

ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Εισαγωγή

Συστάσεις

- Ποιος είμαι εγώ:
 - Παναγιώτης Τσαπάρας
 - Email: tsap@cs.uoi.gr
 - Γραφείο: B.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:
 - Δευτέρα 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 καθορισθεί από τα παρακάτω:
 - Εργαστήρια ή Πρόοδος [20%]
 - Ασκήσεις [30%]
 - Τελική Εξέταση [50%]
- Η συμμετοχή στα εργαστήρια είναι υποχρεωτική μόνο για τους πρωτοετείς.
- Όσοι έχετε ξαναπάρει το μάθημα Τεχνικές Αντικειμενοστραφούς Προγραμματισμού μπορείτε αν θέλετε να κρατήσετε το βαθμό του εργαστηρίου.
- Για όλους όσους δεν εμπίπτουν σε μία από τις παραπάνω κατηγορίες, θα πρέπει να δώσετε υποχρεωτική πρόοδο.
- Οι ασκήσεις είναι υποχρεωτικές για όλους.

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.
- Χρησιμοποιείτε τα εργαστήρια για να καταλάβετε καλύτερα.
- Σεβαστείτε τους συμφοιτητές σας και τους διδάσκοντες, μην κάνετε φασαρία.

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

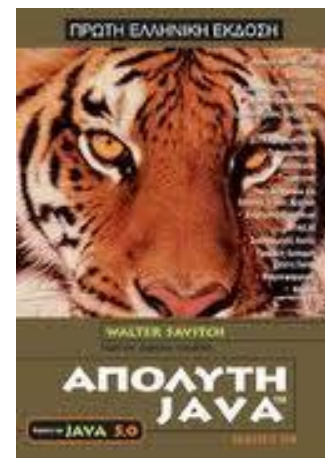
- Αρχές αντικειμενοστραφούς προγραμματισμού
 - Κλάσεις και αντικείμενα
 - Ενθυλάκωση και απόκρυψη
 - Πολυμορφισμός και Κληρονομικότητα
 - Αφηρημένες κλάσεις, Διεπαφές (Interfaces)
 - Γενικευμένες κλάσεις, συλλογές
- Εισαγωγή στη Java
 - Βασικό συντακτικό και δομή προγράμματος
 - Είσοδος, έξοδος δεδομένων
 - 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 εκτυπώσω τις διαφάνειες από περίσου.

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.
- 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

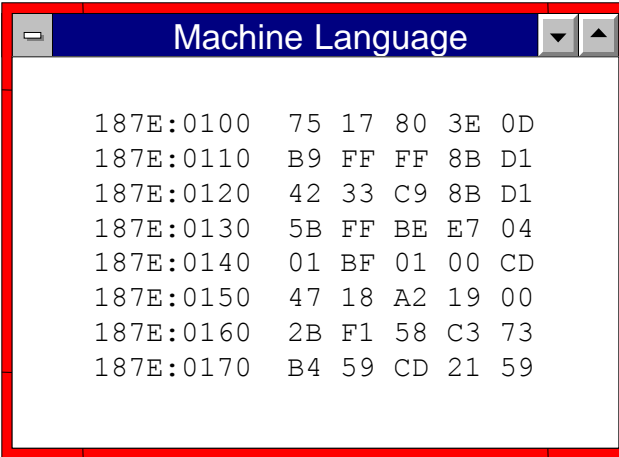
- **Πρώτη γενιά:** Γλώσσες μηχανής

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.



```
Machine Language
187E:0100  75 17 80 3E 0D
187E:0110  B9 FF FF 8B D1
187E:0120  42 33 C9 8B D1
187E:0130  5B FF BE E7 04
187E:0140  01 BF 01 00 CD
187E:0150  47 18 A2 19 00
187E:0160  2B F1 58 C3 73
187E:0170  B4 59 CD 21 59
```

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**

```
Assembly Language
POP  SI
MOV  AX, [BX+03]
SUB  AX, SI
MOV  WORD PTR [TOT_AMT], E0D7
MOV  WORD PTR [CUR_AMT], E1DB
ADD  [TOT_AMT], AX
```

Translate into machine operation codes (op-codes)

```
Machine Language
187E:0100  75 17 80 3E 0D
187E:0110  B9 FF FF 8B D1
187E:0120  42 33 C9 8B D1
187E:0130  5B FF BE E7 04
187E:0140  01 BF 01 00 CD
187E:0150  47 18 A2 19 00
187E:0160  2B F1 58 C3 73
187E:0170  B4 59 CD 21 59
```

Program executed as machine language

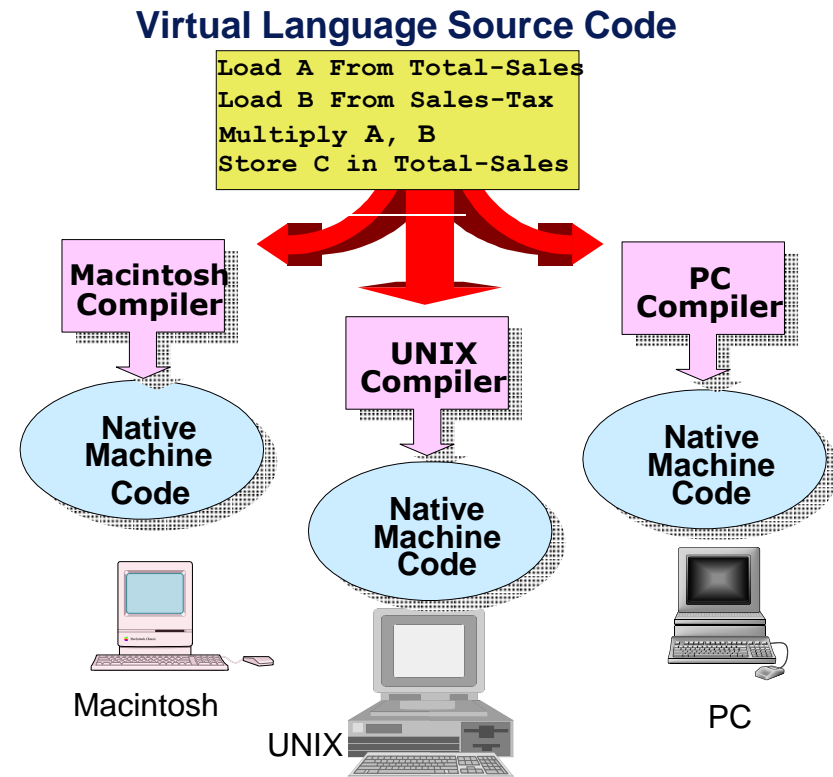
Πέντε γενεές γλωσσών προγραμματισμού

- **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

The **COMPILER** translates:
Load the purchase price
Multiply it by the sales tax
Add the purchase price to the result
Store the result in total price

```
High-Level Language
```

```
-  
salesTax = purchasePric * TAX_RATE;  
totalSales = purchasePrice + salesTax;
```

