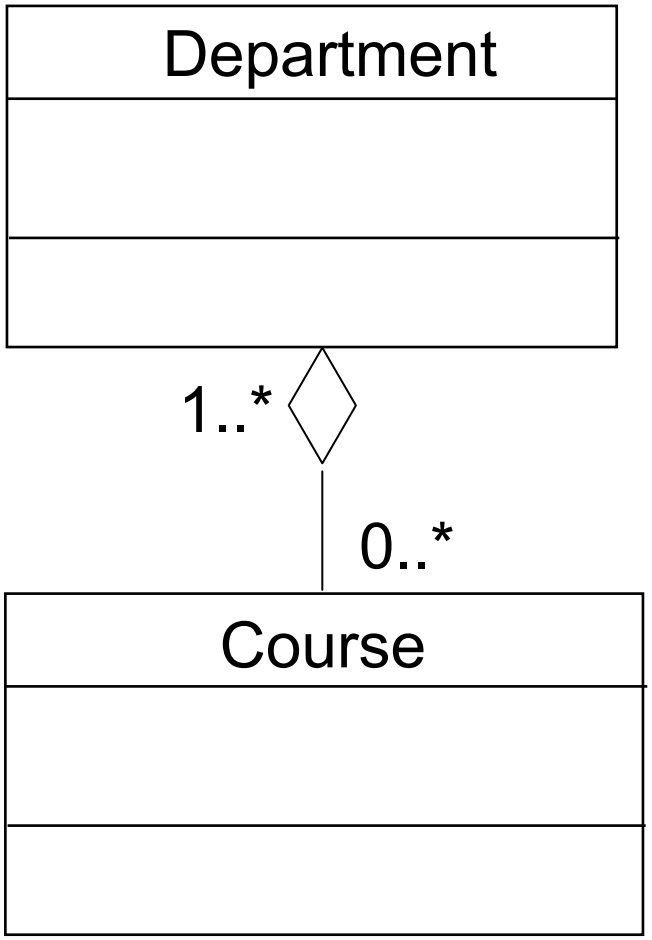

**+ : ταχύτητα**

**- : πολυπλοκότητα (σε κάθε αλλαγή καθηγητή απαιτείται ενημέρωση 2 δεσμών)**

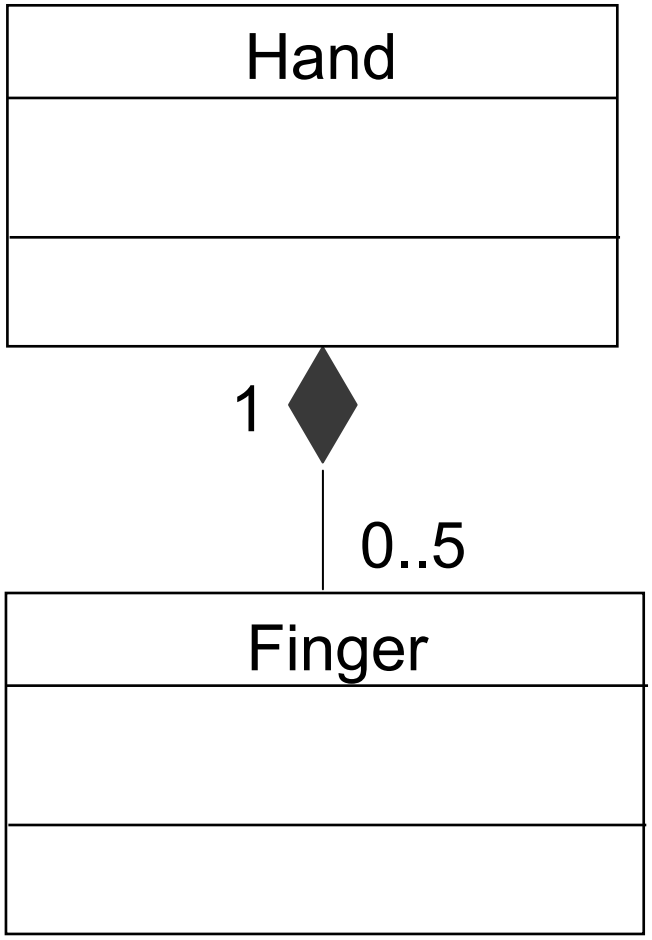
# (4) Συνάθροιση vs Σύνθεση

- *Δηλώνουν σχέσεις του τύπου το **σύνολο** αποτελείται-από **μέρη***
- Συνάθροιση (Aggregation)
  - Το μέρος δημιουργείται και καταστρέφεται ανεξάρτητα από το σύνολο
  - Το μέρος μπορεί να ανήκει σε περισσότερα από ένα σύνολα
- Σύνθεση (Composition)
  - Δυνατότερη σχέση από τη  συνάθροιση
  - Το μέρος δημιουργείται και καταστρέφεται με το σύνολο
  - Το μέρος μπορεί να ανήκει μόνο σε ένα σύνολο
- Οι ορισμοί και η μεταξύ τους διάκριση είναι ακόμα υπό συζήτηση

# Συνάθροιση




# Σύνθεση



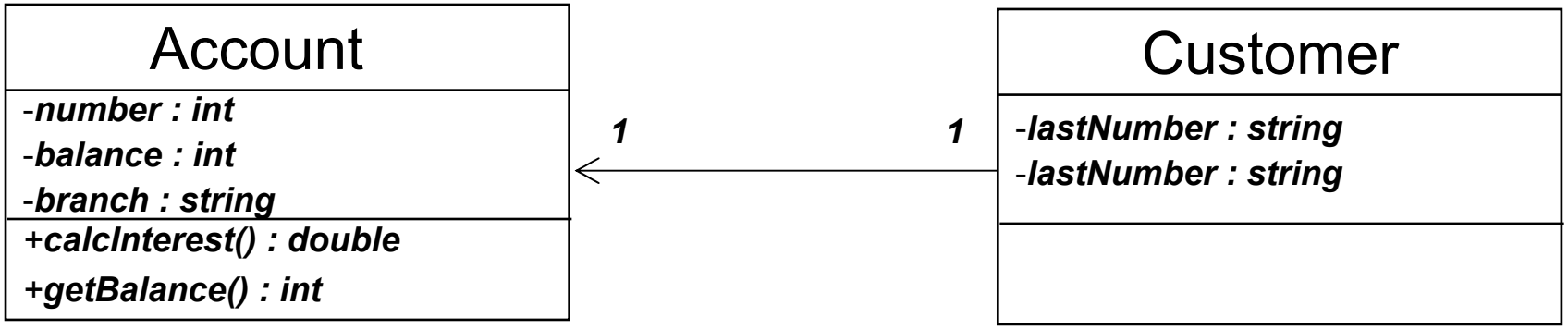
# Υλοποίηση Διαγραμμάτων Κλάσεων και Java

# Κλάσεις

| Account |
