ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Εισαγωγή

Συστάσεις

- Ποιος είμαι εγώ:
  - Παναγιώτης Τσαπάρας
    - Email: tsap@cs.uoi.gr
    - Γραφείο: Β.3 (προτιμώμενες ώρες: μετά τις 10, πριν τις 7)
    - Web: http://www.cs.uoi.gr/~tsap
  - Ενδιαφέροντα
    - Web mining, Social networks, User Generated Content
    - Mobile applications, Mining of mobile data.

Συστάσεις

Συντονίστρια Εργαστηρίων

- Μαρία Χρόνη
  - Email: mchroni@cs.uoi.gr
  - Web: http://www.cs.uoi.gr/~mchroni/

- Βοηθοί: Θα οριστούν αργότερα

Γενικές πληροφορίες

- Web: http://www.cs.uoi.gr/~tsap/teaching/cse205/
- Διαλέξεις:
  - Τρίτη 3-5 μ.μ.
  - Πέμπτη 12-2 μ.μ.
- Εργαστήρια:
  - Πέμπτη 3-7 μ.μ.
  - Θα ξεκινήσουν σε μερικές εβδομάδες
- Ώρες γραφείου: Μπορείτε ανά πάσα στιγμή να χτυπήσετε την πόρτα του γραφείου μου. Πιο εύκολο να με βρείτε μέσω email.
- Φροντιστήρια: Ώρες για την απάντηση ερωτήσεων και βοήθεια με ασκήσεις.
- Παρασκευή: υποψήφια μέρα για αναπλήρωση χαμένων ωρών.

eCourse

- Θα πρέπει όλοι να γραφτείτε στο eCourse ώστε να βλέπετε ανακοινώσεις και λεπτομέρειες για το μάθημα. Θα ανοίξει την επόμενη εβδομάδα.
- Ανακοινώσεις και διαφάνειες θα εμφανίζονται και στη σελίδα του μαθήματος.

Βαθμολογία

- Η βαθμολογία θα καθοριστεί από τα παρακάτω:
  - Εργαστήρια ή Πρόοδος [20%]
  - Ασκήσεις [30%]
  - Τελική Εξέταση [50%]

- Η συμμετοχή στα εργαστήρια είναι υποχρεωτική μόνο για τους πρωτοετείς. Δεν θα υπάρχουν εργαστήρια για τα μεγαλύτερα έτη αλλά μπορείτε να κάνετε τις ασκήσεις και να πάρετε σχόλια.
- Όσοι έχετε ξαναπάρει το μάθημα Τεχνικές Αντικειμενοστραφούς Προγραμματισμού μπορείτε αν θέλετε να κρατήσετε το βαθμό του εργαστηρίου.
- Όσοι δεν έχετε βαθμό εργαστηρίου ή δεν θέλετε να κρατήσετε το βαθμό των εργαστηρίων, θα πρέπει να δώσετε υποχρεωτική πρόοδο.
- Οι ασκήσεις είναι υποχρεωτικές για όλους, δεν μπορείτε να κρατήσετε τον βαθμό των ασκήσεων από τα προηγούμενα χρόνια.
- Για να περάσετε το μάθημα θα πρέπει να έχετε γράψετε τουλάχιστον 4 στην τελική εξέταση, και να έχετε βαθμό τουλάχιστον 5 συνολικά.

Αλγόριθμος Βαθμολογίας

- if (έτος == 1)
  - X = βαθμός εργαστηρίων
- else if (έτος > 1)
  - if (Υπάρχει βαθμός εργαστηρίου == false)
    - X = βαθμός προόδου
  - else if (Υπάρχει βαθμός εργαστηρίου == true)
    - If (Θέλετε να κρατήσετε τον βαθμό του εργαστηρίου == true)
      - X = Παλιός βαθμός εργαστηρίου
    - else if (Θέλετε να κρατήσετε τον βαθμό του εργαστηρίου == false)
      - X = βαθμός προόδου
- A = βαθμός ασκήσεων
- T = βαθμός τελικής εξέτασης
- if (T >= 4)
  - B = 0.2*X + 0.3*A + 0.5*T
  - if (B >= 5)
    - Περάσατε το μάθημα
  - else if (B < 5)
    - Την επόμενη φορά
- else if (T < 4)
  - Την επόμενη φορά

Μάθημα

- Η παρακολούθηση και συμμετοχή βοηθάνε στην κατανόηση.
- Κάνετε ερωτήσεις. Καμία ερώτηση δεν είναι «χαζή».
- Κάτι που ξέρει πολύς κόσμος αν δεν το έχετε διδαχτεί δεν είναι απαραίτητο να το ξέρετε. Ρωτήστε να το εξηγήσουμε.
- Αν κάτι είναι δυσνόητο ζητήστε να το επαναλάβουμε ή να δώσουμε παραδείγματα.
- Χρησιμοποιείστε τα εργαστήρια για να καταλάβετε καλύτερα.
- Χρησιμοποιήστε τα φροντιστήρια για να κάνετε ερωτήσεις για τις ασκήσεις και για θέματα που δεν έχετε καταλάβει.

Συμπεριφορά

- Σεβαστείτε τους συμφοιτητές σας και τους διδάσκοντες, μην κάνετε φασαρία.
- Δεν είσαστε στο σχολείο πλέον, έρχεστε στα μαθήματα γιατί το επιλέγετε
- Αν δεν ενδιαφέρεστε να παρακολουθήσετε δεν υπάρχει λόγος να έρχεστε στο μάθημα.

Στόχοι του μαθήματος

- Να μάθετε τις βασικές αρχές και τεχνικές του αντικειμενοστραφούς προγραμματισμού (object oriented programming)
- Να εξασκηθείτε στην πράξη με την γλώσσα προγραμματισμού Java
- Να κάνετε τα πρώτα σας «μεγάλα» προγράμματα

Ύλη που θα καλύψουμε

- Αρχές αντικειμενοστραφούς προγραμματισμού
  - Κλάσεις και αντικείμενα
  - Ενθυλάκωση και απόκρυψη
  - Πολυμορφισμός και Κληρονομικότητα
  - Αφηρημένες κλάσεις, Διεπαφές (Interfaces)
  - Γενικευμένες κλάσεις, συλλογές

- Εισαγωγή στη Java
  - Βασικό συντακτικό και δομή προγράμματος
  - Είσοδος, έξοδος δεδομένων
  - Εξαιρέσεις
  - Γραφικά/Μικροεφαρμογές

Βιβλιογραφία -Εύδοξος

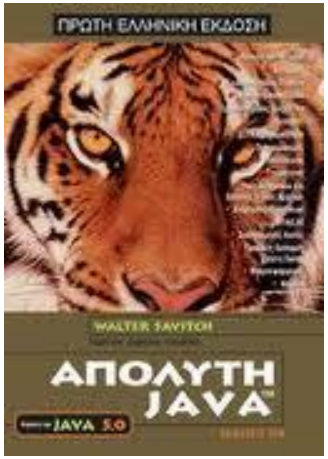
- Απόλυτη Java (περιέχει CD), Savitch Walter Λεπτομέρειες

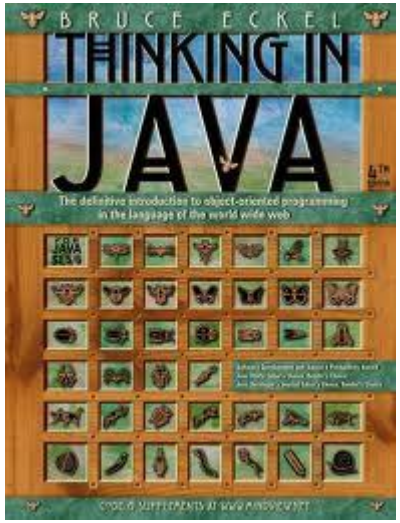
- JAVA ΜΕ UML: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΣΧΕΔΙΑΣΗ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ, ELSE LERVIK, VEGARD B. HAVDAL Λεπτομέρειες

- ΑΝΑΠΤΥΞΗ ΠΡΟΓΡΑΜΜΑΤΩΝ ΣΕ JAVA: ΑΦΑΙΡΕΣΕΙΣ, ΠΡΟΔΙΑΓΡΑΦΕΣ, ΚΑΙ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΣΧΕΔΙΑΣΜΟΣ, BARBARA LISKOV, JOHN GUTTAG Λεπτομέρειες

Βιβλιογραφία

Το κύριο βιβλίο του μαθήματος θα είναι:
Απολυτη Java, Walter Savitch





Δωρεάν online βιβλίο: Thinking In Java, Bruce Eckel
http://www.mindview.net/Books/TIJ/

Οι διαφάνειες του μαθήματος θα μπαίνουν στη σελίδα του μαθήματος και θα εκτυπωθούν οι διαφάνειες από πέρυσι.

Βιβλιογραφία

- Java Docs: Online documentation της Oracle για τη γλώσσα Java
  - Λεπτομερής περιγραφή για κάθε κλάση και κάθε μέθοδο
- Το Web: Για κάθε προγραμματιστική (ή άλλη) ερώτηση που έχετε μπορείτε να βρείτε απαντήσεις online.
  - Π.χ., stackoverflow.com είναι ένα online forum στο οποίο έμπειροι προγραμματιστές απαντάνε σε ερωτήσεις
  - Για κάθε μήνυμα λάθους μπορείτε να βρείτε πληροφορίες για το τι σημαίνει και πως μπορείτε να το λύσετε.
- Βοηθάει για να εξοικειωθείτε και με την αγγλική ορολογία, θα την χρησιμοποιούμε κατά καιρούς και στο μάθημα.

