ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Εισαγωγή

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

(Ευχαριστίες στον καθηγητή Βασίλη Χριστοφίδη)

Λίγο Ιστορία

- Οι πρώτες γλώσσες προγραμματισμού δεν ήταν για υπολογιστές
  - Αυτόματη δημιουργία πρωτοτύπων για ραπτομηχανές
  - Μουσικά κουτιά ή ρολά για πιάνο
  - Η αφαιρετική μηχανή του Turing

Γλώσσες προγραμματισμού

- Πρώτη γενιά: Γλώσσες μηχανής

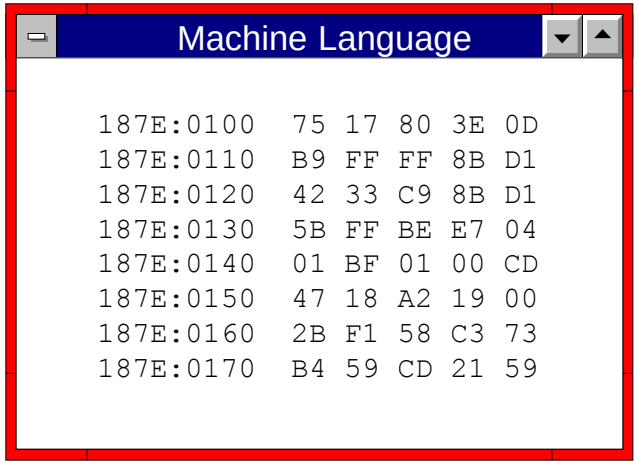
Ο προγραμματιστής μετατρέπει το πρόβλημα του σε ένα πρόγραμμα

- Π.χ. πώς να υπολογίσω το μέγιστο κοινό διαιρέτη δύο αριθμών

Και γράφει ακριβώς τις εντολές που θα πρέπει να εκτελέσει ο υπολογιστής

- Θα πρέπει να ξέρει ακριβώς την δυαδική αναπαράσταση των εντολών.

Στους πρώτους υπολογιστές οι εντολές κωδικοποιούνταν σε διάτρητες κάρτες



Program entered and executed as machine language

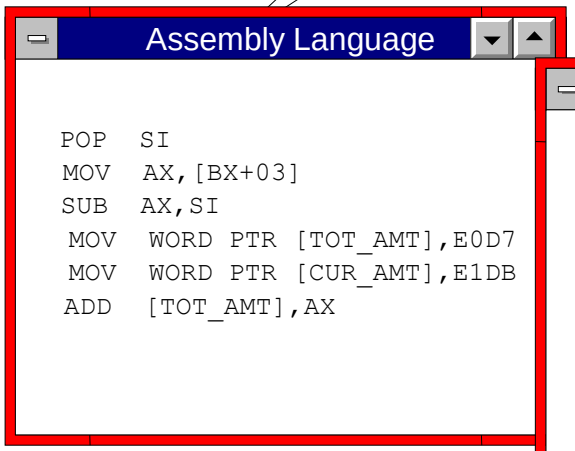
Πέντε γενεές γλωσσών προγραμματισμού

- Πρώτη γενιά: Γλώσσες μηχανής
- Δεύτερη γενιά: Assembly

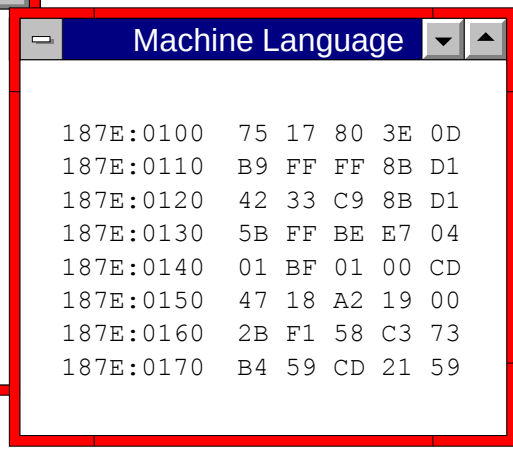
The ASSEMBLER converts instructions to op-codes:
What is the instruction to load from memory?
Where is purchase price stored?
What is the instruction to multiply?
What do I multiply by?
What is the instruction to add from memory?
What is the instruction to store back into memory?

Ο προγραμματιστής δεν χρειάζεται να ξέρει ακριβώς την δυαδική αναπαράσταση των εντολών.

- Χρησιμοποιεί πιο κατανοητούς μνημονικούς κανόνες.
- Ο Assembler μετατρέπει τα σύμβολα σε γλώσσα μηχανής.
- Οι γλώσσες εξαρτώνται από το hardware



Translate into machine operation codes (op-codes)



Program executed as machine language

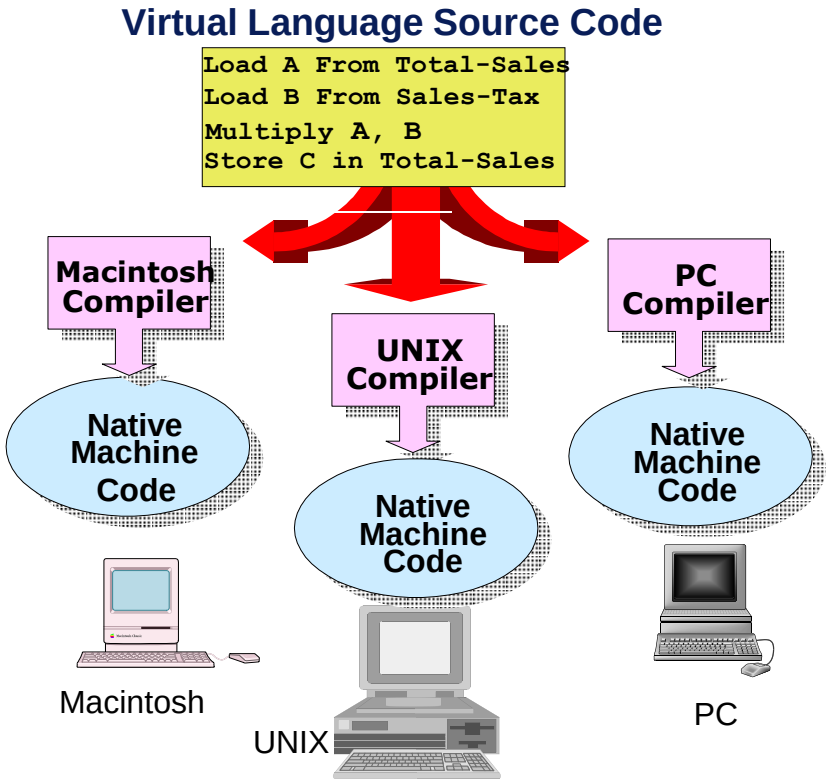
Πέντε γενεές γλωσσών προγραμματισμού

- Πρώτη γενιά: Γλώσσες μηχανής
- Δεύτερη γενιά: Assembly
- Τρίτη γενιά: Υψηλού επιπέδου (high-level) γλώσσες

Ο προγραμματιστής δίνει εντολές στον υπολογιστή σε μια κατανοητή και καλά δομημένη γλώσσα (source code)