| --- |
| -number : int<br>-balance : int<br>-branch : string |
| +calcInterest() : double<br>+getBalance() : int |

```
public class Account
{
      private int number;
      private int balance;
      private String branch;

     public double calcInterest(){
        return 0;
     }

     public getBalance(){
       return 0;
     }
}
```

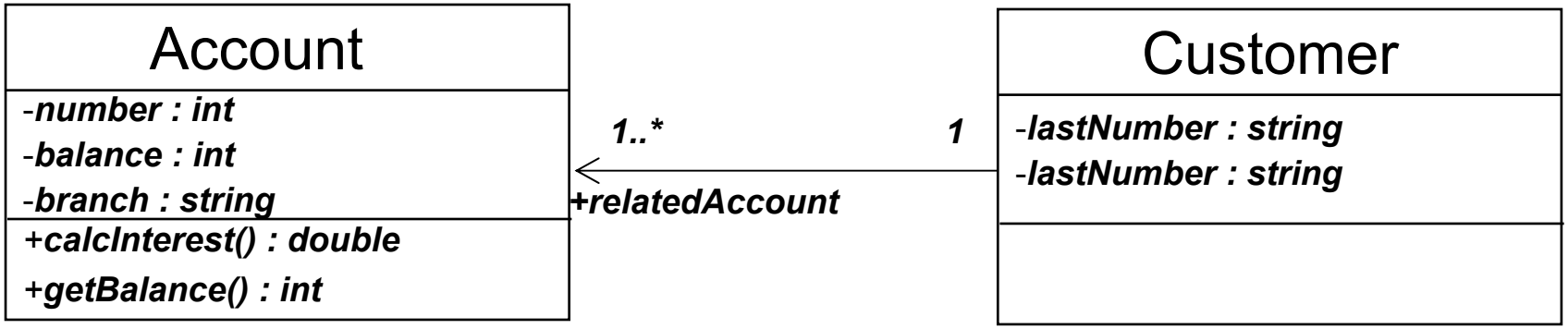
# Συσχετίσεις Απλής Κατεύθυνσης




```
public class Account
{
      private int number;
      private int balance;
      private String branch;

     public double calcInterest(){
         return 0;
     }

     public getBalance(){
        return 0;
     }
}
```

```
public class Customer
{
      private String lastName;
      private String firstName;
      private Account  theAccount;
}
```