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

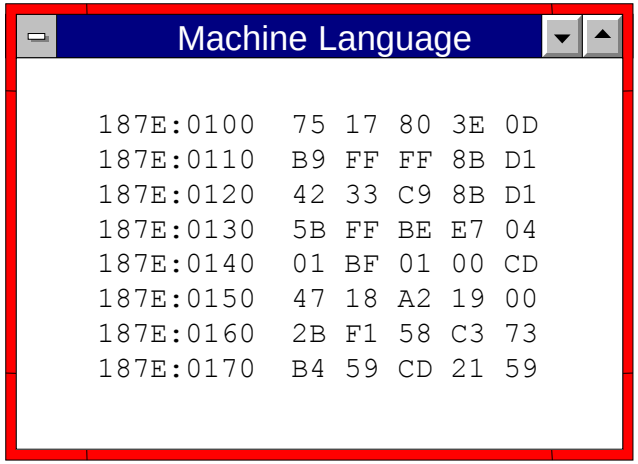
- Πρώτη γενιά: Γλώσσες μηχανής

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.



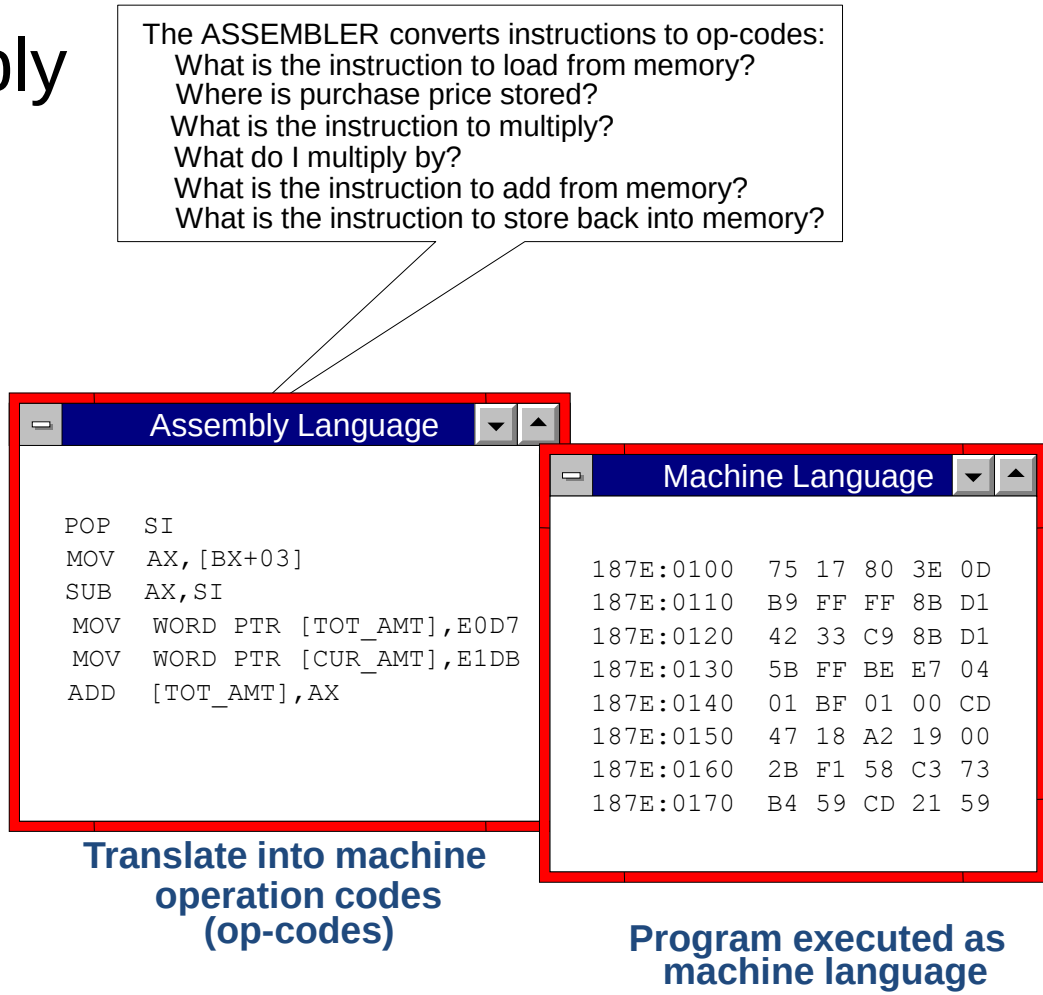
Program entered and executed as machine language

Πέντε γενεές γλωσσών προγραμματισμού

- Πρώτη γενιά: Γλώσσες μηχανής
- Δεύτερη γενιά: Assembly

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



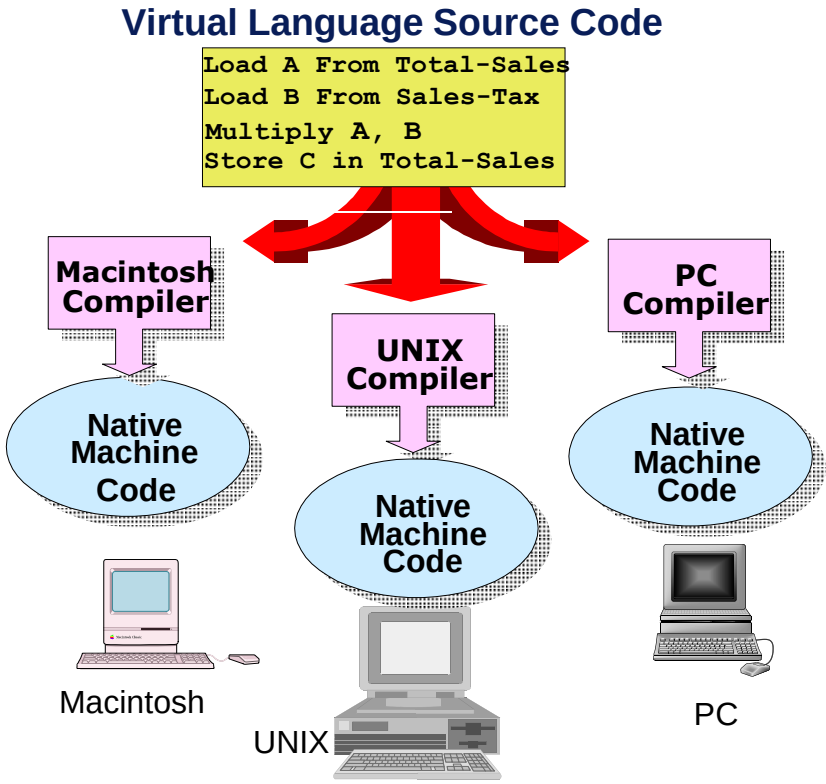
Πέντε γενεές γλωσσών προγραμματισμού

- 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)

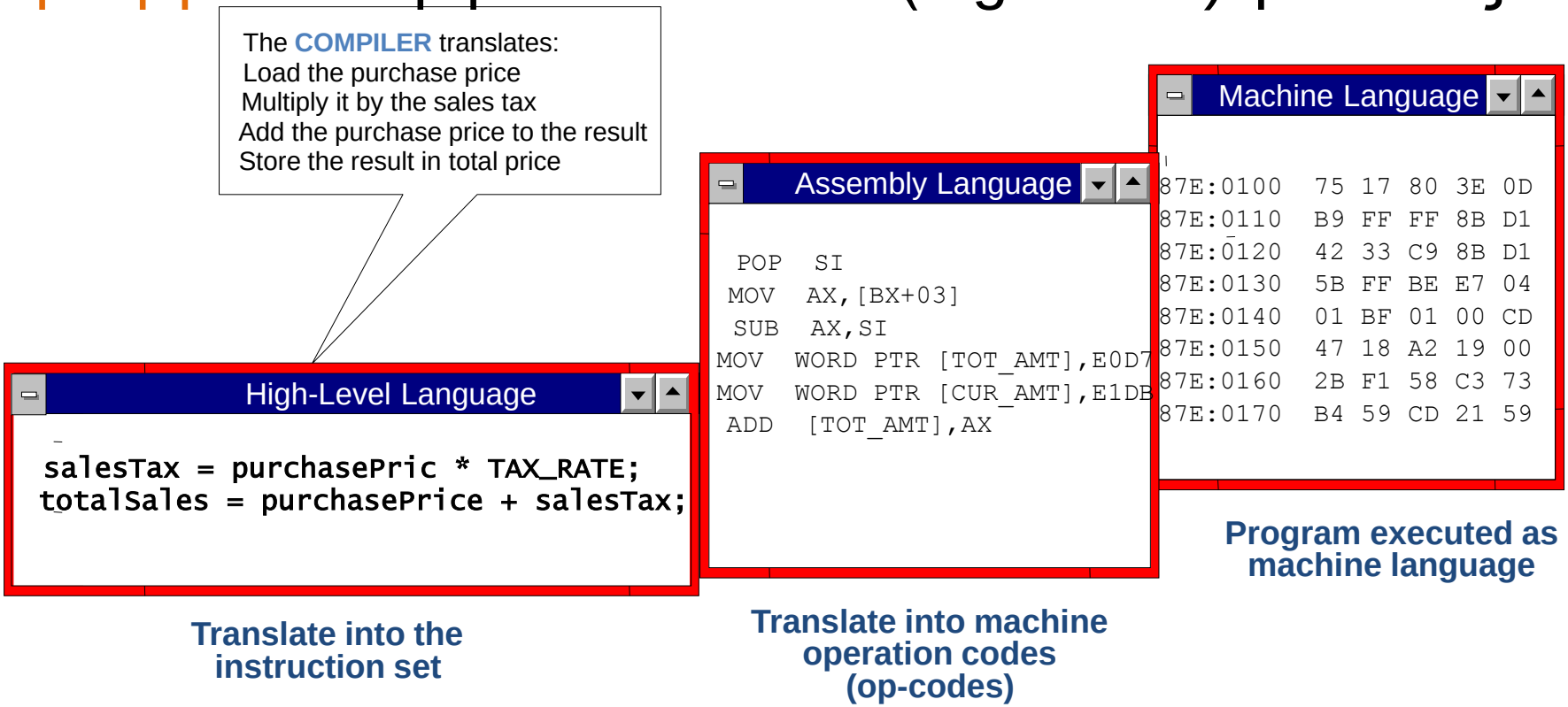
Ο compiler τις μετατρέπει σε ενδιάμεσο κώδικα (object code)