Translate into the instruction set

```
Assembly Language
```

```
POP SI  
MOV AX, [BX+03]  
SUB AX, SI  
MOV WORD PTR [TOT_AMT], E0D7  
MOV WORD PTR [CUR_AMT], E1DE  
ADD [TOT_AMT], AX
```

Translate into machine operation codes (op-codes)

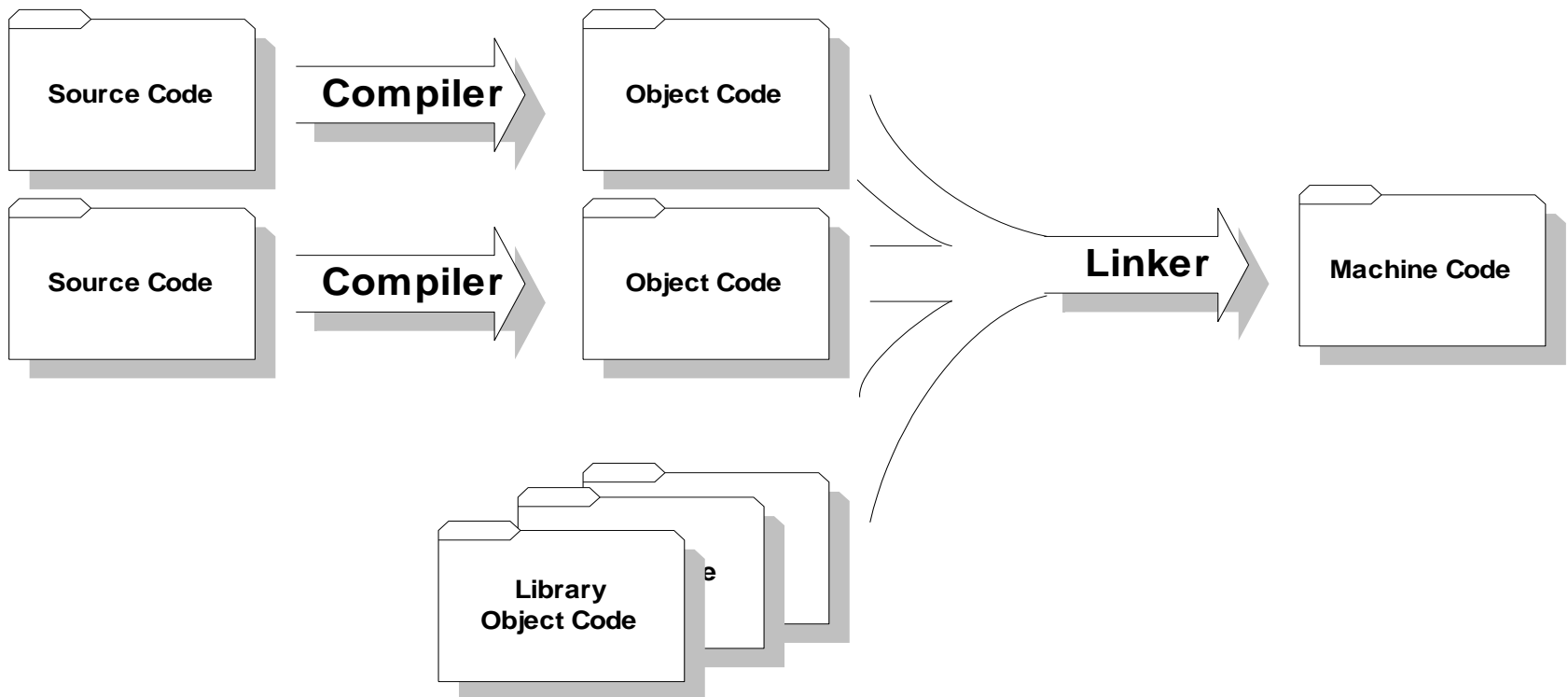
```
Machine Language
```

```
1  
87E:0100 75 17 80 3E 0D  
87E:0110 B9 FF FF 8B D1  
87E:0120 42 33 C9 8B D1  
87E:0130 5B FF BE E7 04  
87E:0140 01 BF 01 00 CD  
87E:0150 47 18 A2 19 00  
87E:0160 2B F1 58 C3 73  
87E:0170 B4 59 CD 21 59
```