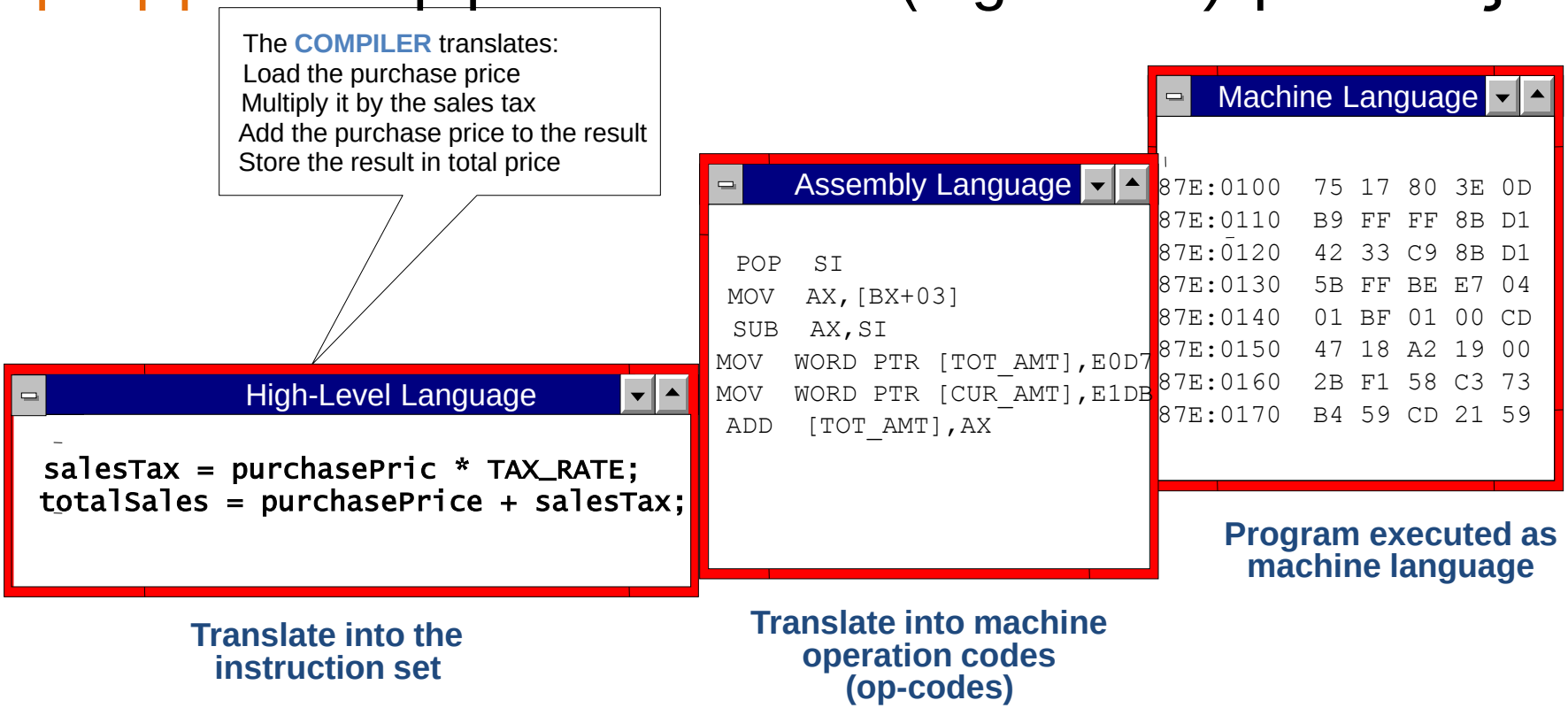
Ο compiler τις μετατρέπει σε ενδιάμεσο κώδικα (object code)

Ο ενδιάμεσος κώδικας μετατρέπεται σε γλώσσα μηχανής (machine code)



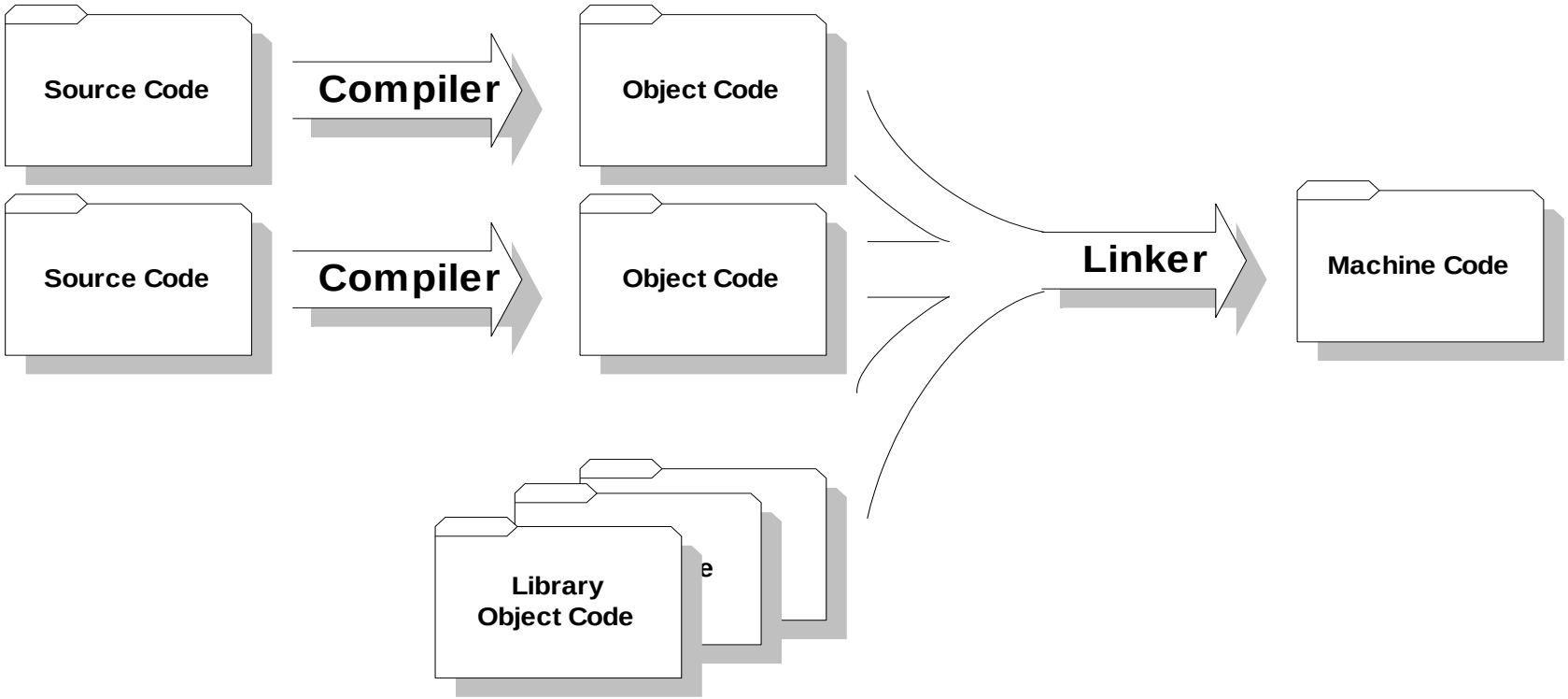
Πέντε γενεές γλωσσών προγραμματισμού

- Πρώτη γενιά: Γλώσσες μηχανής
- Δεύτερη γενιά: Assembly
- Τρίτη γενιά: Υψηλού επιπέδου (high-level) γλώσσες



Πέντε γενεές γλωσσών προγραμματισμού

- Πρώτη γενιά: Γλώσσες μηχανής
- Δεύτερη γενιά: Assembly
- Τρίτη γενιά: Υψηλού επιπέδου (high-level) γλώσσες



Πέντε γενεές γλωσσών προγραμματισμού

- Πρώτη γενιά: Γλώσσες μηχανής
- Δεύτερη γενιά: Assembly
- Τρίτη γενιά: Υψηλού επιπέδου (high-level) γλώσσες
- Τέταρτη γενιά: Εξειδικευμένες γλώσσες
- Πέμπτη γενιά: «Φυσικές» γλώσσες.

- Κάθε γενιά προσθέτει ένα επίπεδο αφαίρεσης.

Προγραμματιστικά Υποδείγματα (paradigms)

- Προγραμματισμός των πρώτων ημερών.

Spaghetti code

Δύσκολο να διαβαστεί και να κατανοηθεί η ροή του

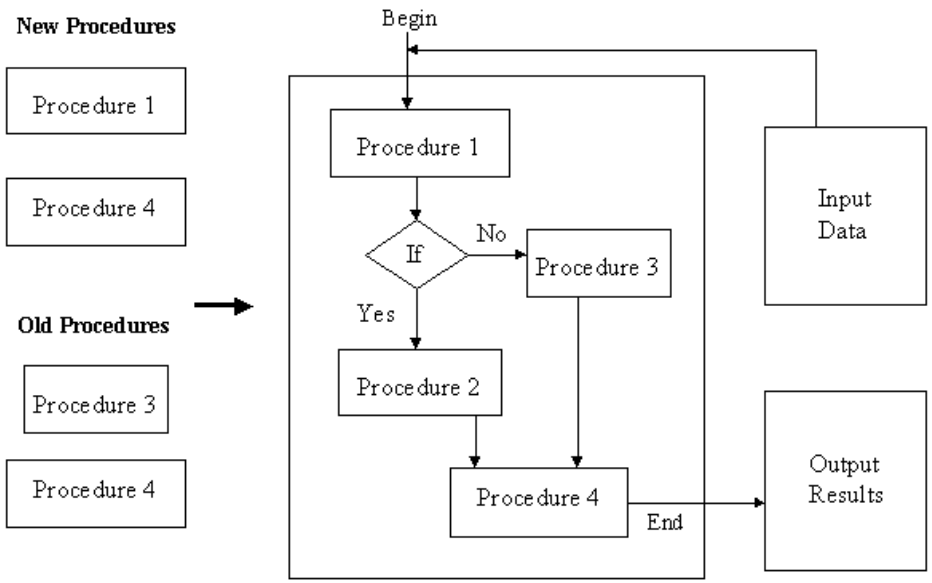
```
100   GOTO 500
110   PRINT I;
120   GOTO 400
150   PRINT I * 12;
160   GOTO 450
200   PRINT " = ";
210   GOTO 150
300   PRINT " 12 ";
310   GOTO 200
400   PRINT " * ";
410   GOTO 300
450   I = I + 1
460   IF I > 12 THEN STOP
460   GOTO 110
500   PRINT "The Twelves Table"
510   I = 1
520   GOTO 110
```