Ο ενδιάμεσος κώδικας μετατρέπεται σε γλώσσα μηχανής (machine code)



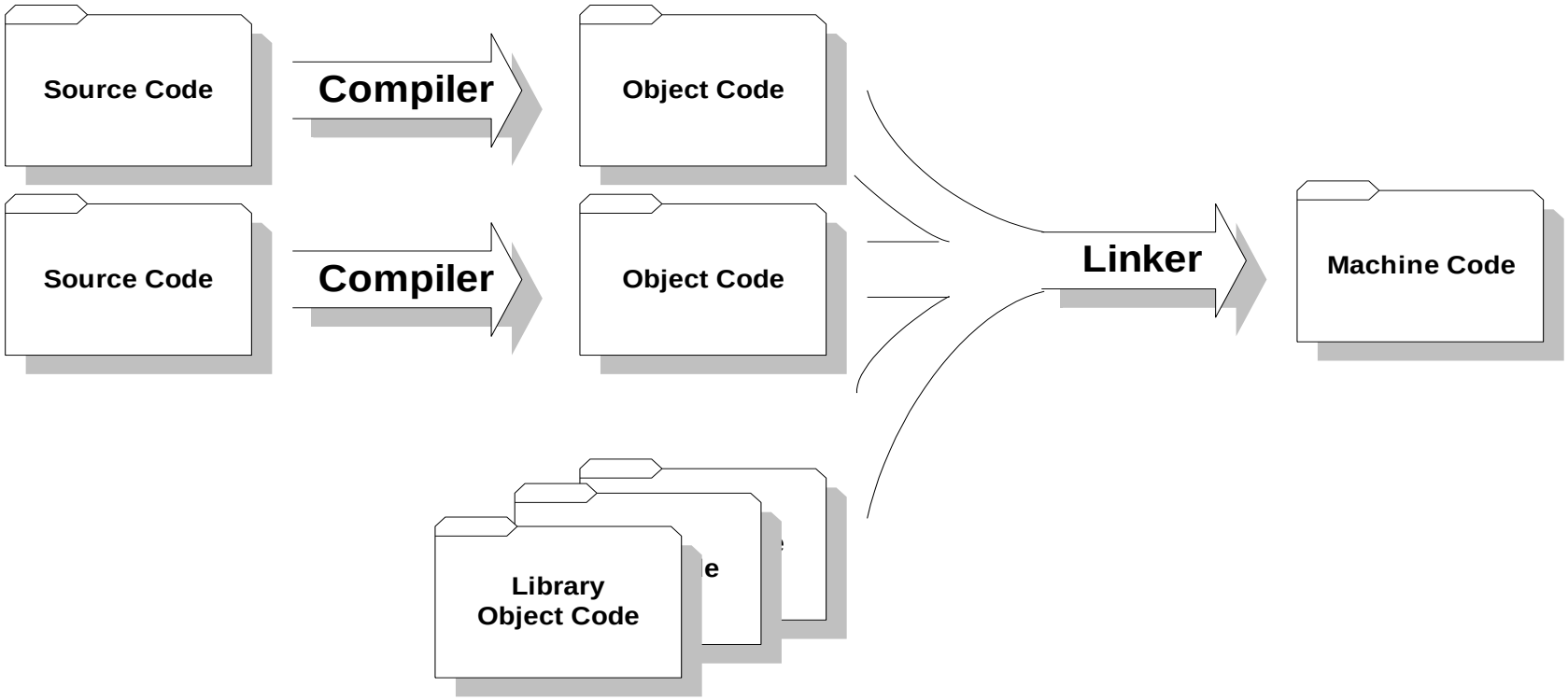
Πέντε γενεές γλωσσών προγραμματισμού

- 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 Παραδείγματα (paradigms)

- Προγραμματισμός των πρώτων ημερών.