Program executed as machine language

Πέντε γενεές γλωσσών προγραμματισμού

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



Πέντε γενεές γλωσσών προγραμματισμού

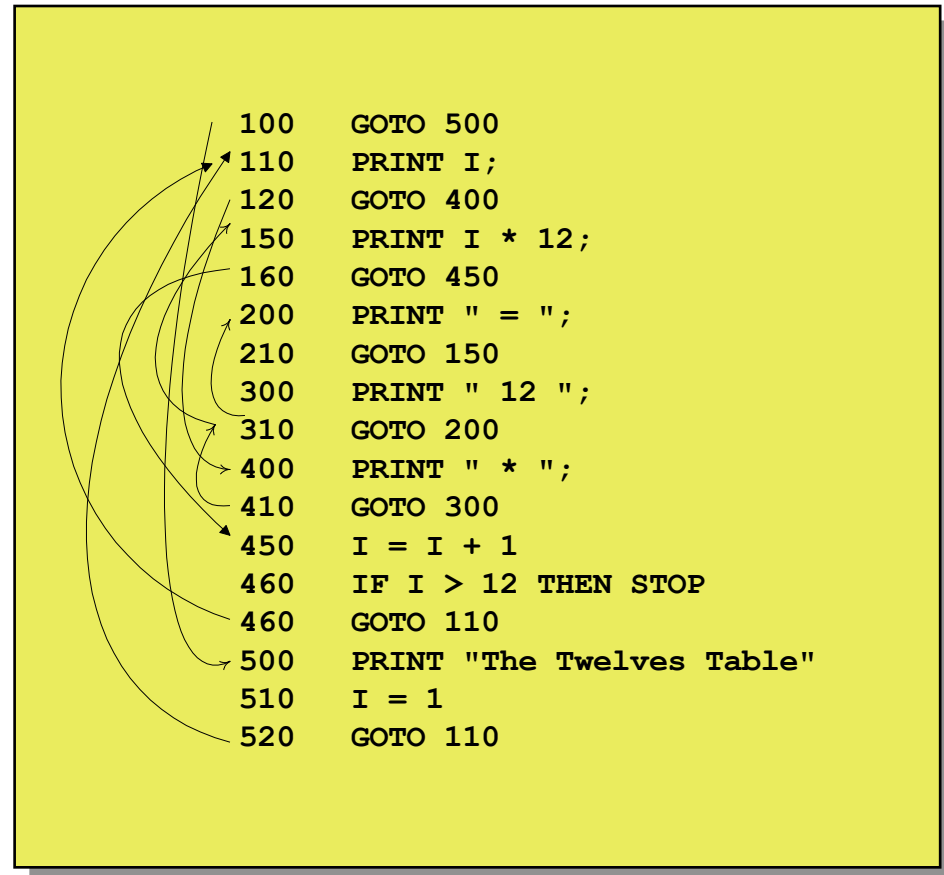
- **Πρώτη γενιά:** Γλώσσες μηχανής
 - **Δεύτερη γενιά:** Assembly
 - **Τρίτη γενιά:** Υψηλού επιπέδου (high-level) γλώσσες
 - **Τέταρτη γενιά:** Εξειδικευμένες γλώσσες
 - **Πέμπτη γενιά:** «Φυσικές» γλώσσες.
-
- Κάθε γενιά προσθέτει ένα επίπεδο **αφαίρεσης**.

Προγραμματιστικά Παραδείγματα (paradigms)

- Προγραμματισμός των πρώτων ημερών.

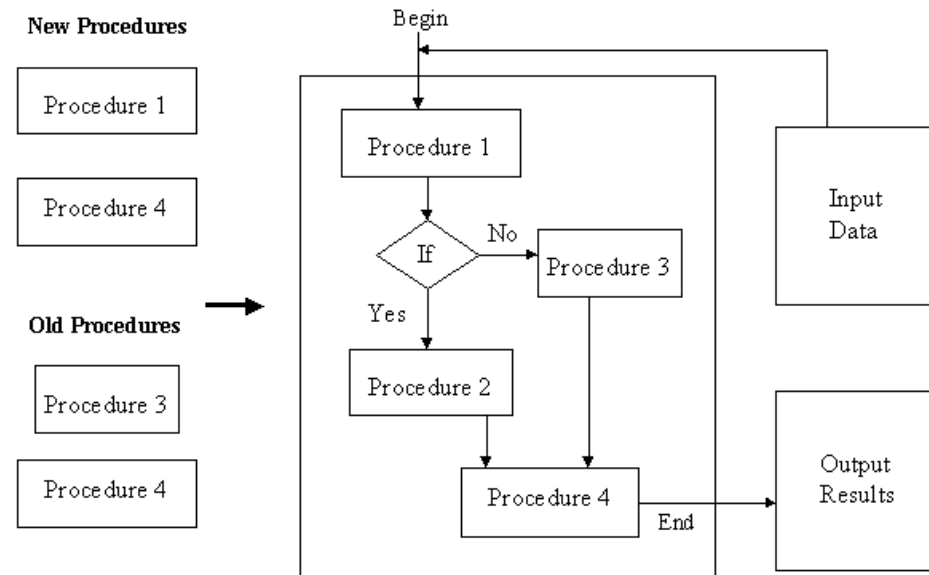
Spaghetti code

Δύσκολο να διαβαστεί και να κατανοηθεί η ροή του



Δομημένος Προγραμματισμός

- Τέσσερις 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)**

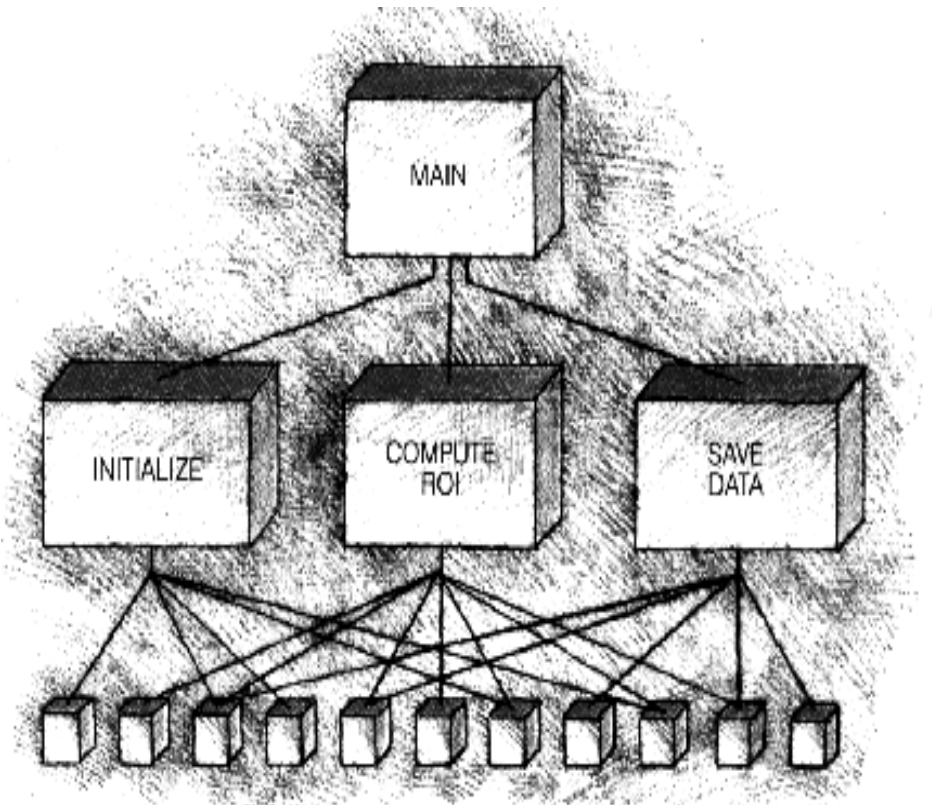


Διαδικασιακός Προγραμματισμός

- 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**)