Δομημένος Προγραμματισμός

- Τέσσερεις προγραμματιστικές δομές
  - Sequence – ακολουθιακές εντολές
  - Selection – επιλογή με if-then-else
  - Iteration – δημιουργία βρόγχων
  - Recursion - αναδρομή

- Ο κώδικας σπάει σε λογικά blocks που έχουν ένα σημείο εισόδου και εξόδου.
  - Κατάργηση της GOTO εντολής.

- Οργάνωση του κώδικα σε διαδικασίες (procedures)



Διαδικασιακός Προγραμματισμός

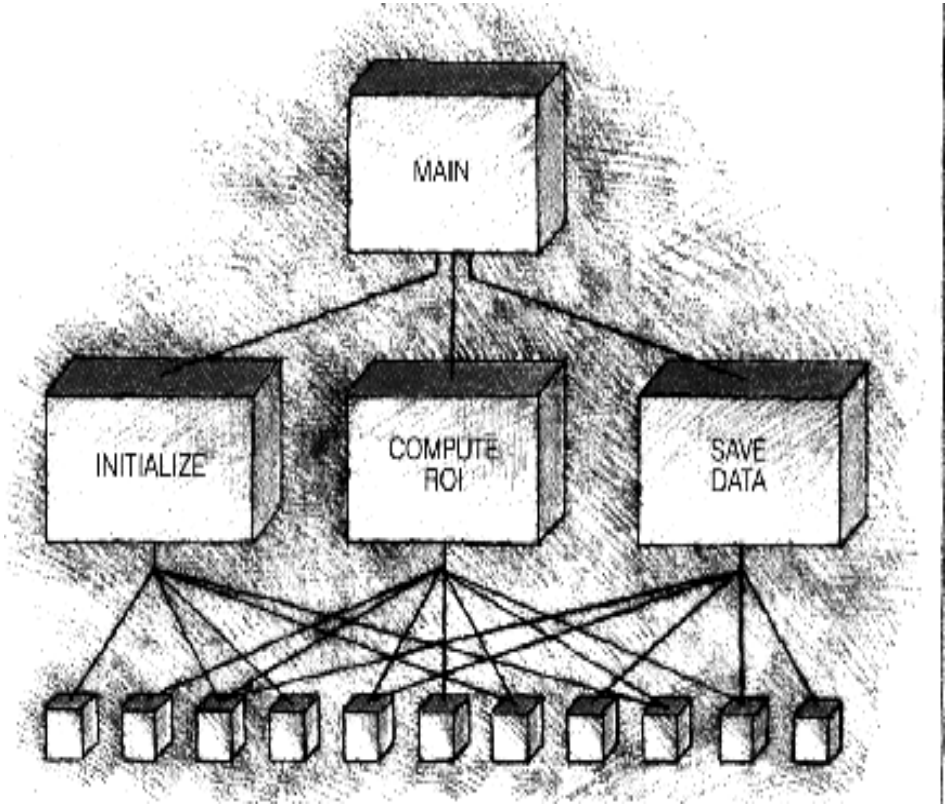
- Το πρόγραμμα μας σπάει σε πολλαπλές διαδικασίες.
  - Κάθε διαδικασία λύνει ένα υπο-πρόβλημα και αποτελεί μια λογική μονάδα (module)
  - Μια διαδικασία μπορούμε να την επαναχρησιμοποιήσουμε σε διαφορετικά δεδομένα.
- Το πρόγραμμα μας είναι τμηματοποιημένο (modular)

Κοινά Δεδομένα

- Ο διαδικασιακός προγραμματισμός τμηματοποιεί τον κώδικα αλλά όχι απαραίτητα τα δεδομένα

- Π.χ., με τη χρήση καθολικών μεταβλητών (global variables) όλες οι διαδικασίες μπορεί να χρησιμοποιούν τα ίδια δεδομένα και άρα να εξαρτώνται μεταξύ τους.

- Πρέπει να αποφεύγουμε τη χρήση καθολικών μεταβλητών!



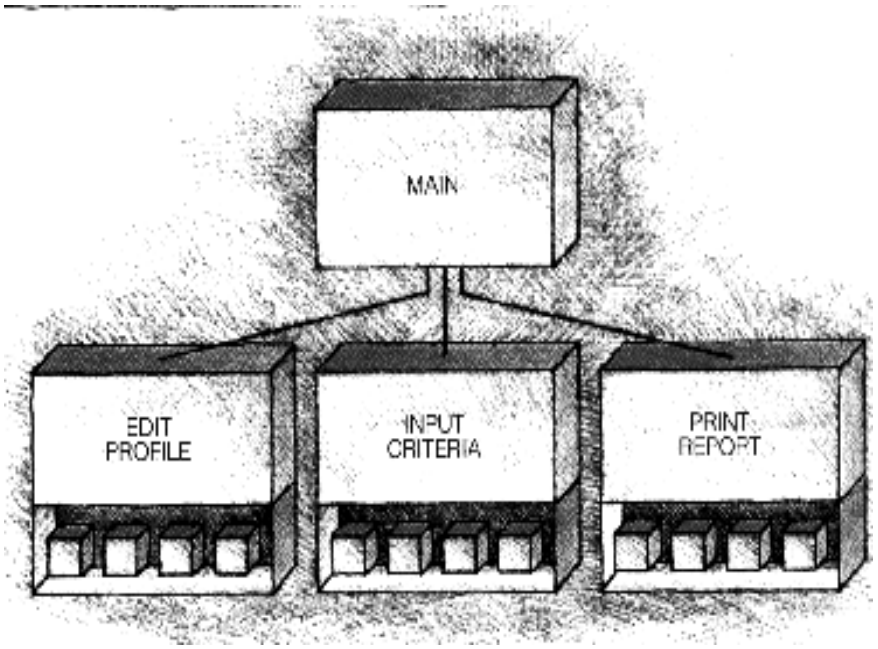
Απόκρυψη δεδομένων

- Με τη δημιουργία τοπικών μεταβλητών μέσα στις διαδικασίες αποφεύγουμε την ύπαρξη κοινών δεδομένων

- Ο κώδικας γίνεται πιο εύκολο να σχεδιαστεί, να γραφτεί και να συντηρηθεί

- Η επικοινωνία μεταξύ των διαδικασιών γίνεται με ορίσματα.