Spaghetti code

Δύσκολο να διαβαστεί και να κατανοηθεί η ροή του

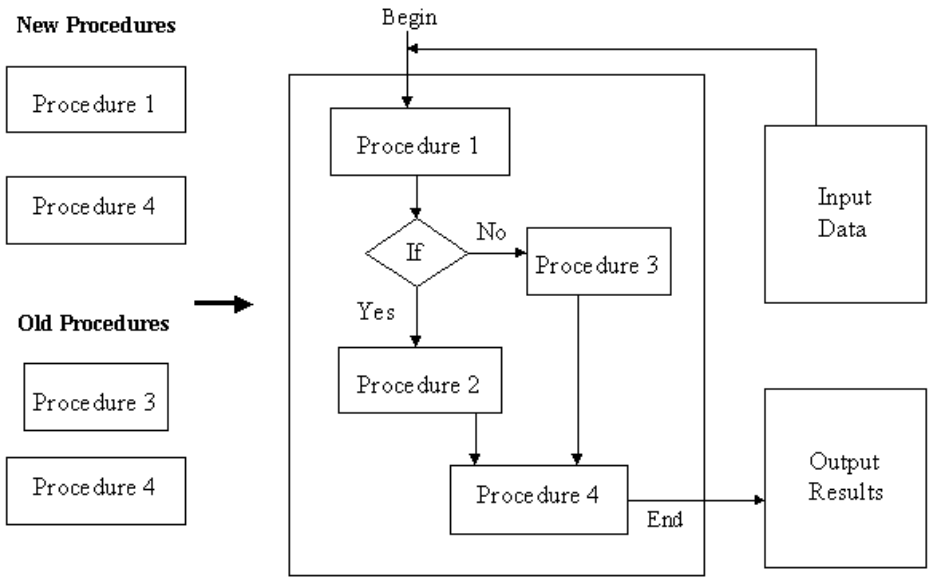
```
100   GOTO 500
110   PRINT I;
120   GOTO 400
150   PRINT I * 12;
160   GOTO 450
200   PRINT " = ";
210   GOTO 150
300   PRINT " 12 ";
310   GOTO 200
400   PRINT " * ";
410   GOTO 300
450   I = I + 1
460   IF I > 12 THEN STOP
460   GOTO 110
500   PRINT "The Twelves Table"
510   I = 1
520   GOTO 110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

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

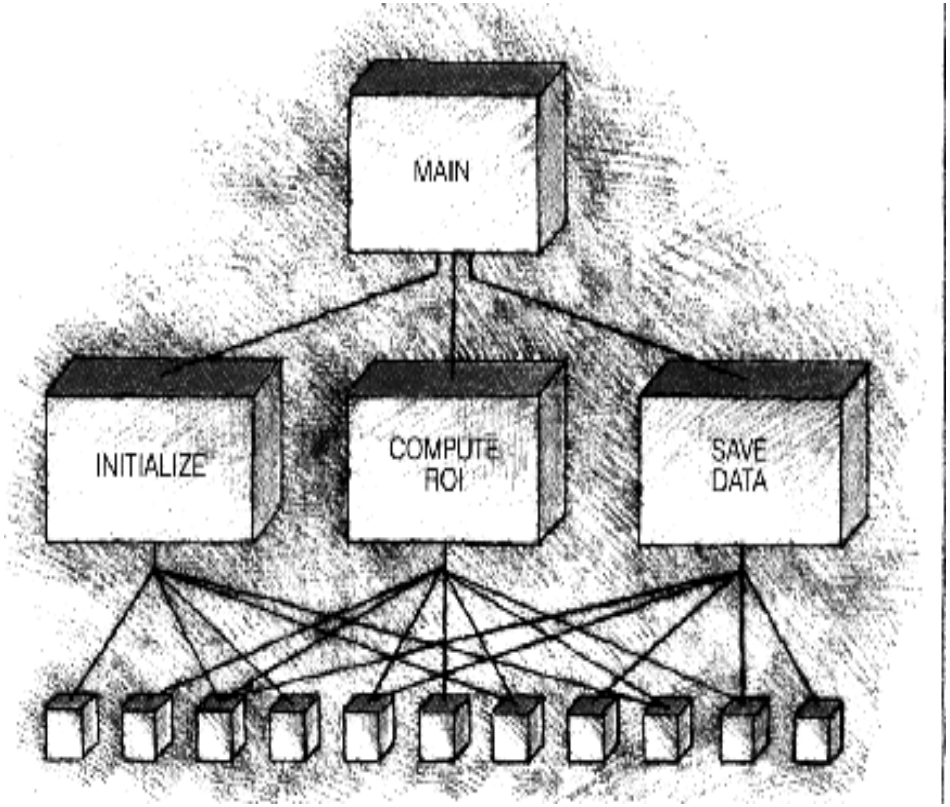
- Το πρόγραμμα μας σπάει σε πολλαπλές διαδικασίες.
  - Κάθε διαδικασία λύνει ένα υπο-πρόβλημα και αποτελεί μια λογική μονάδα (module)
  - Μια διαδικασία μπορούμε να την επαναχρησιμοποιήσουμε σε διαφορετικά δεδομένα.
- Το πρόγραμμα μας είναι τμηματοποιημένο (modular)