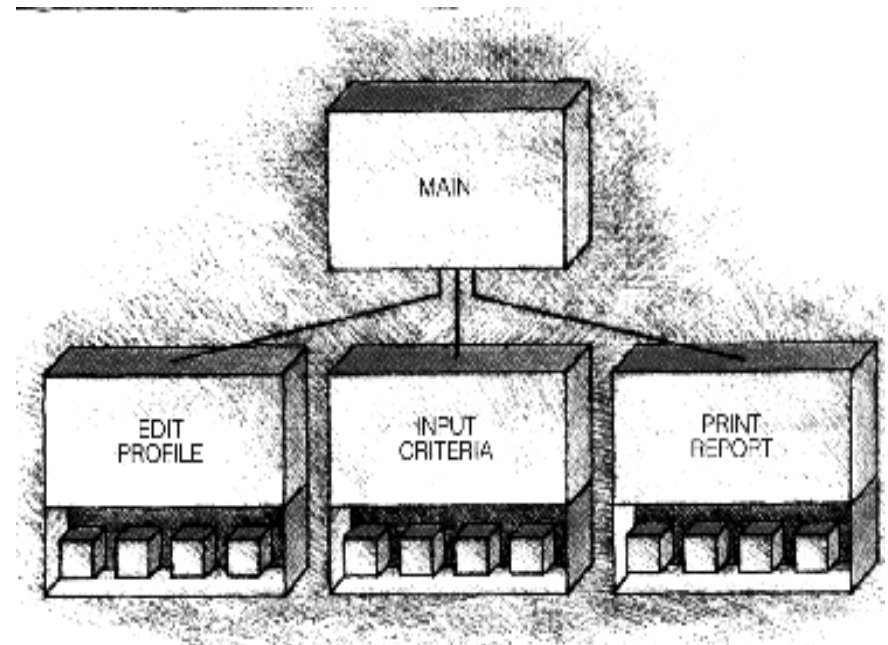
Κοινά Δεδομένα

- 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

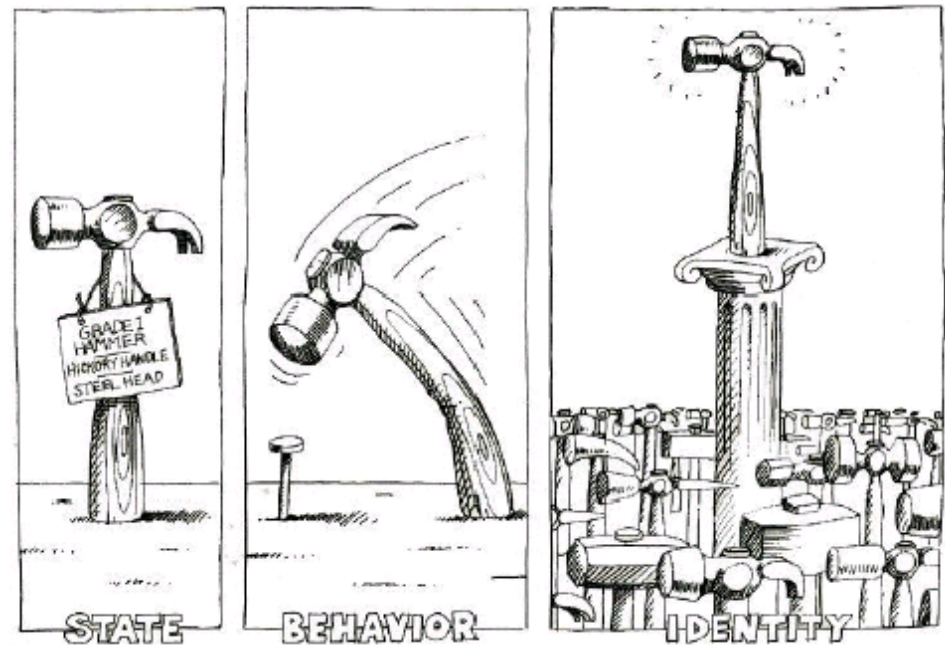
- Ο διαδικασιακός προγραμματισμός δουλεύει ΟΚ για μικρά προγράμματα, αλλά για μεγάλα συστήματα είναι δύσκολο να **σχεδιάσουμε**, να **υλοποιήσουμε** και να **συντηρήσουμε** τον κώδικα.
 - Δεν είναι εύκολο να προσαρμοστούμε σε αλλαγές, και δεν μπορούμε να προβλέψουμε όλες τις ανάγκες που θα έχουμε
- Π.χ.,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

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

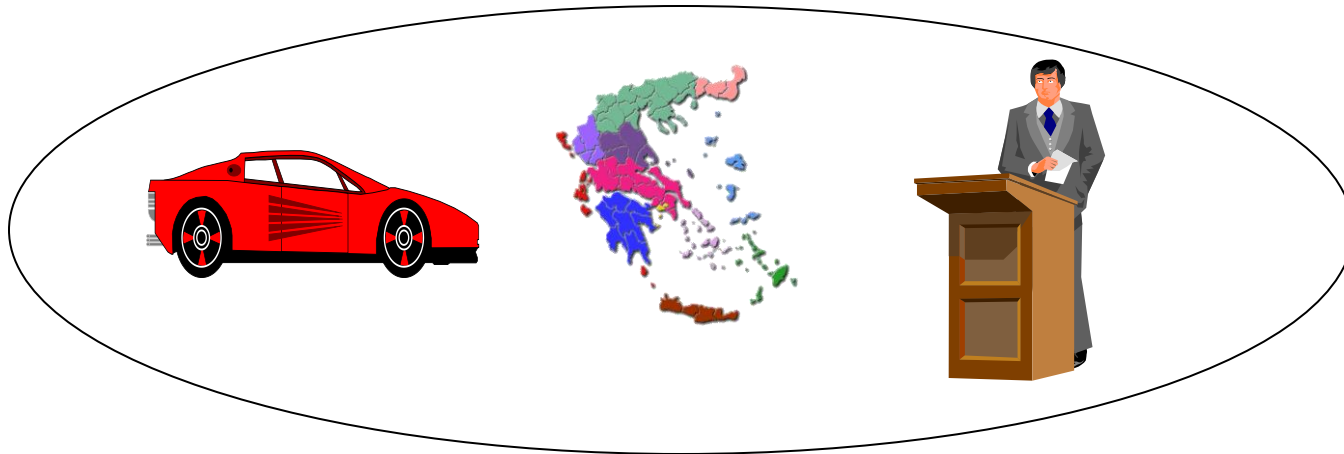
- Ένα αντικείμενο στον κώδικα αναπαριστά μια μονάδα/οντότητα/έννοια η οποία έχει:
 - Μια **κατάσταση**, η οποία ορίζεται από ορισμένα **χαρακτηριστικά**
 - Μια **συμπεριφορά**, η οποία ορίζεται από ορισμένες **ενέργειες** που μπορεί να εκτελέσει το αντικείμενο
 - Μια **ταυτότητα** που το ξεχωρίζει από τα υπόλοιπα.