- Τμηματοποιημένος προγραμματισμός (modular programming)

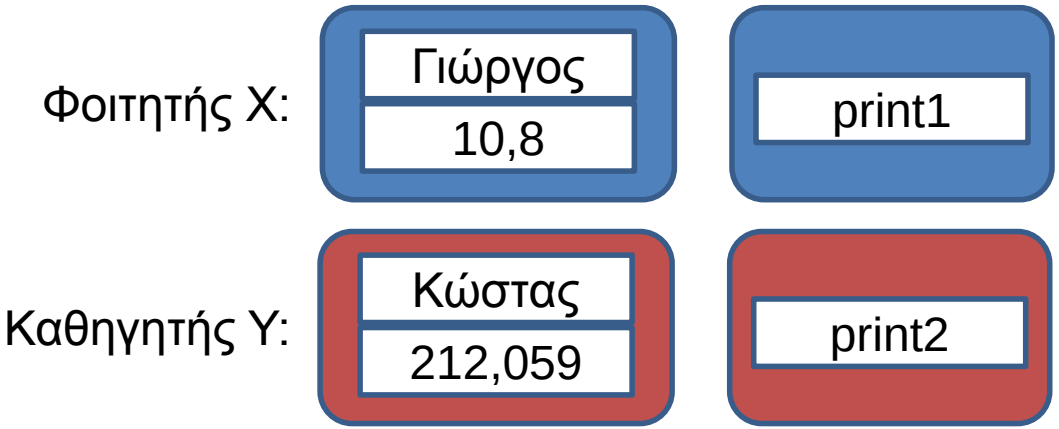


Περιορισμοί του διαδικασιακού προγραμματισμού

- Ο διαδικασιακός προγραμματισμός δουλεύει ΟΚ για μικρά προγράμματα, αλλά για μεγάλα συστήματα είναι δύσκολο να σχεδιάσουμε, να υλοποιήσουμε και να συντηρήσουμε τον κώδικα.
  - Δεν είναι εύκολο να προσαρμοστούμε σε αλλαγές, και δεν μπορούμε να προβλέψουμε όλες τις ανάγκες που θα έχουμε

- Π.χ., το πανεπιστήμιο έχει ένα σύστημα για να κρατάει πληροφορίες για φοιτητές και καθηγητές
  - Υπάρχει μια διαδικασία print που τυπώνει στοιχεία και βαθμούς φοιτητών
  - Προκύπτει ανάγκη για μια διαδικασία που να τυπώνει τα μαθήματα των καθηγητών
    - Χρειαζόμαστε μια print2

Παράδειγμα


Φοιτητής Χ:
Γιώργος
10,8
print1
Καθηγητής Υ:
Κώστας
212,059
print2

Αντικειμενοστραφής προγραμματισμός

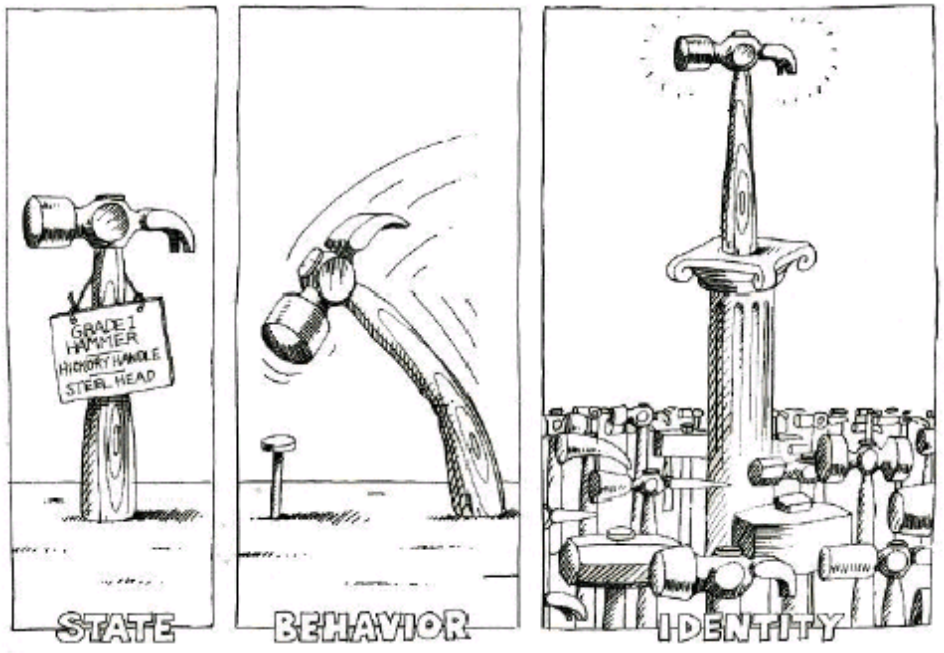
- Τα προβλήματα αυτά προσπαθεί να αντιμετωπίσει ο αντικειμενοστραφής προγραμματισμός (object-oriented programming)
  - Ο αντικειμενοστραφής προγραμματισμός βάζει μαζί τα δεδομένα και τις διαδικασίες (μεθόδους) σχετικές με τα δεδομένα
  - Π.χ., ο φοιτητής ή ο καθηγητής έρχεται με μια δικιά του διαδικασία print

- Αυτό επιτυγχάνεται με αντικείμενα και κλάσεις

Αντικείμενο

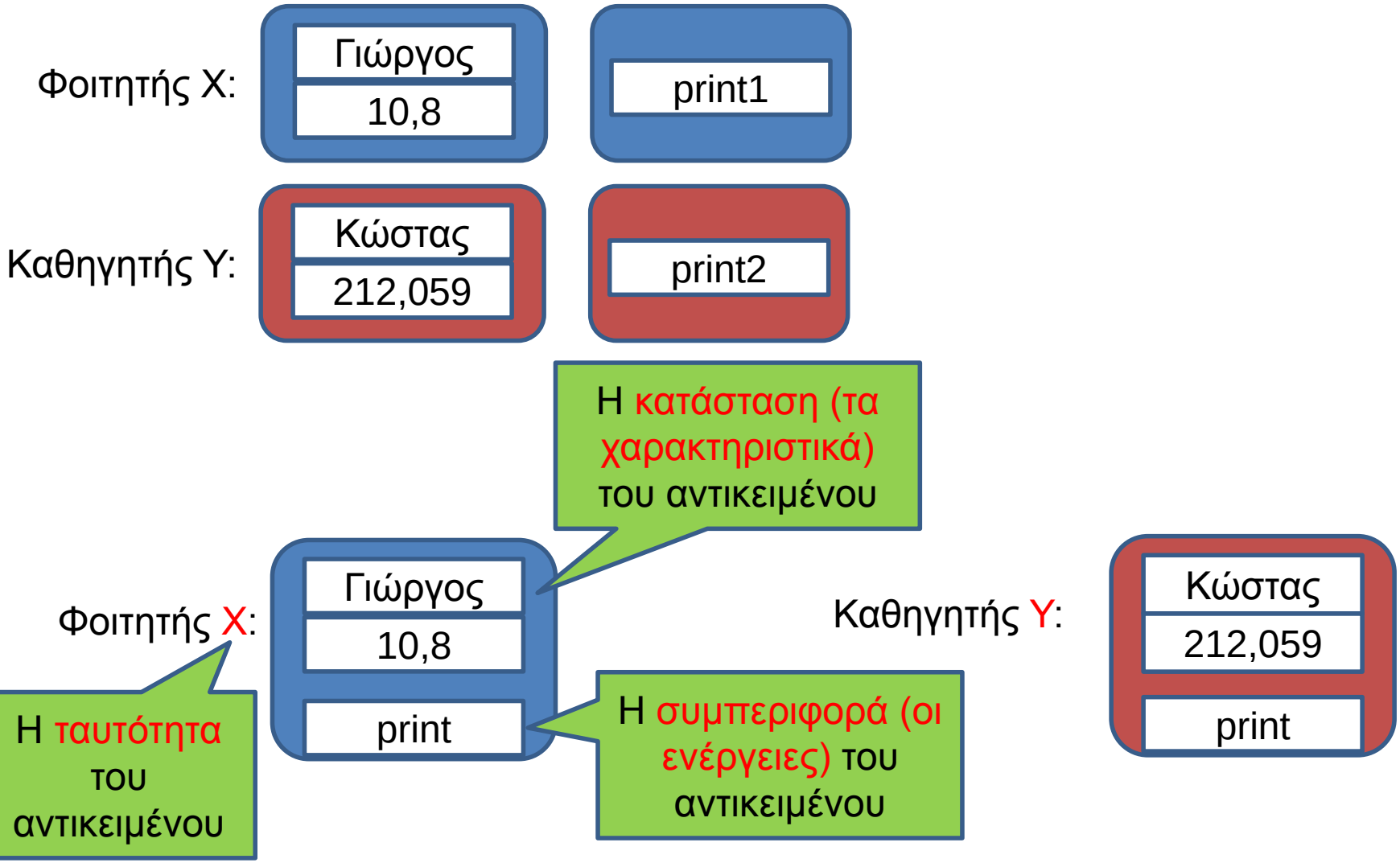
- Ένα αντικείμενο στον κώδικα αναπαριστά μια μονάδα/οντότητα/έννοια η οποία έχει:

- Μια κατάσταση, η οποία ορίζεται από ορισμένα χαρακτηριστικά
- Μια συμπεριφορά, η οποία ορίζεται από ορισμένες ενέργειες που μπορεί να εκτελέσει το αντικείμενο
- Μια ταυτότητα που το ξεχωρίζει από τα υπόλοιπα αντικείμενα ίδιου τύπου.