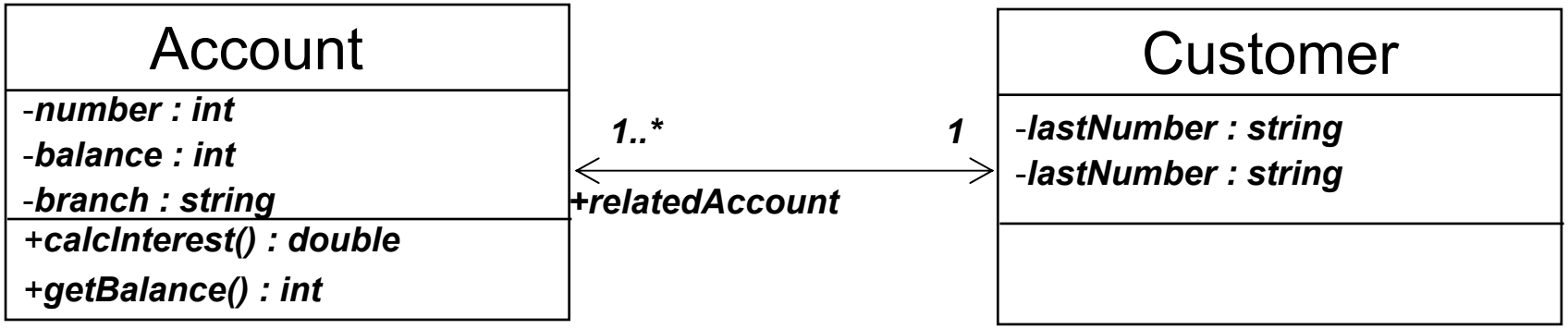
# Συσχετίσεις Απλής Κατεύθυνσης με Πολλαπλότητα




```
public class Account
{
      private int number;
      private int balance;
      private String branch;

      public double calcInterest(){
         return 0;
      }

      public getBalance(){
        return 0;
      }
}
```

```
public class Customer
{
      private String lastName;
      private String firstName;
      private ArrayList  theAccounts;
}
```

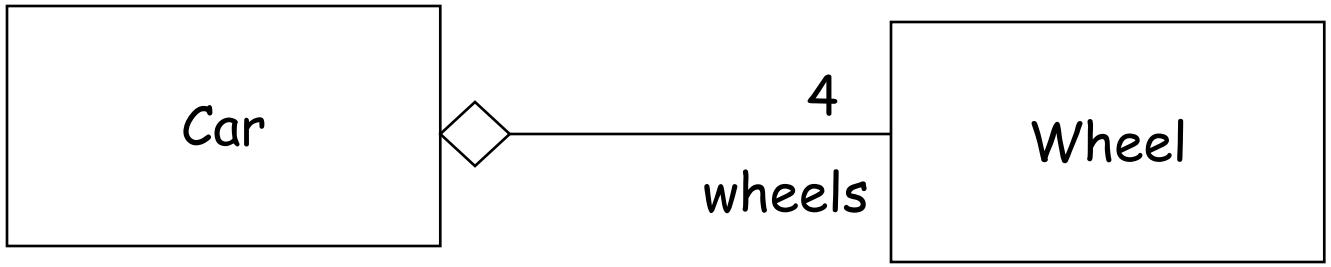
# Συσχετίσεις Διπλής Κατεύθυνσης με Πολλαπλότητα




```
public class Account
{
      private int number;
      private int balance;
      private String branch;
     private Customer theCustomer;

     public double calcInterest(){
        return 0;
     }
     public getBalance(){
        return 0;
    }
}
```