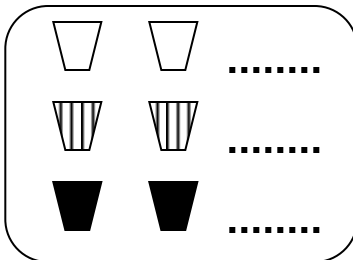
Παραδείγματα: ένας άνθρωπος, ένα πράγμα, ένα μέρος, μια υπηρεσία

Διαδικασιακή αναπαράσταση

Real world entities

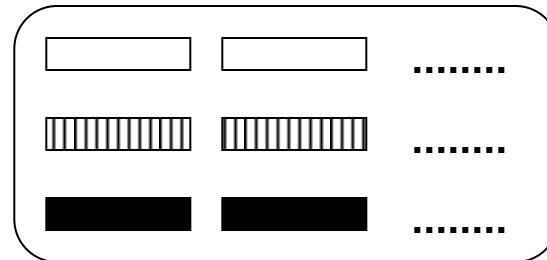


data



+

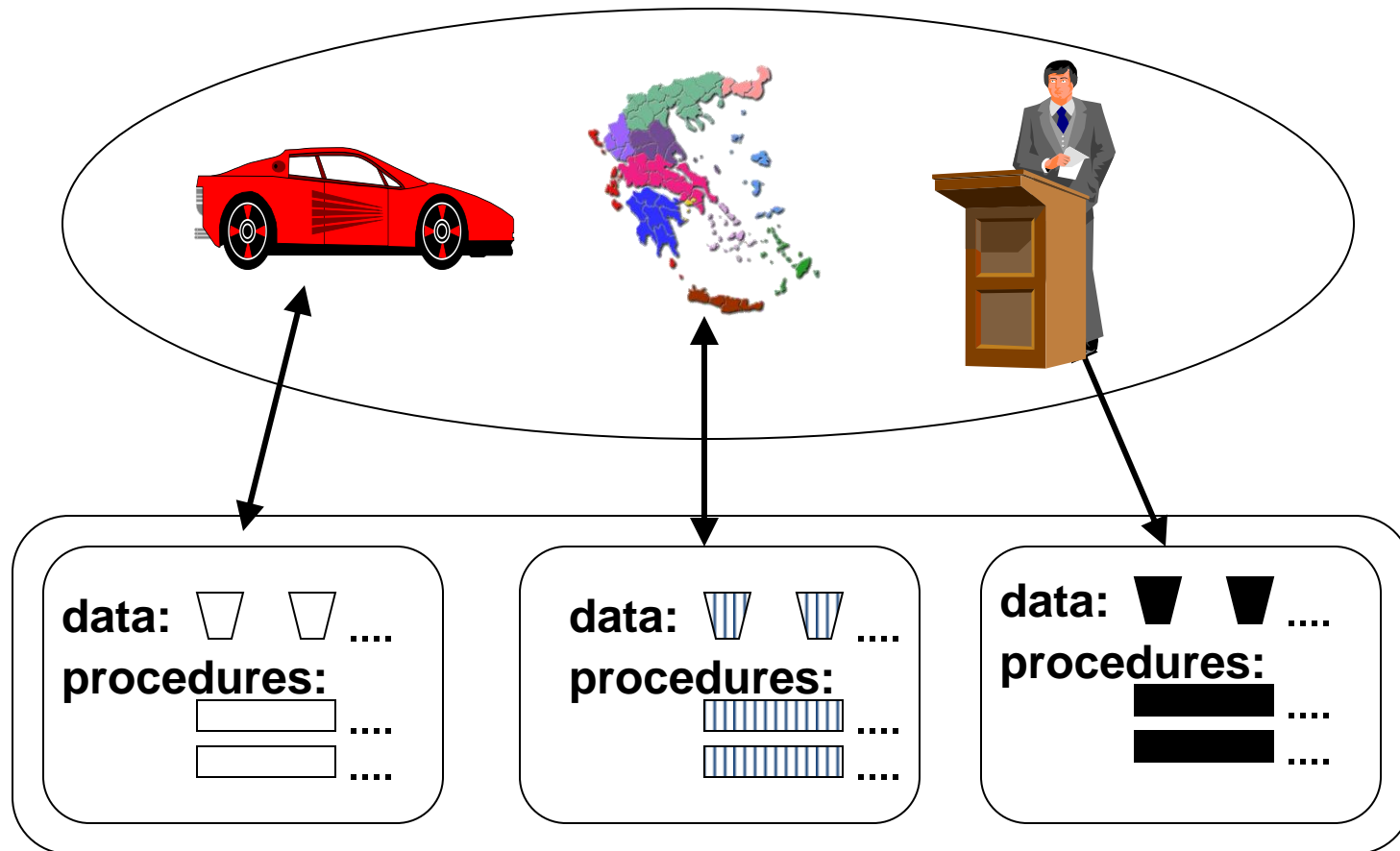
procedures



Software Representation

Αντικειμενοστραφής αναπαράσταση

Real world entities



Software Representation

Παράδειγμα

Φοιτητής Χ:

Γιώργος
10,8

print1

Καθηγητής Υ:

Κώστας
212,059

print2

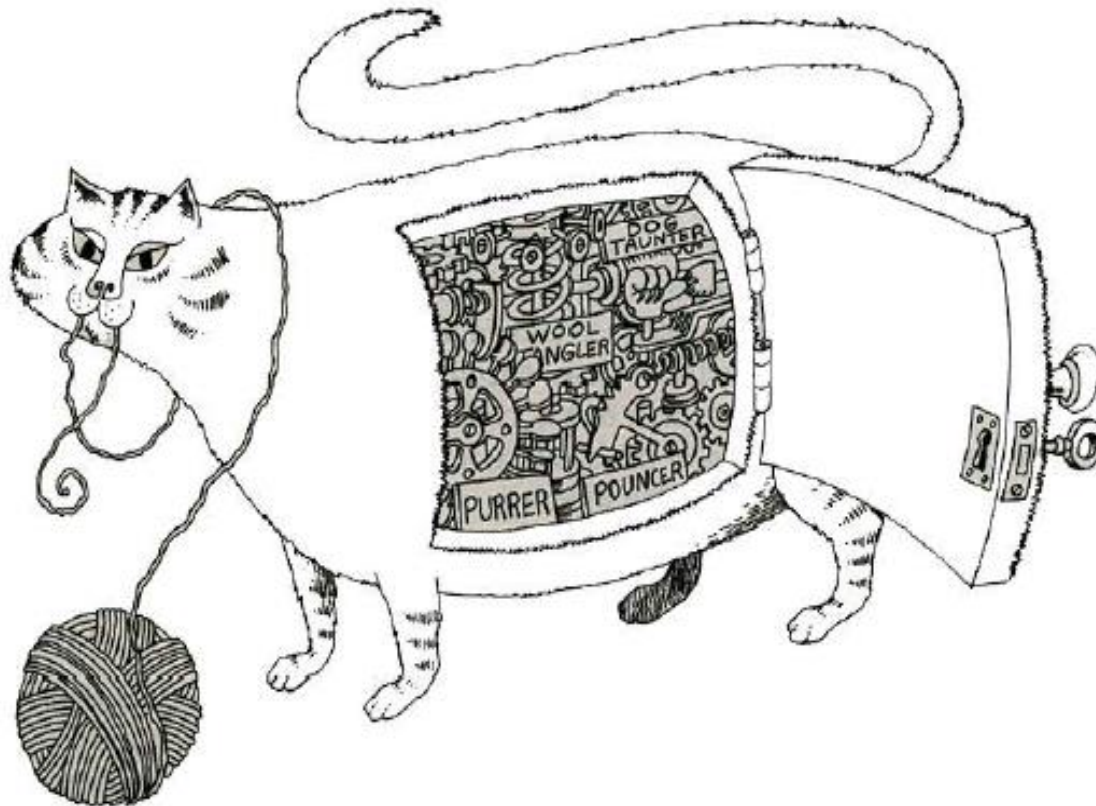
Φοιτητής Χ:

Γιώργος
10,8
print

Καθηγητής Υ:

Κώστας
212,059
print

Ενθυλάκωση



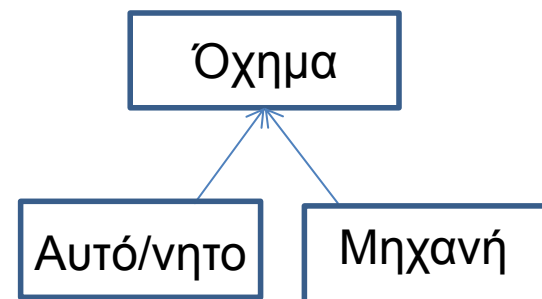
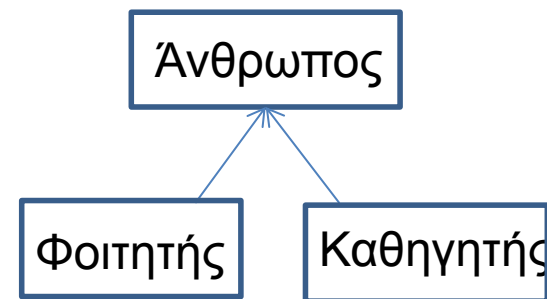
- Η στεγανοποίηση της κατάστασης και της συμπεριφοράς ώστε οι λεπτομέρειες της υλοποίησης να είναι κρυμμένες από το χρήστη του αντικειμένου.

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 X 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**)

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**.