Παραδείγματα: ένας άνθρωπος, ένα πράγμα, ένα μέρος, μια υπηρεσία

Παράδειγμα


Φοιτητής Χ:
Γιώργος
10,8
print1
Καθηγητής Υ:
Κώστας
212,059
print2
Η κατάσταση (τα χαρακτηριστικά) του αντικειμένου
Φοιτητής Χ:
Γιώργος
10,8
print
Η ταυτότητα του αντικειμένου
Η συμπεριφορά (οι ενέργειες) του αντικειμένου
Καθηγητής Υ:
Κώστας
212,059
print

Κλάσεις

- Κλάση: Μια αφηρημένη περιγραφή αντικειμένων με κοινά χαρακτηριστικά και κοινή συμπεριφορά
  - Ένα καλούπι που παράγει αντικείμενα
  - Ένα αντικείμενο είναι ένα στιγμιότυπο μίας κλάσης.

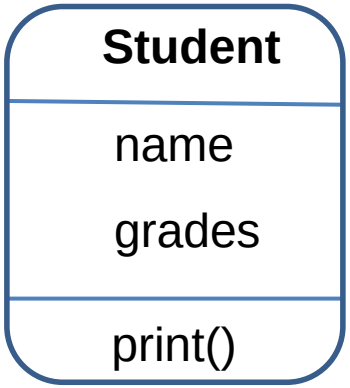
- Π.χ., η κλάση φοιτητής έχει τα γενικά χαρακτηριστικά (όνομα, βαθμοί) και τη συμπεριφορά print
  - Ο φοιτητής Χ είναι ένα αντικείμενο της κλάσης φοιτητής

- Η κλάση Car έχει τα χαρακτηριστικά (brand, color) και τη συμπεριφορά (drive, stop)
  - Το αυτοκίνητο INI2013 είναι ένα αντικείμενο της κλάσης Car με κατάσταση τα χαρακτηριστικά (BMW, red)

Κλάσεις και Αντικείμενα

Κλάση

Μια αφηρημένη περιγραφή ενός φοιτητή.



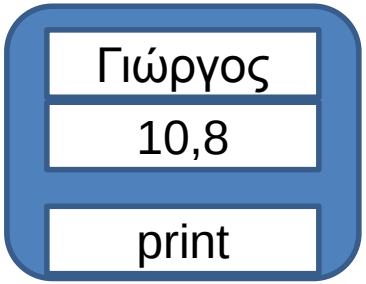
Όνομα κλάσης

Πεδία κλάσης: Ιδιότητες/Χαρακτηριστικά

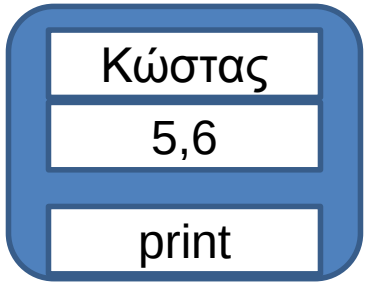
Μέθοδοι κλάσης: λειτουργίες

Αντικείμενα

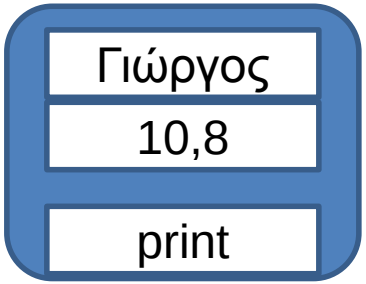
Φοιτητής X:



Φοιτητής Y:



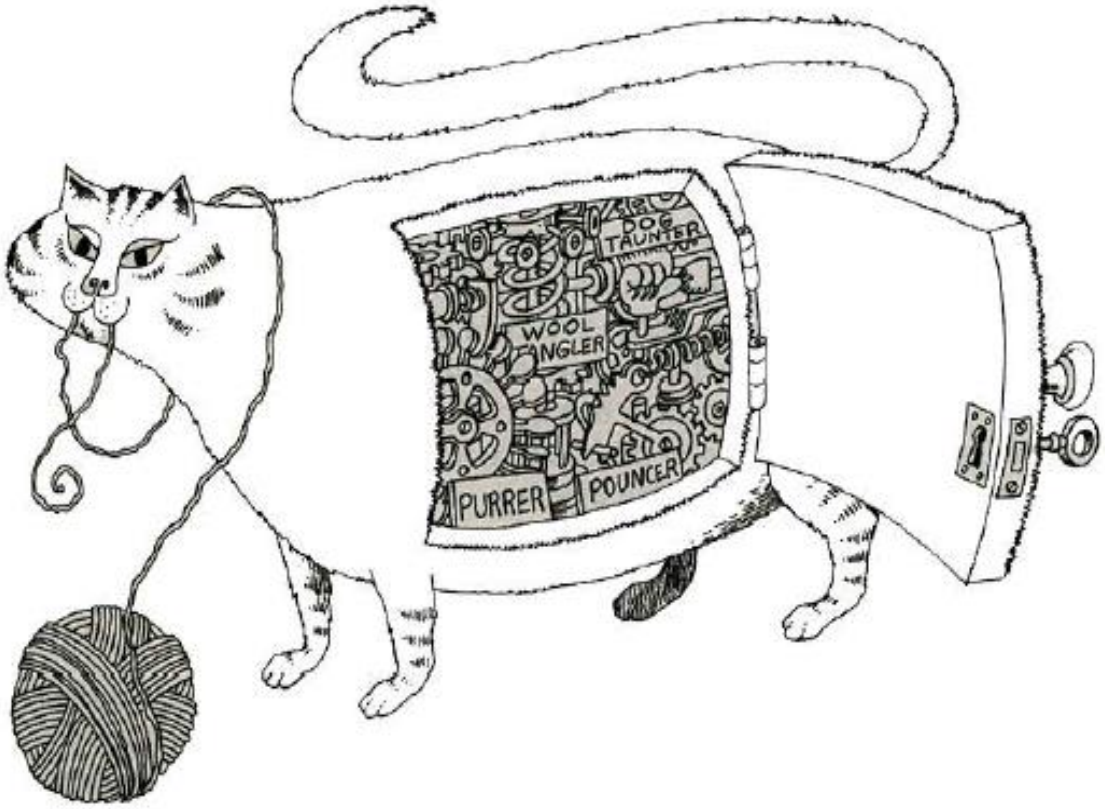
Φοιτητής Z:



Το κάθε αντικείμενο έχει

- Μια κατάσταση (το συγκεκριμένο όνομα και τους συγκεκριμένους βαθμούς)
- Ενέργειες (τις λειτουργείες/μεθόδους)
- Ταυτότητα (X,Y,Z)

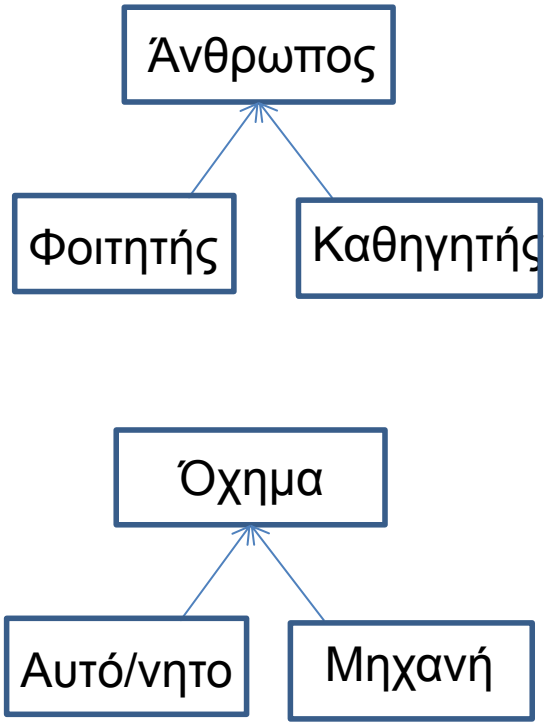
Ενθυλάκωση



- Η στεγανοποίηση της κατάστασης και της συμπεριφοράς ώστε οι λεπτομέρειες της υλοποίησης να είναι κρυμμένες από το χρήστη του αντικειμένου.