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!



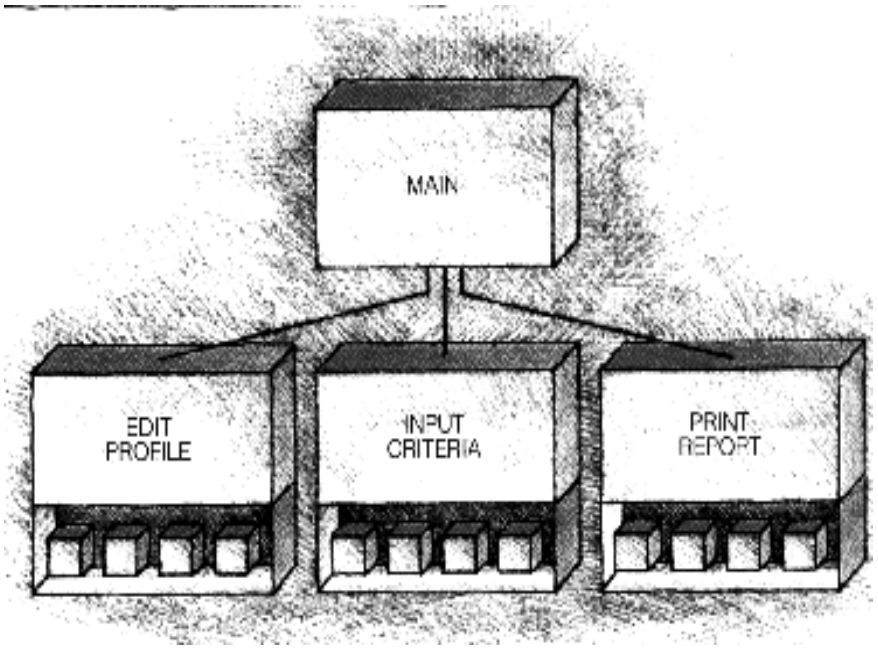
Απόκρυψη δεδομένων

- 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)

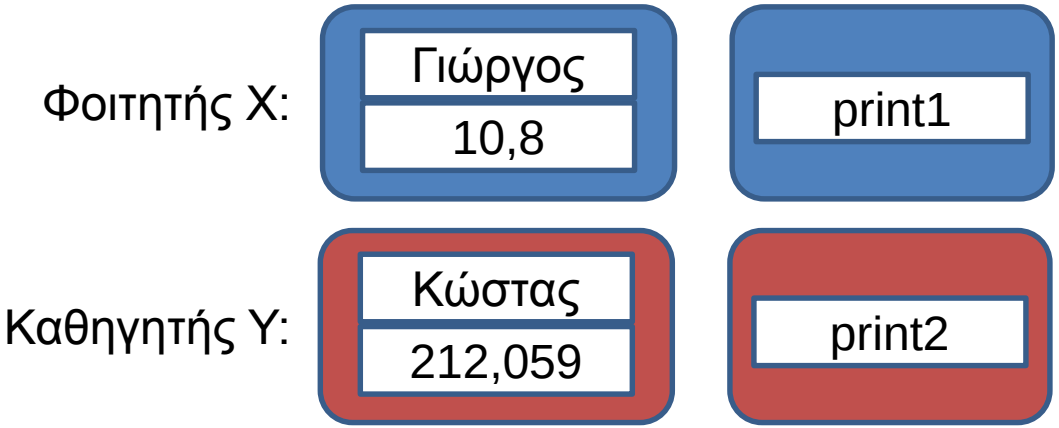


Περιορισμοί του διαδικασιακού προγραμματισμού

- 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

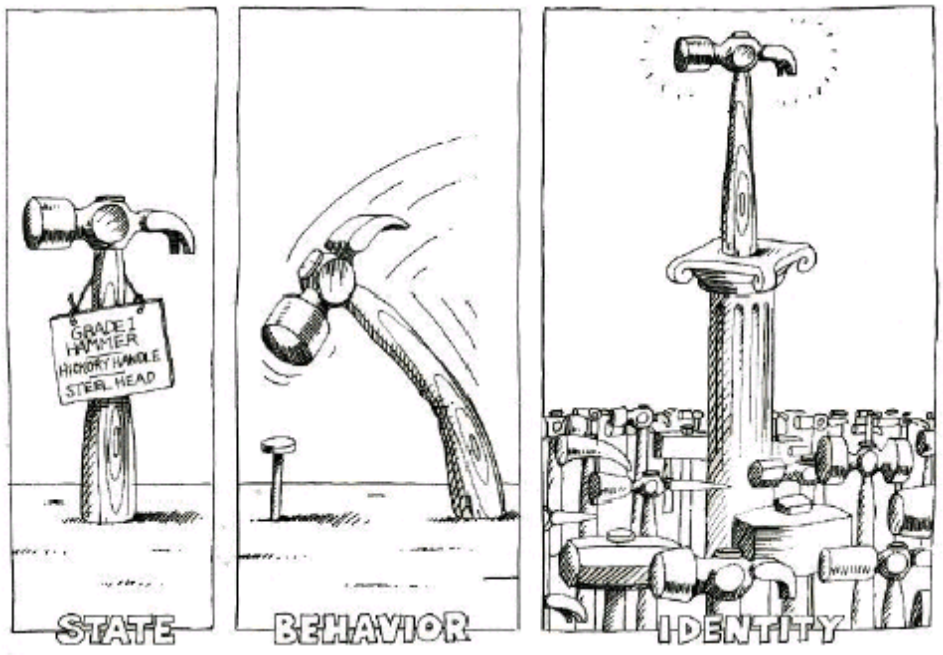
- Τα προβλήματα αυτά προσπαθεί να αντιμετωπίσει ο αντικειμενοστραφής προγραμματισμός (object-oriented programming)
  - OOP βάζει μαζί τα δεδομένα και τις διαδικασίες (μεθόδους) σχετικές με τα δεδομένα
  - Π.χ., ο κάθε φοιτητής ή καθηγητής έρχεται με μια δικιά του διαδικασία print

- Αυτό επιτυγχάνεται με αντικείμενα και κλάσεις

Αντικείμενο

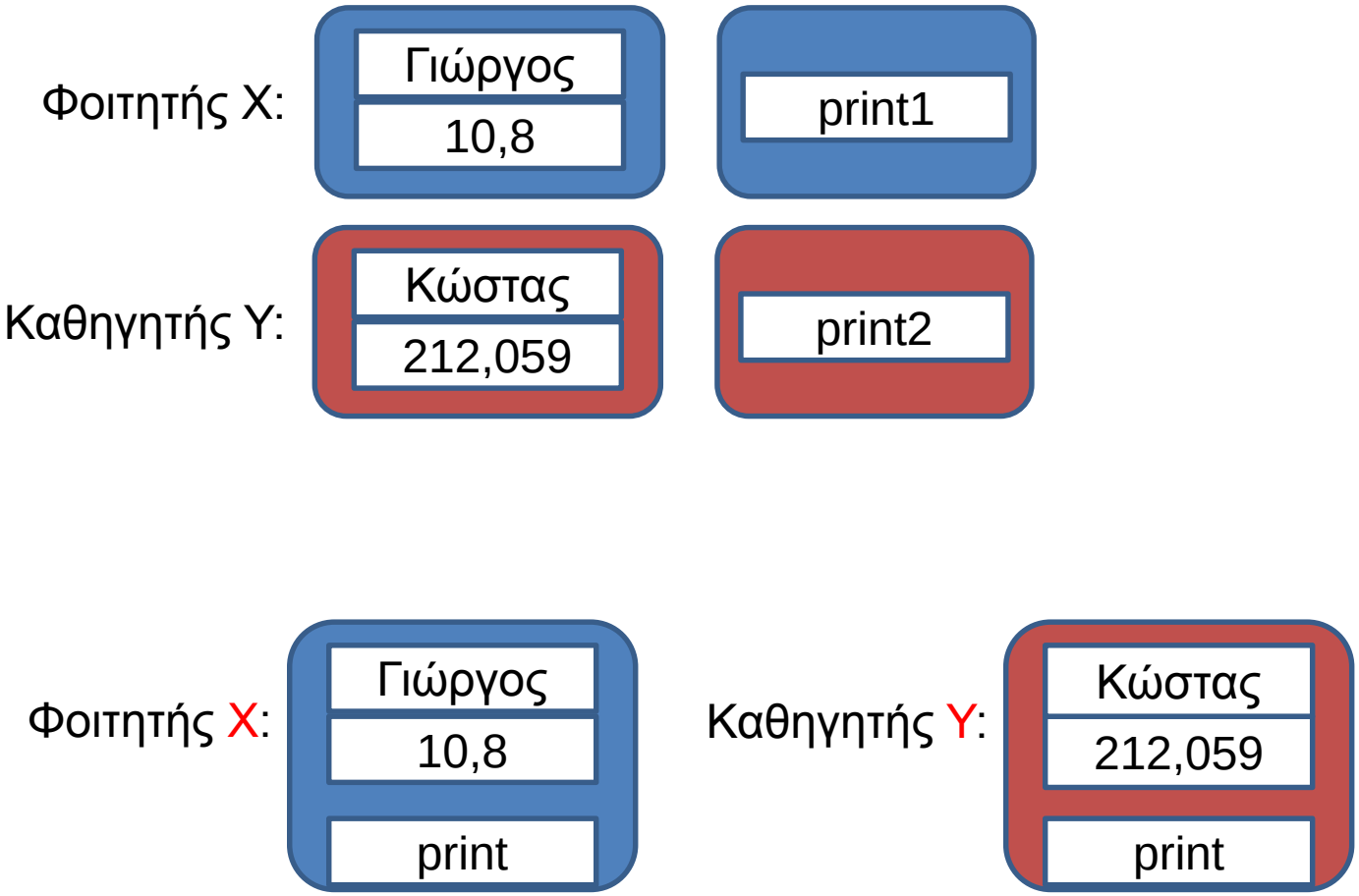
- Ένα αντικείμενο στον κώδικα αναπαριστά μια μονάδα/οντότητα/έννοια η οποία έχει:

- 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.



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

Παράδειγμα


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

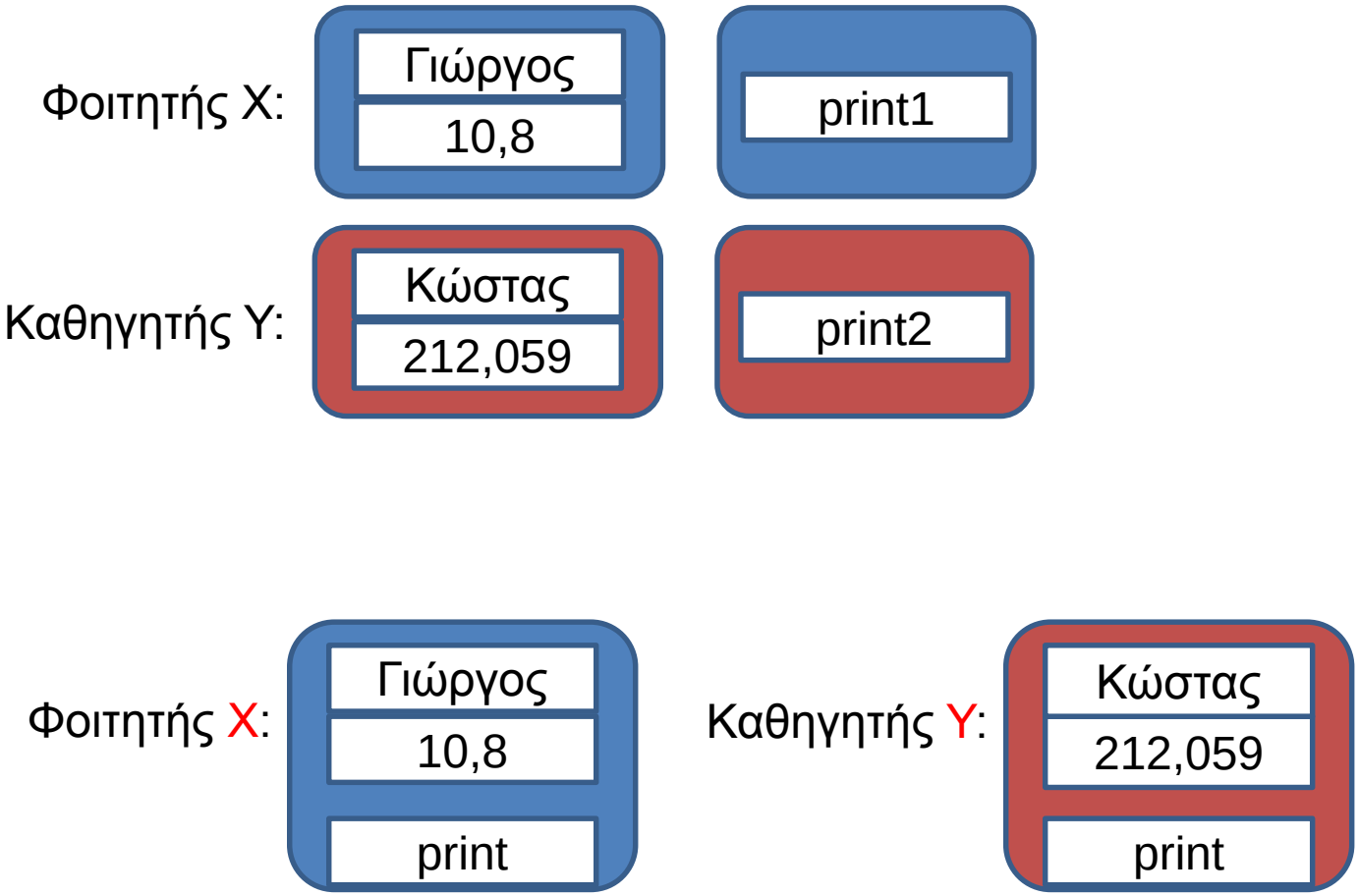
Κλάσεις

- 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

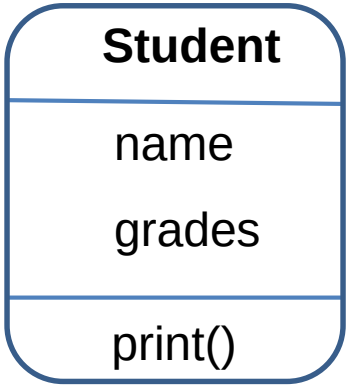
- Η κλάση Car έχει τα χαρακτηριστικά (brand, color) και τη συμπεριφορά (drive, stop)
  - Το αυτοκίνητο INI2013 είναι ένα αντικείμενο της κλάσης Car με κατάσταση τα χαρακτηριστικά (BMW, red)

Παράδειγμα


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

Κλάσεις και Αντικείμενα

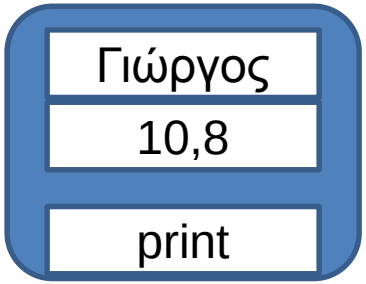
Κλάση



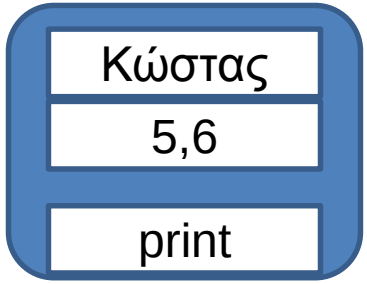
Μια αφηρημένη περιγραφή ενός φοιτητή. Περιλαμβάνει τις ιδιότητες του και τις λειτουργίες του

Αντικείμενα

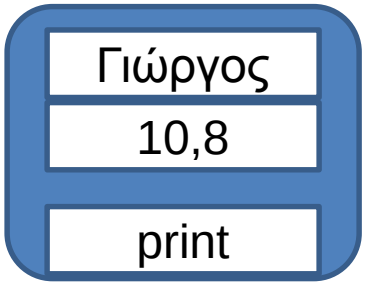
Φοιτητής X:



Φοιτητής Y:



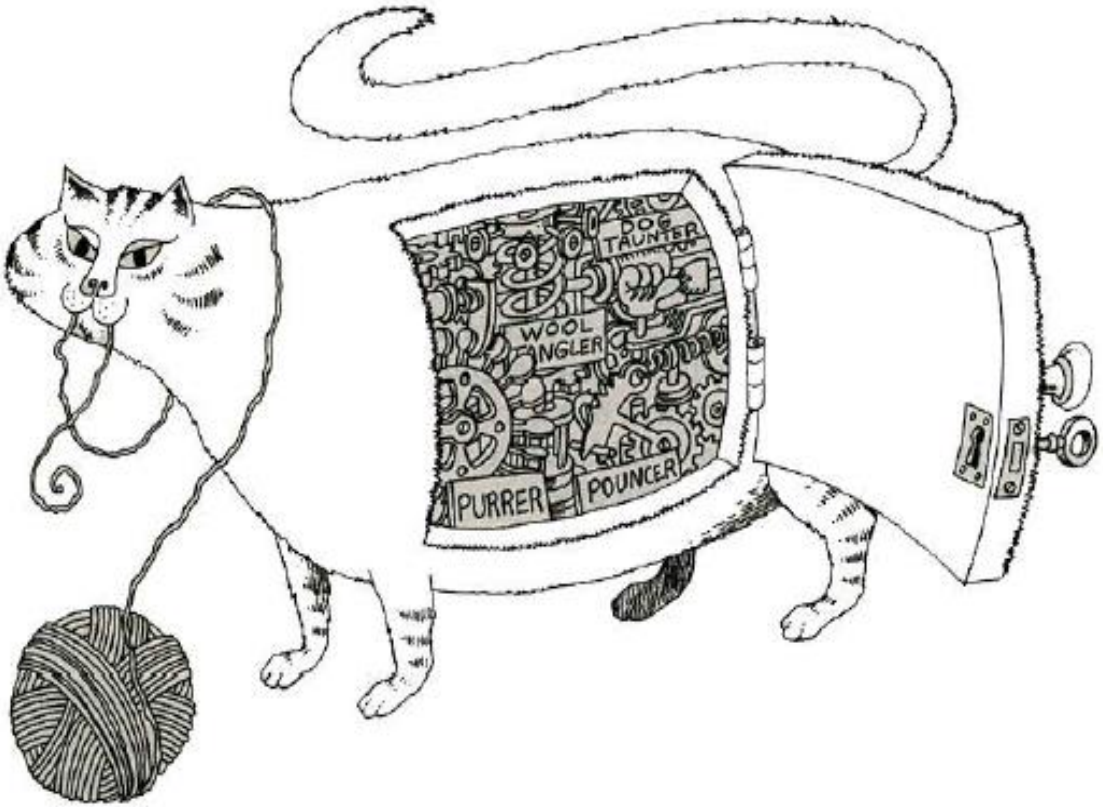
Φοιτητής Z:



Το κάθε αντικείμενο έχει

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

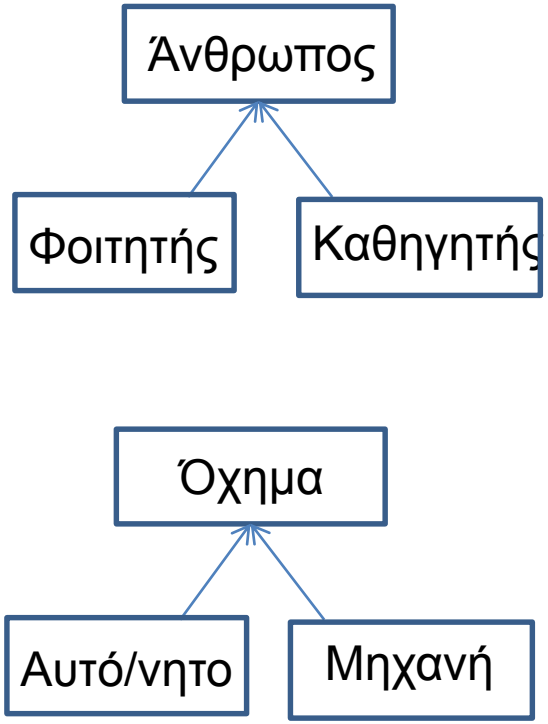
Ενθυλάκωση



- 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.



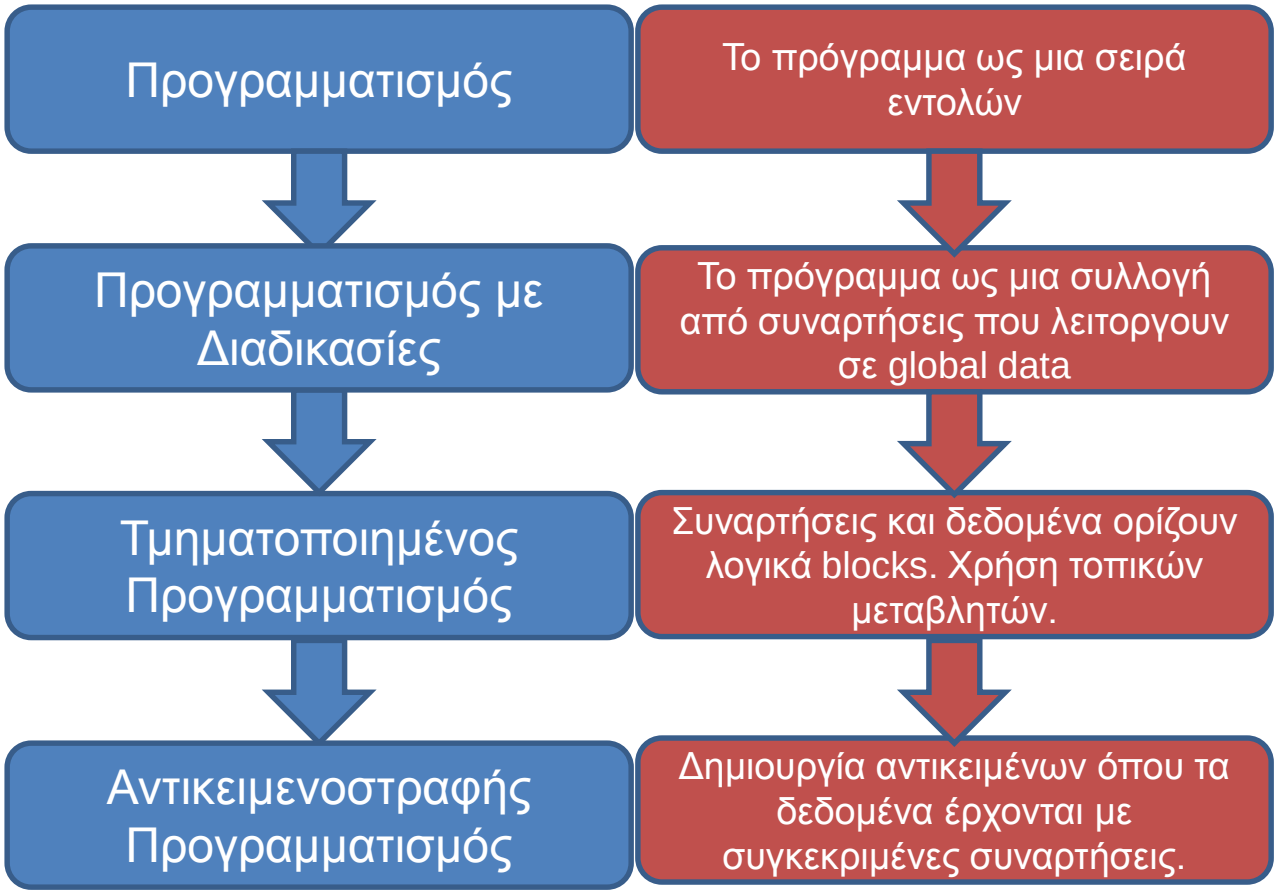
Πολυμορφισμός

- 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

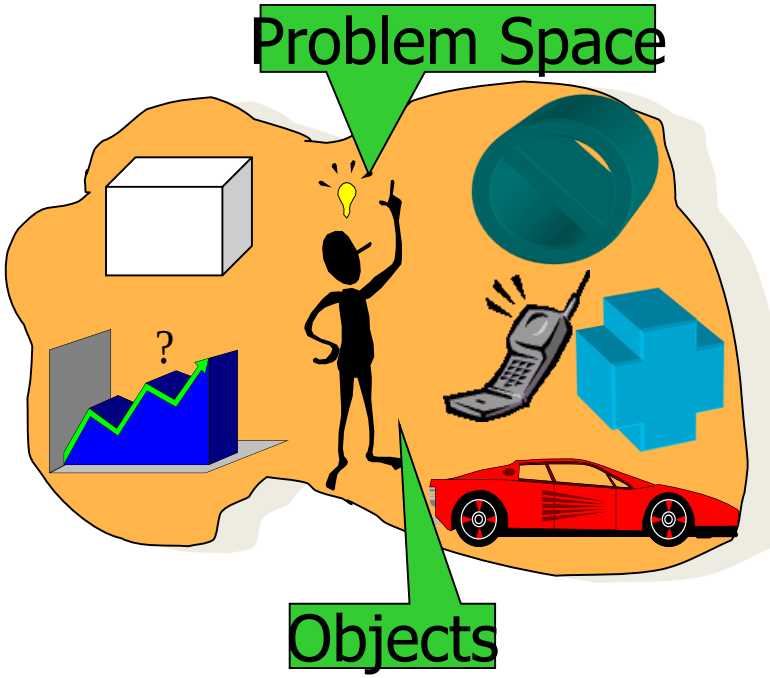
- Χρησιμοποιώντας τις κλάσεις μπορούμε να ορίσουμε τους δικούς μας τύπους δεδομένων
  - Έτσι μπορούμε να φτιάξουμε αντικείμενα με συγκεκριμένα χαρακτηριστικά και συμπεριφορά.
- Χρησιμοποιώντας την κληρονομικότητα και τον πολυμορφισμό, μπορούμε να επαναχρησιμοποιήσουμε υπάρχοντα χαρακτηριστικά και μεθόδους.

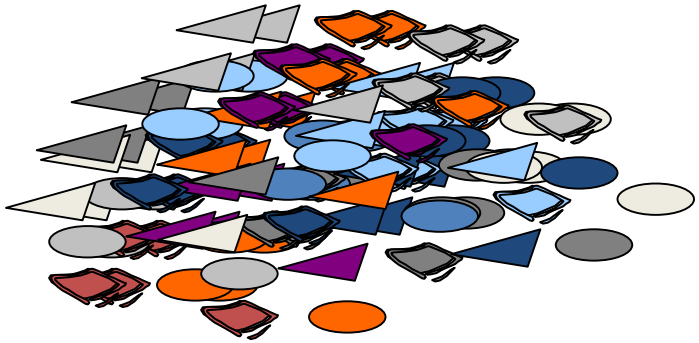
Η εξέλιξη του προγραμματισμού



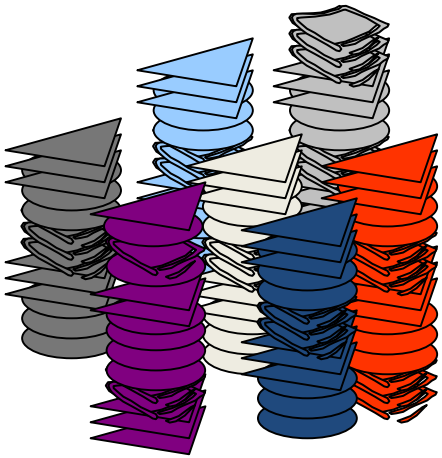