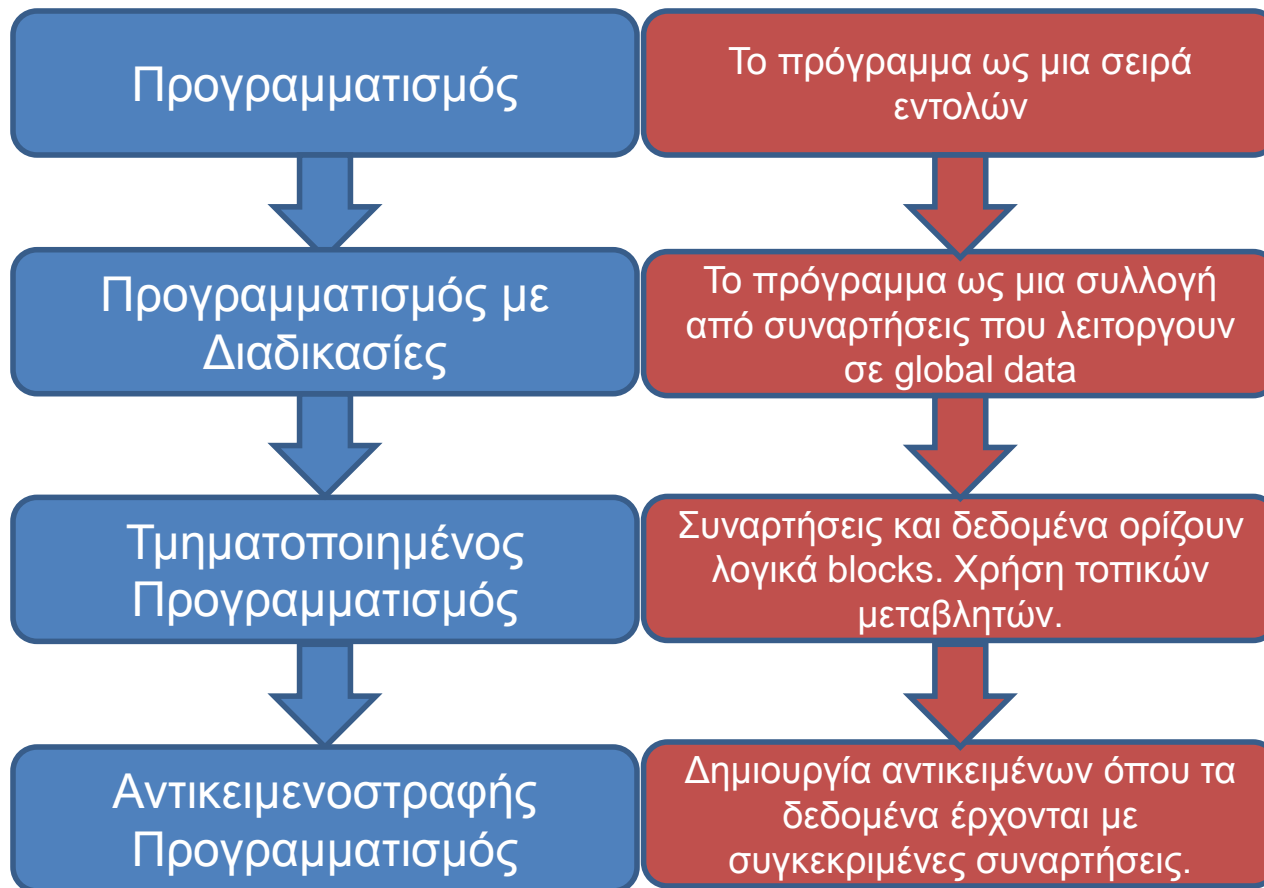
Πολυμορφισμός

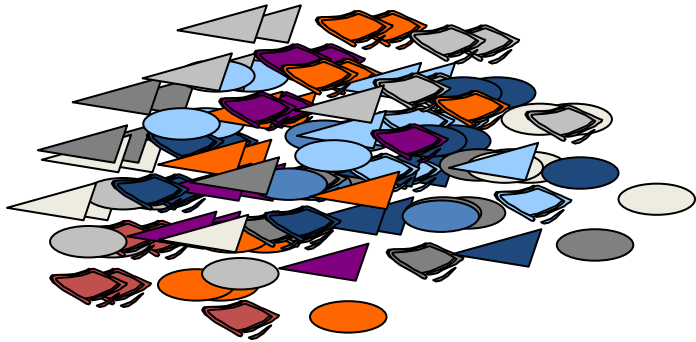
- Κλάσεις με κοινό πρόγονο έχουν κοινά χαρακτηριστικά, αλλά έχουν και διαφορές
 - Π.χ., είναι διαφορετικό το **παρκάρισμα** για ένα αυτοκίνητο και μια μηχανή
- Ο **πολυμορφισμός** μας επιτρέπει να δώσουμε μια **κοινή** συμπεριφορά σε κάθε κλάση (μια μέθοδο **park**), η οποία όμως **υλοποιείται διαφορετικά** 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

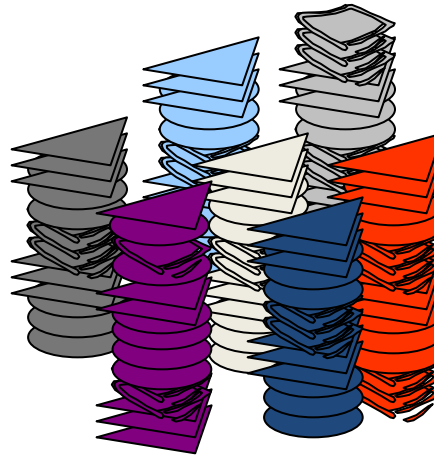
- Χρησιμοποιώντας τις κλάσεις μπορούμε να ορίσουμε τους δικούς μας **τύπους δεδομένων**
 - Έτσι μπορούμε να φτιάξουμε αντικείμενα με συγκεκριμένα χαρακτηριστικά και συμπεριφορά.
- Χρησιμοποιώντας την κληρονομικότητα και τον πολυμορφισμό, μπορούμε να **επαναχρησιμοποιήσουμε** υπάρχοντα χαρακτηριστικά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

Η εξέλιξη του προγραμματισμού

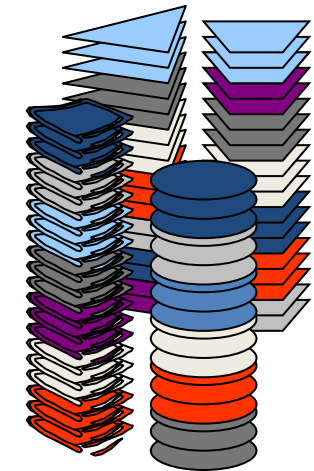




1st Generation
Spaghetti-Code



2nd & 3rd Generation :
functional decomposition

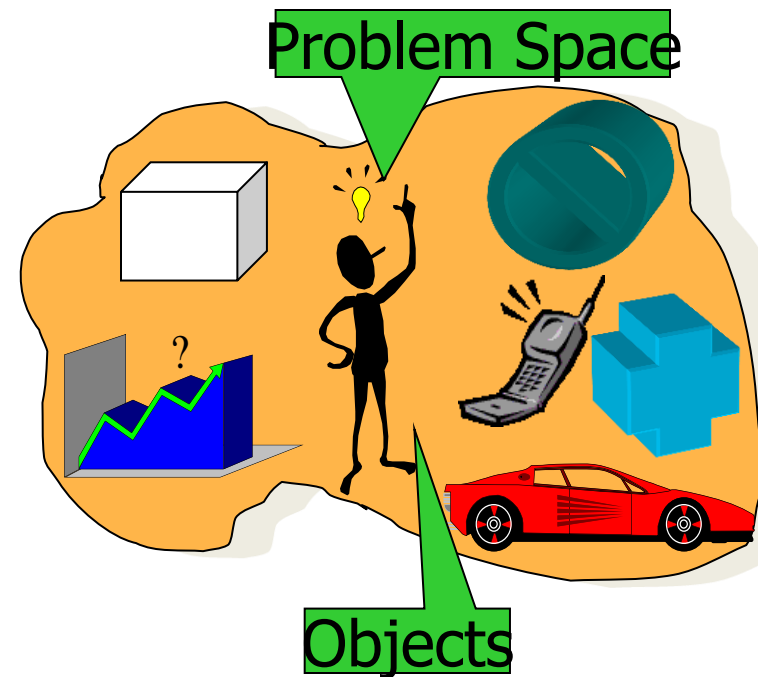


4th Generation
object decomposition

Software =
Data (Shapes)
+
Functions (Colors)

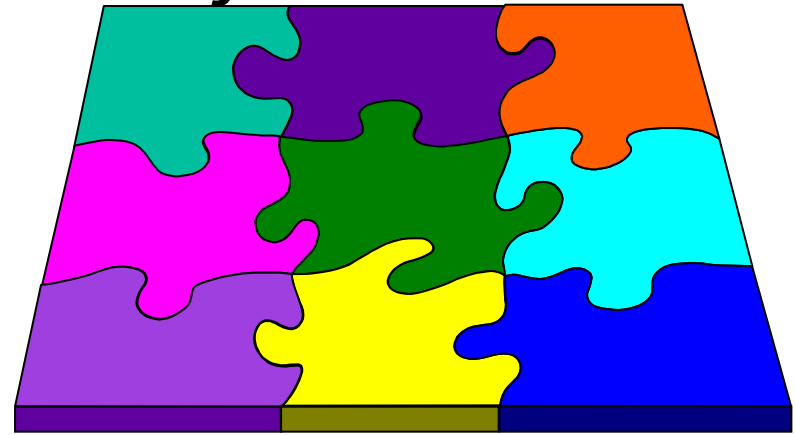
Διαδικασιακός vs. Αντικειμενοστραφής Προγραμματισμός