Κληρονομικότητα

- Οι κλάσεις μας επιτρέπουν να ορίσουμε μια ιεραρχία
  - Π.χ., και ο Φοιτητής και ο Καθηγητής ανήκουν στην κλάση Άνθρωπος.
  - Η κλάση Αυτοκίνητο ανήκει στην κλάση Όχημα η οποία περιέχει και την κλάση Μοτοσυκλέτα
- Οι κλάσεις πιο χαμηλά στην ιεραρχία κληρονομούν χαρακτηριστικά και συμπεριφορά από τις ανώτερες κλάσεις
  - Όλοι οι άνθρωποι έχουν όνομα
  - Όλα τα οχήματα έχουν μέθοδο drive, stop.



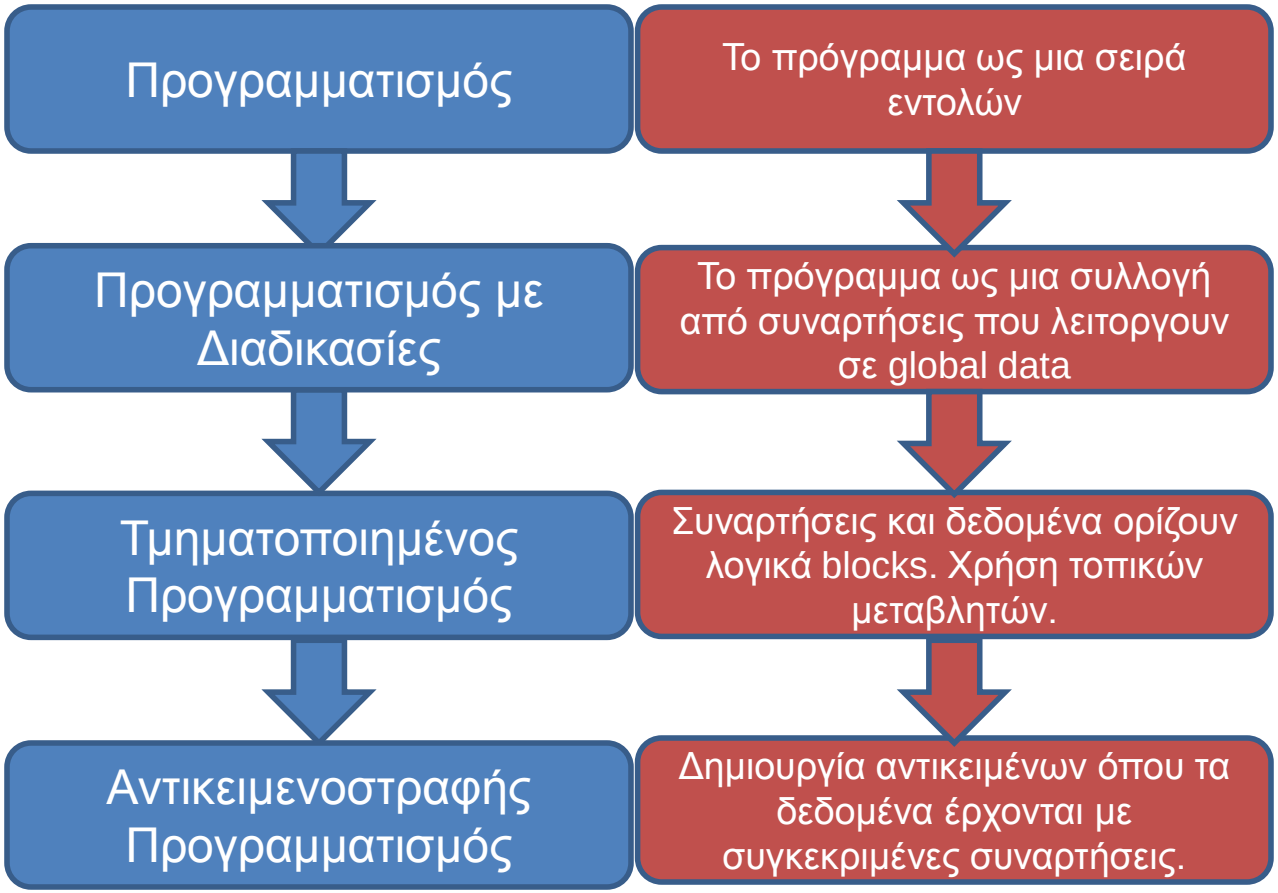
Πολυμορφισμός

- Κλάσεις με κοινό πρόγονο έχουν κοινά χαρακτηριστικά, αλλά έχουν και διαφορές
  - Π.χ., είναι διαφορετικό το παρκάρισμα για ένα αυτοκίνητο και μια μηχανή
- Ο πολυμορφισμός μας επιτρέπει να δώσουμε μια κοινή συμπεριφορά σε κάθε κλάση (μια μέθοδο park), η οποία όμως υλοποιείται διαφορετικά για αντικείμενα διαφορετικών κλάσεων.
- Μπορούμε επίσης να ορίσουμε αφηρημένες κλάσεις, όπου προϋποθέτουμε μια συμπεριφορά και αυτή πρέπει να υλοποιηθεί σε χαμηλότερες κλάσεις διαφορετικά ανάλογα με τις ανάγκες μας

Αφηρημένοι Τύποι Δεδομένων

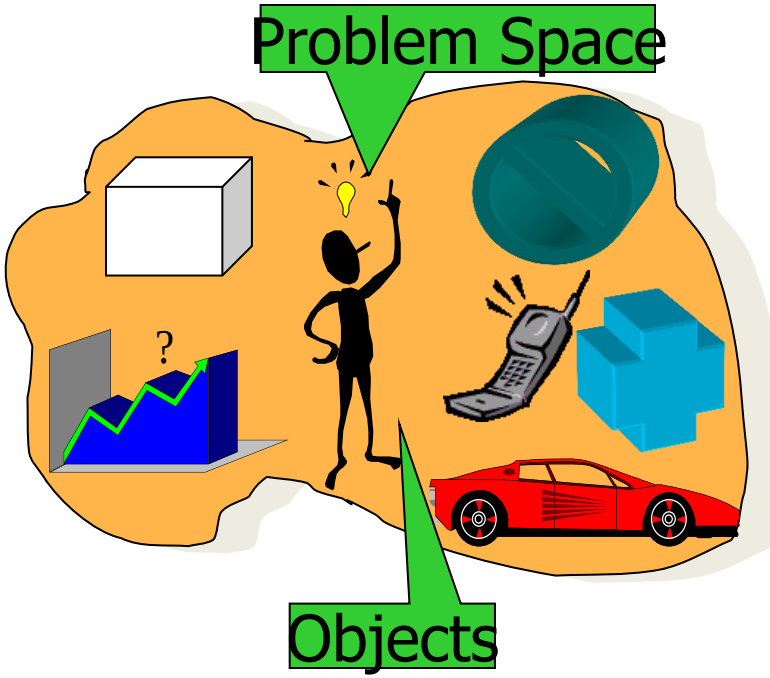
- Χρησιμοποιώντας τις κλάσεις μπορούμε να ορίσουμε τους δικούς μας τύπους δεδομένων
  - Έτσι μπορούμε να φτιάξουμε αντικείμενα με συγκεκριμένα χαρακτηριστικά και συμπεριφορά.
- Χρησιμοποιώντας την κληρονομικότητα και τον πολυμορφισμό, μπορούμε να επαναχρησιμοποιήσουμε υπάρχοντα χαρακτηριστικά και μεθόδους.