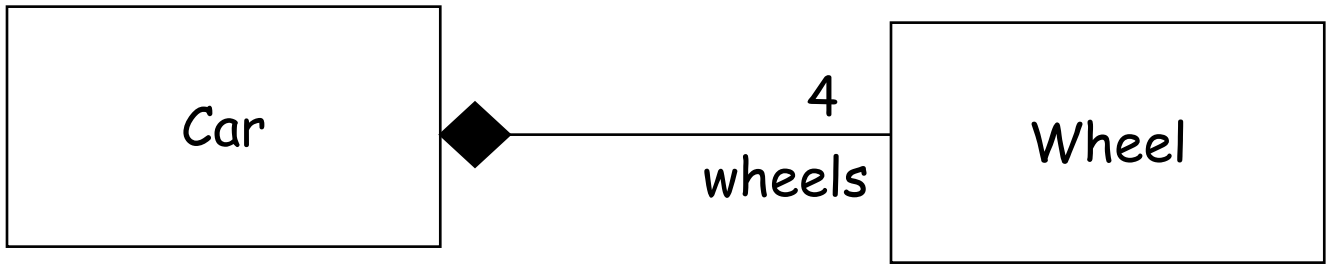
```
public class Customer
{
      private String lastName;
      private String firstName;
      private ArrayList  theAccount;
}
```

# Συναθροίσεις



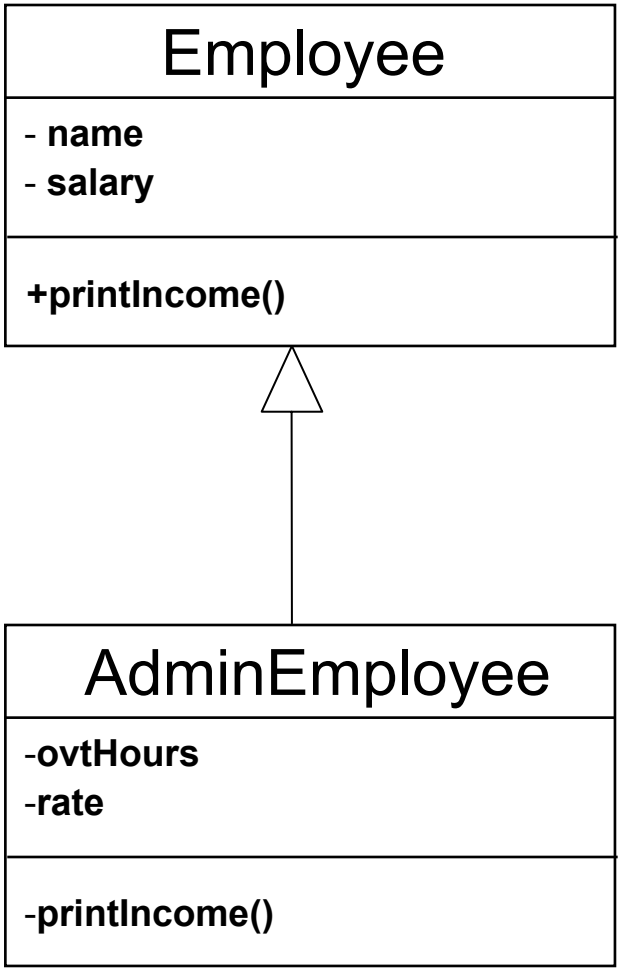

```
public class Car
{    private Wheel wheels[];
     ...
     // τα αντικείμενα wheel δημιουργούνται εξωτερικά
     // και μπαίνουν ορίσματα στον constructor

     public Car( Wheel w1, Wheel w2, … )
     {
          wheels = new Wheel[4];
          wheels[0] = w1;
          wheels[1] = w2;
          …
     }
}
```

# Συνθέσεις




```
public class Car
{
    private Wheel wheels[];
    ...
    public Car()
    {
        wheels = new Wheel[4];
        // τα αντικείμενα wheel δημιουργούνται εσωτερικά στον constructor
        wheels[0] = new Wheel();
        wheels[1] = new Wheel();
        …
    }
    ...
}
```

# Κληρονομικότητα




```
public class Employee {
      private String name;
      private int salary;

      public double printIncome(){
         return 0;
      }
}
```

```
public class AdminEmployee extends Employee{
      private int ovtHours;
      private int rate;

   //Αυτή η συνάρτηση θα κληθεί στη θέση της πατρικής
   //εάν έχει υλοποιηθεί, διαφορετικά καλείται του πατέρα
      public double printIncome(){
         return 0;
      }
}
```