Διαδικασιακός vs. Αντικειμενοστραφής Προγραμματισμός

- 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





1st Generation
Spaghetti-Code



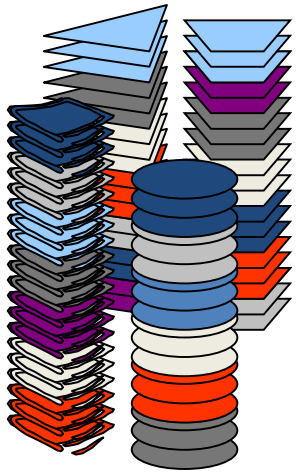
2nd & 3rd Generation :
functional decomposition

Software =

Data (Shapes)

+

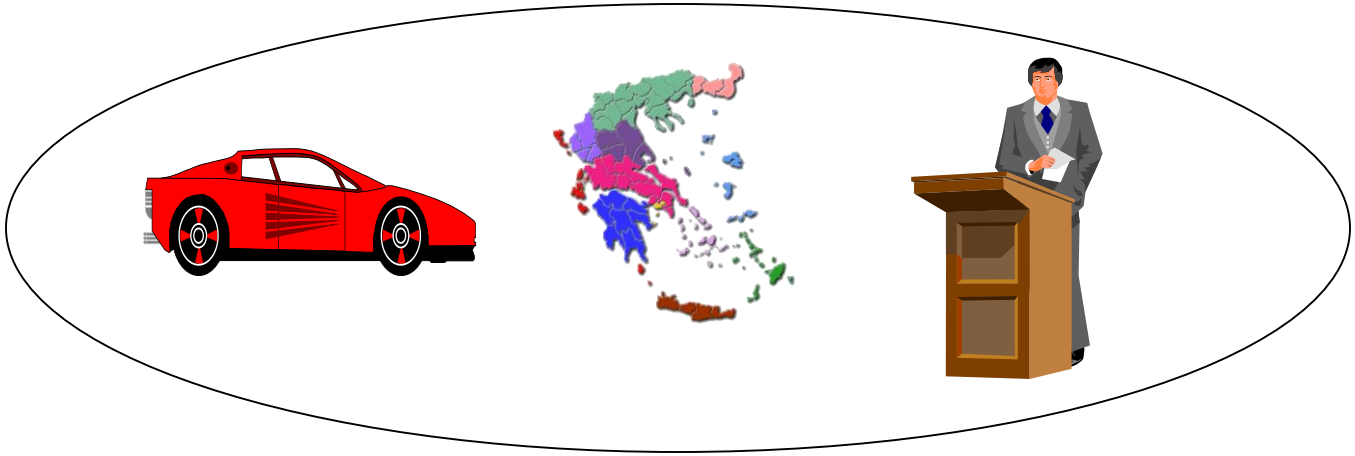
Functions (Colors)

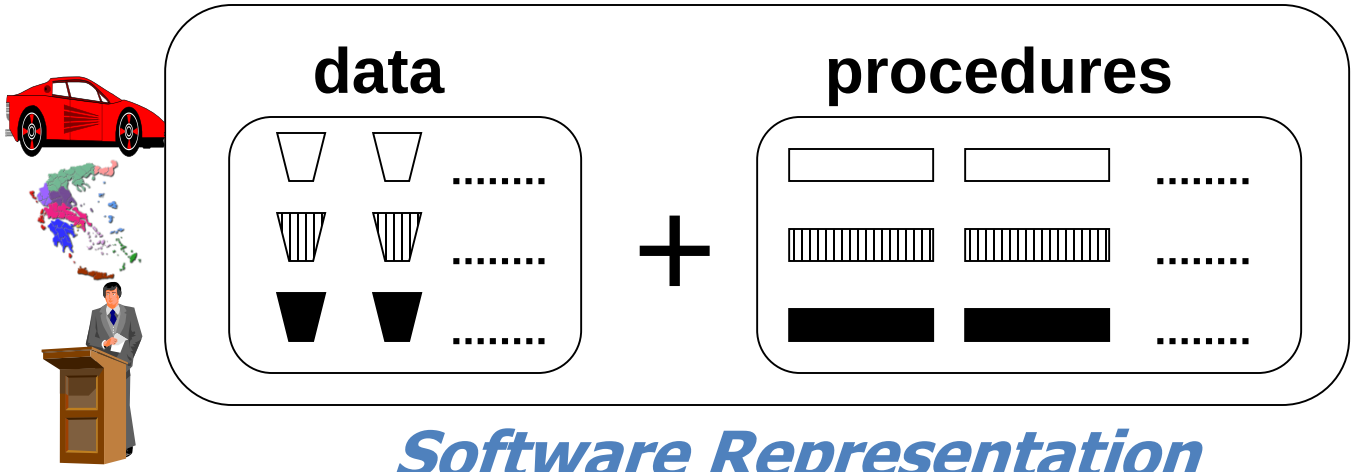


4th Generation
object decomposition

Διαδικασιακή αναπαράσταση

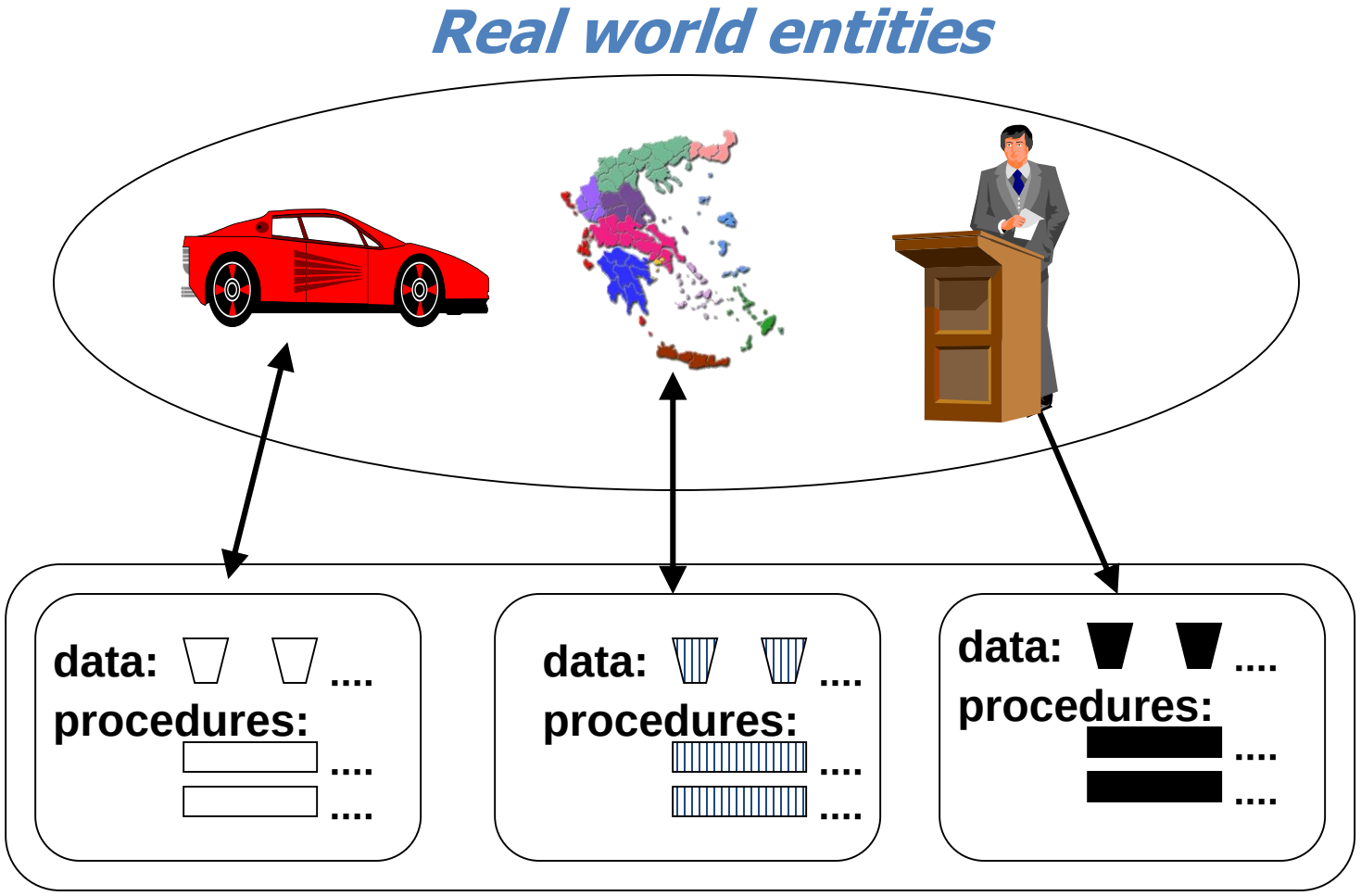
Real world entities





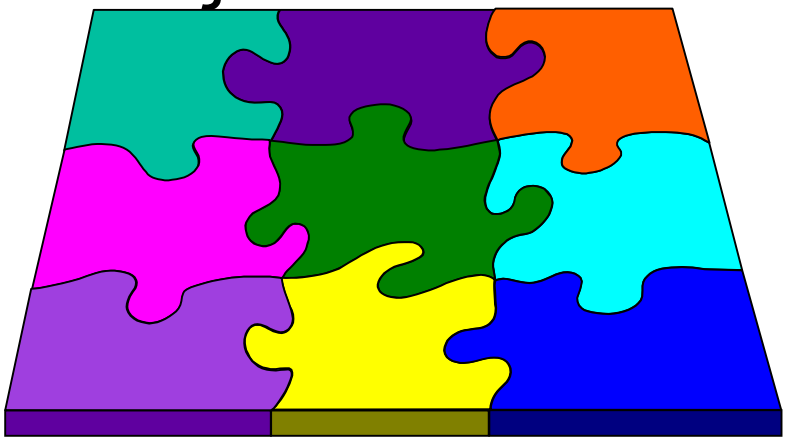
Software Representation

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

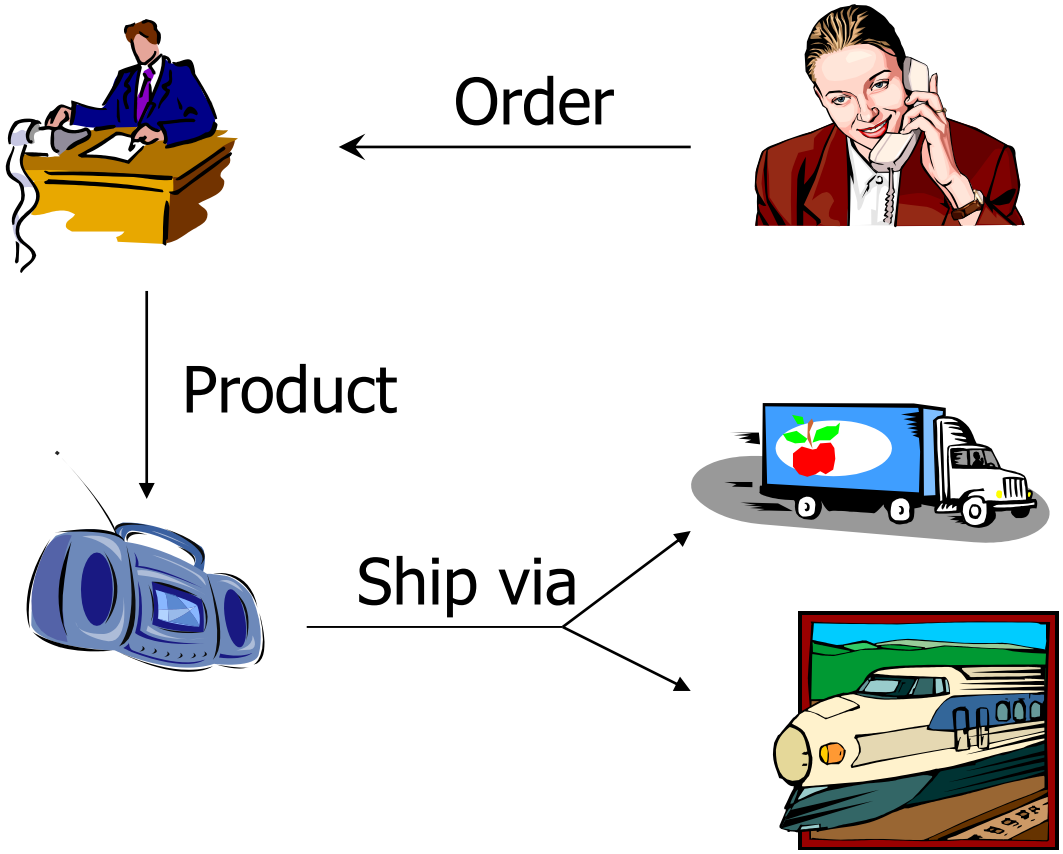


- Ο κώδικας είναι πιο εύκολο να συντηρηθεί λόγω της ενθυλάκωσης

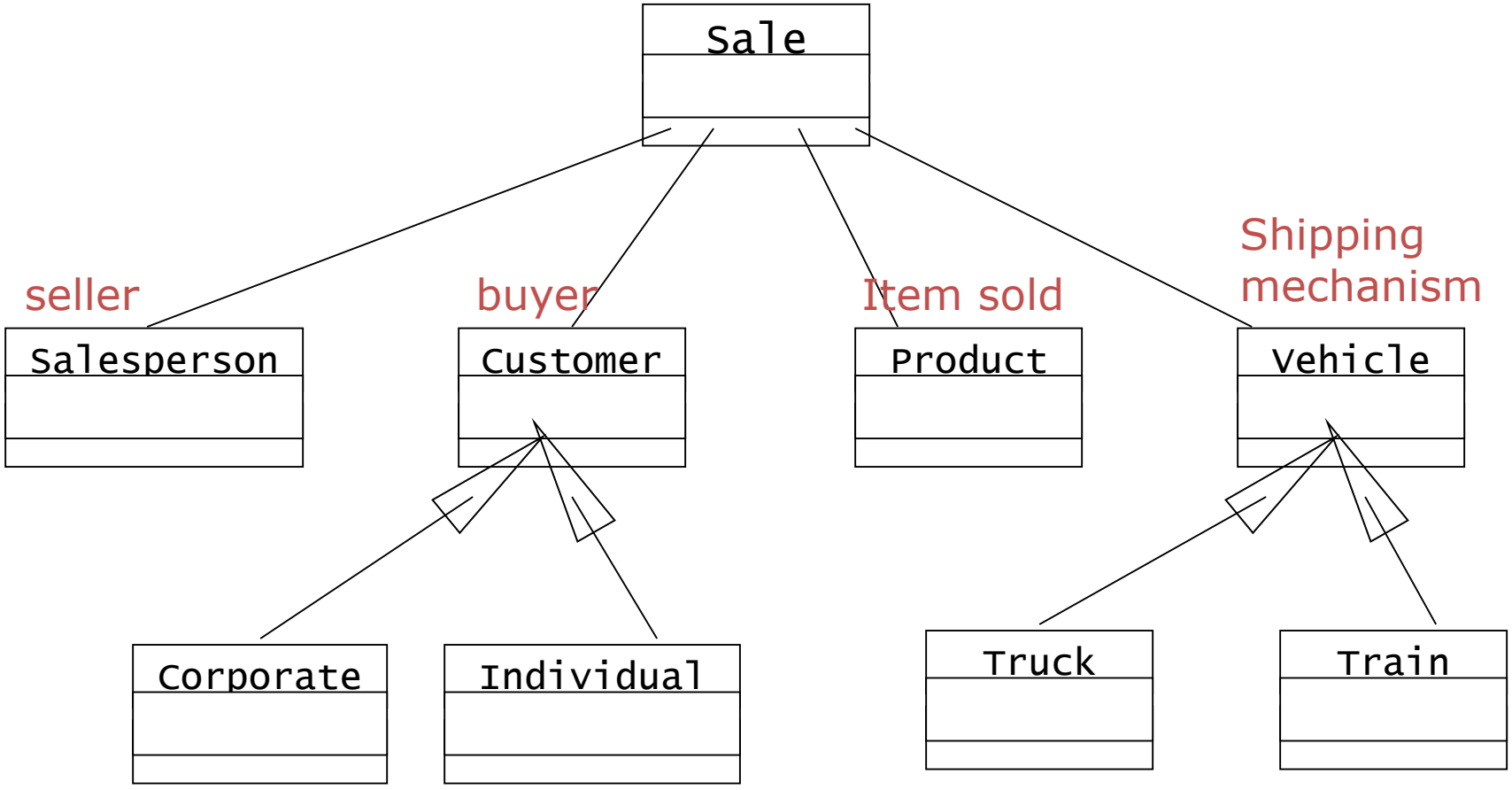
Παράδειγμα: Πωλήσεις

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