Η εξέλιξη του προγραμματισμού



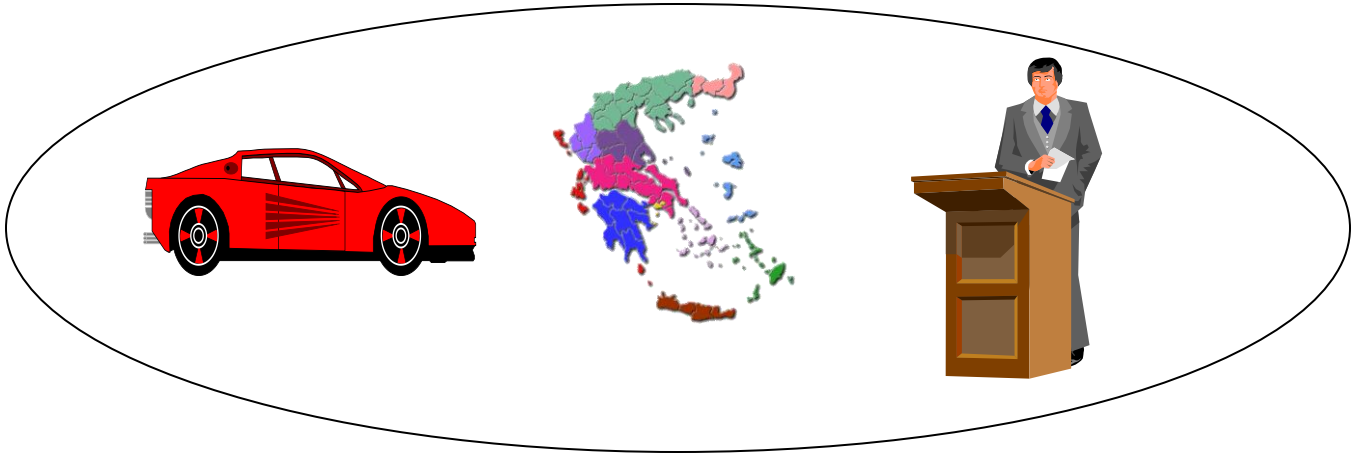
Διαδικασιακός vs. Αντικειμενοστραφής Προγραμματισμός

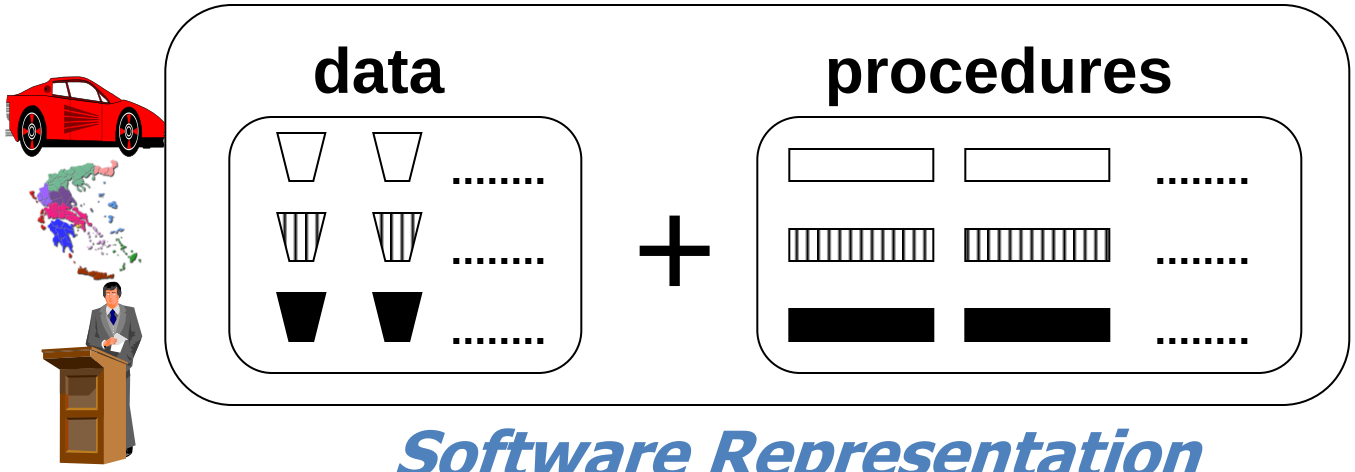
- Διαδιδικασιακός: Έμφαση στις διαδικασίες
  - Οι δομές που δημιουργούμε είναι για να ταιριάζουν με τις διαδικασίες.
  - Οι διαδικασίες προκύπτουν από το χώρο των λύσεων.
- Αντικειμενοστραφής: Έμφαση στα αντικείμενα
  - Τα αντικείμενα δημιουργούνται από το χώρο του προβλήματος
  - Λειτουργούν ακόμη και αν αλλάξει το πρόβλημα μας



Διαδικασιακή αναπαράσταση

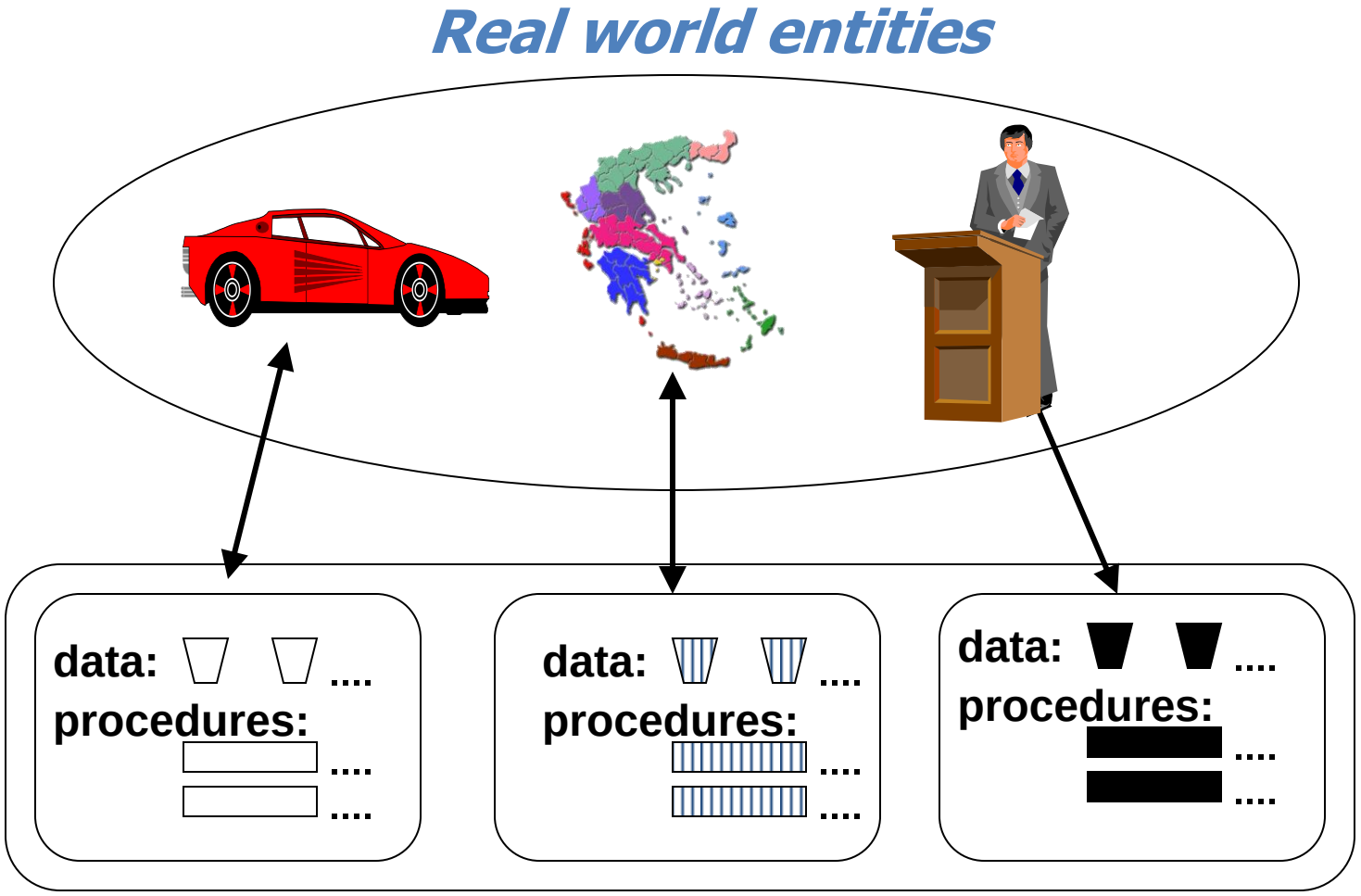
Real world entities





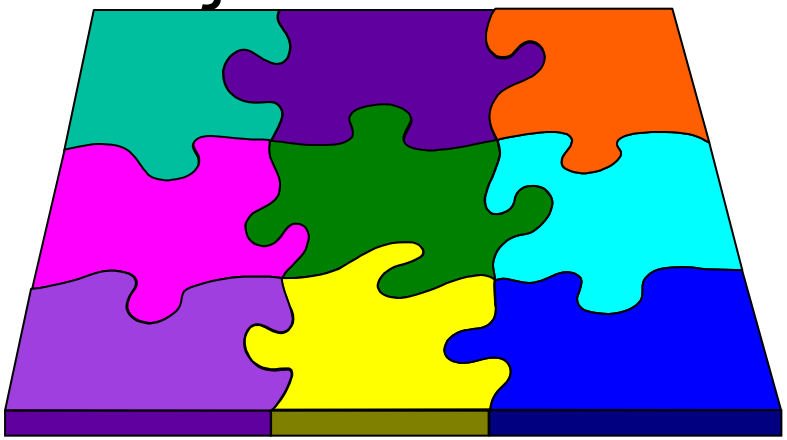
Software Representation

Αντικειμενοστραφής αναπαράσταση



Πλεονεκτήματα αντικειμενοστραφούς προγραμματισμού

- Επειδή προσπαθεί να μοντελοποιήσει τον πραγματικό κόσμο, ο OOP κώδικας είναι πιο κατανοητός.
  - Τα δομικά κομμάτια που δημιουργεί είναι πιο εύκολο να επαναχρησιμοποιηθούν και να συνδυαστούν

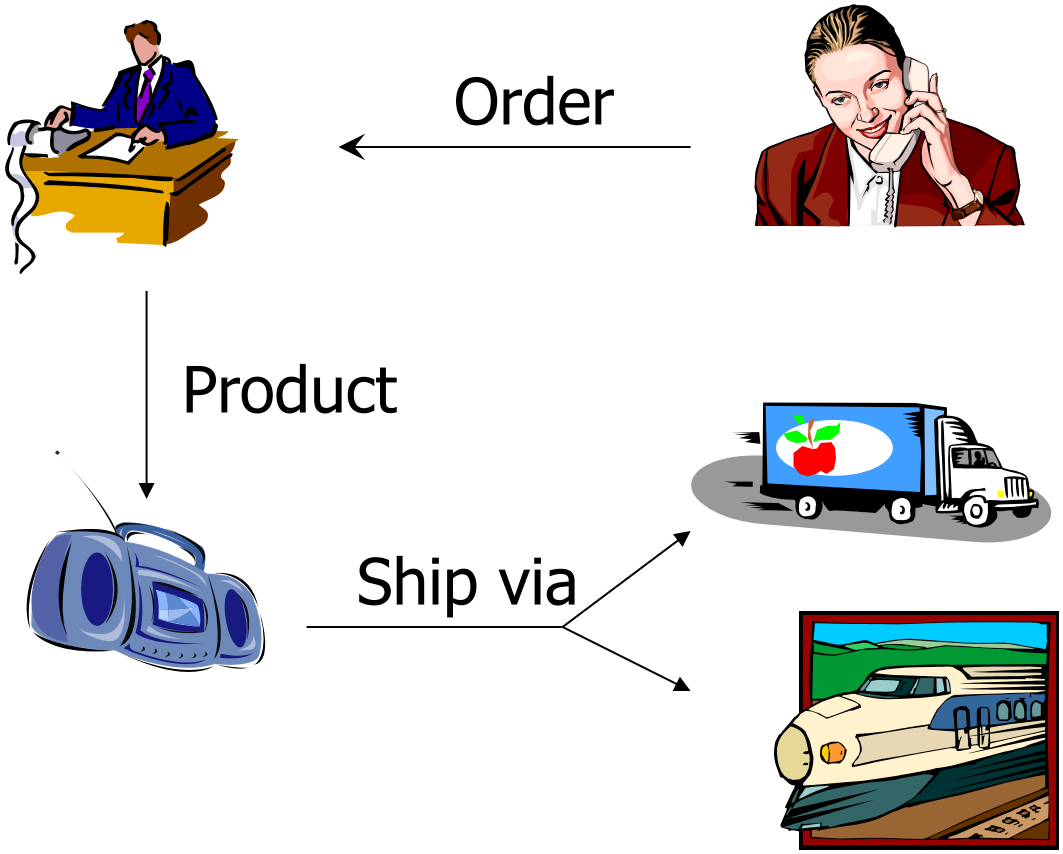


- Ο κώδικας είναι πιο εύκολο να συντηρηθεί λόγω της ενθυλάκωσης

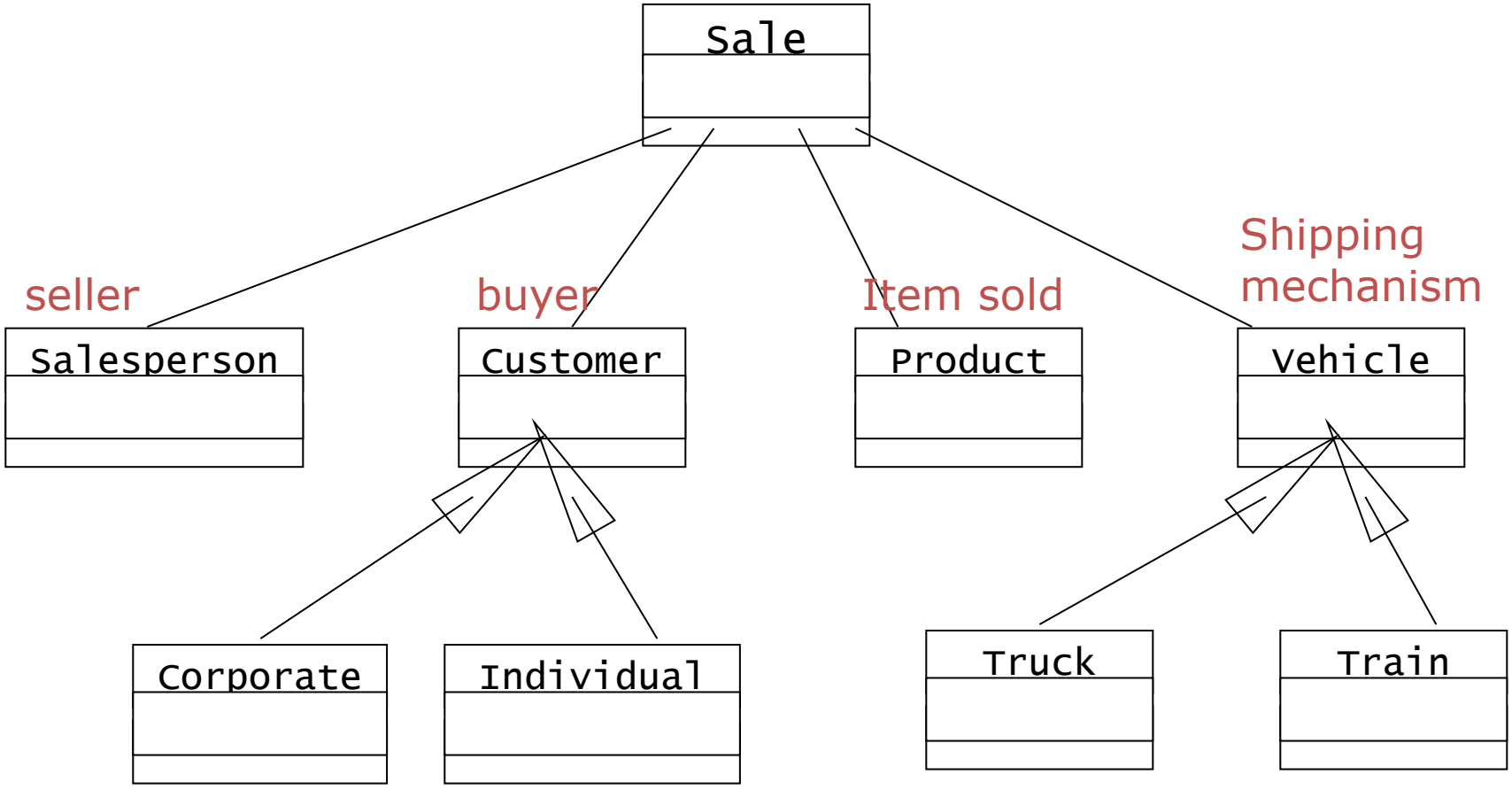
Παράδειγμα: Πωλήσεις

Θέλουμε να δημιουργήσουμε λειτουργικό για ένα σύστημα το οποίο διαχειρίζεται πωλήσεις.

- Πελάτες κάνουν παραγγελίες.
- Οι πωλητές χειρίζονται την παραγγελία
- Οι παραγγελίες είναι για συγκεκριμένα προϊόντα
- Η παραγγελία αποστέλλεται με επιλεγμένο μέσο



Διάγραμμα κλάσεων



Αλλαγή των απαιτήσεων

Προσθήκη αεροπορικής μεταφοράς