- **Διαδικασιακός:** 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

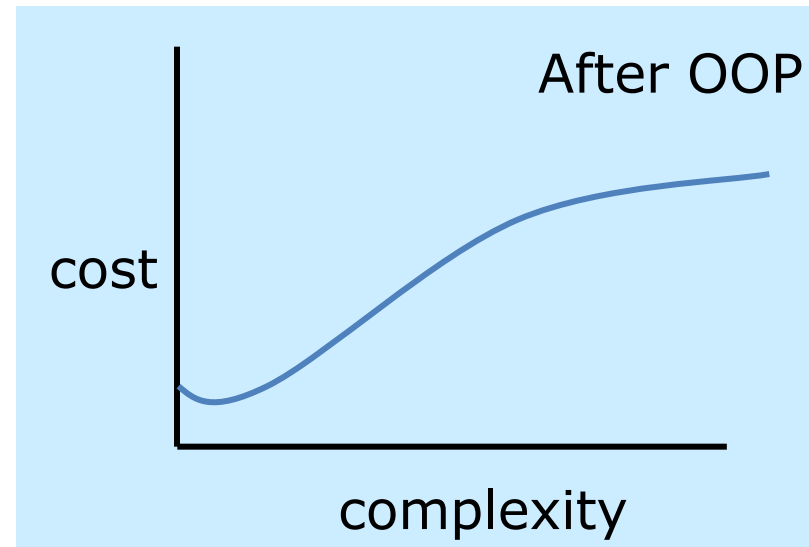
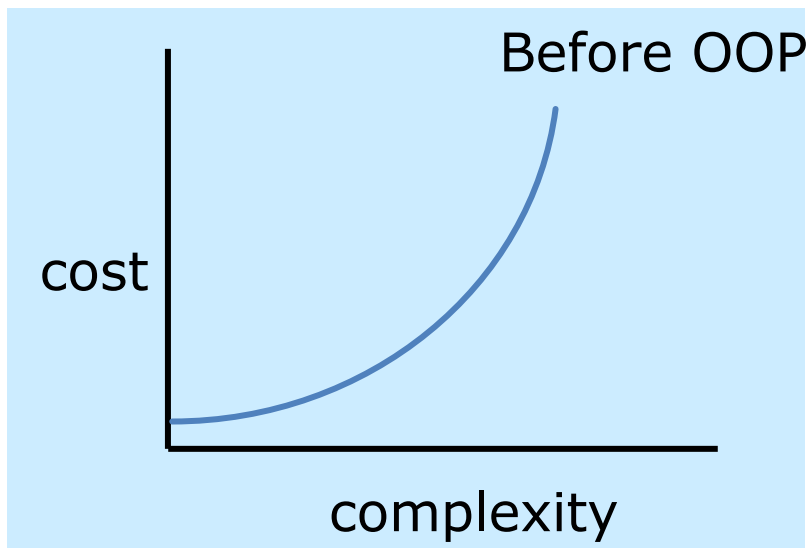


Πλεονεκτήματα αντικειμενοστραφούς προγραμματισμού

- 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



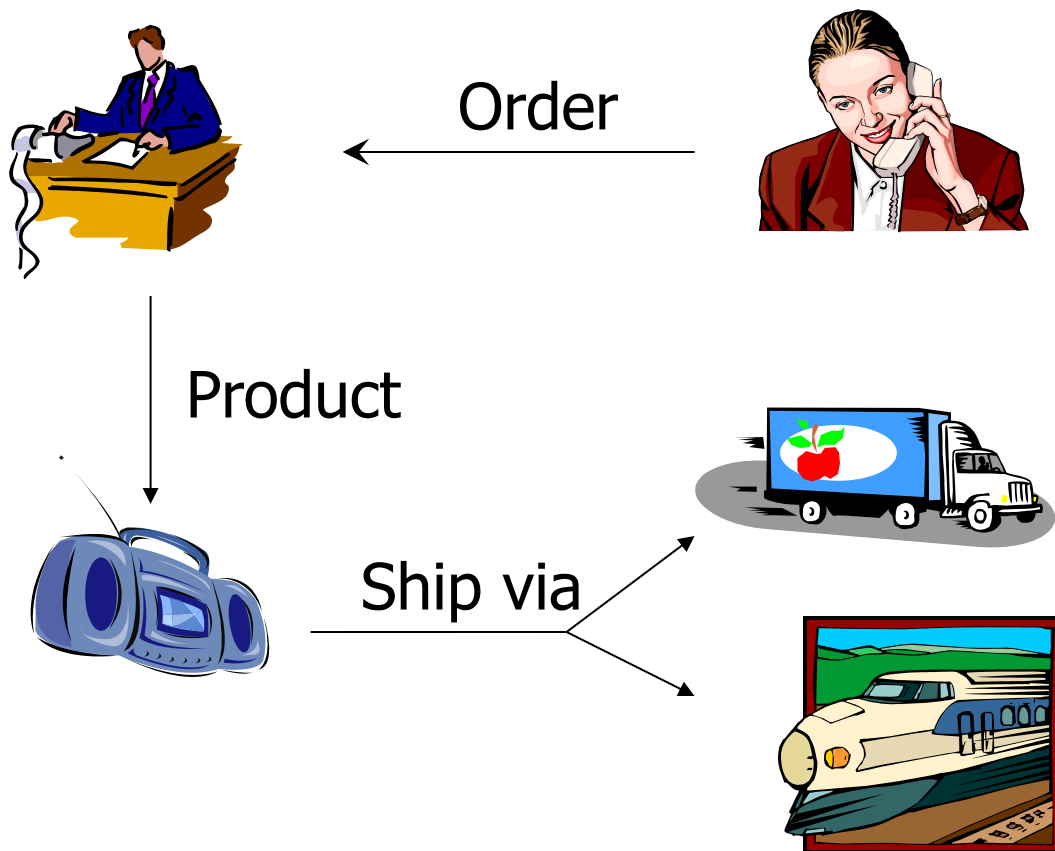
Παραγωγικότητα



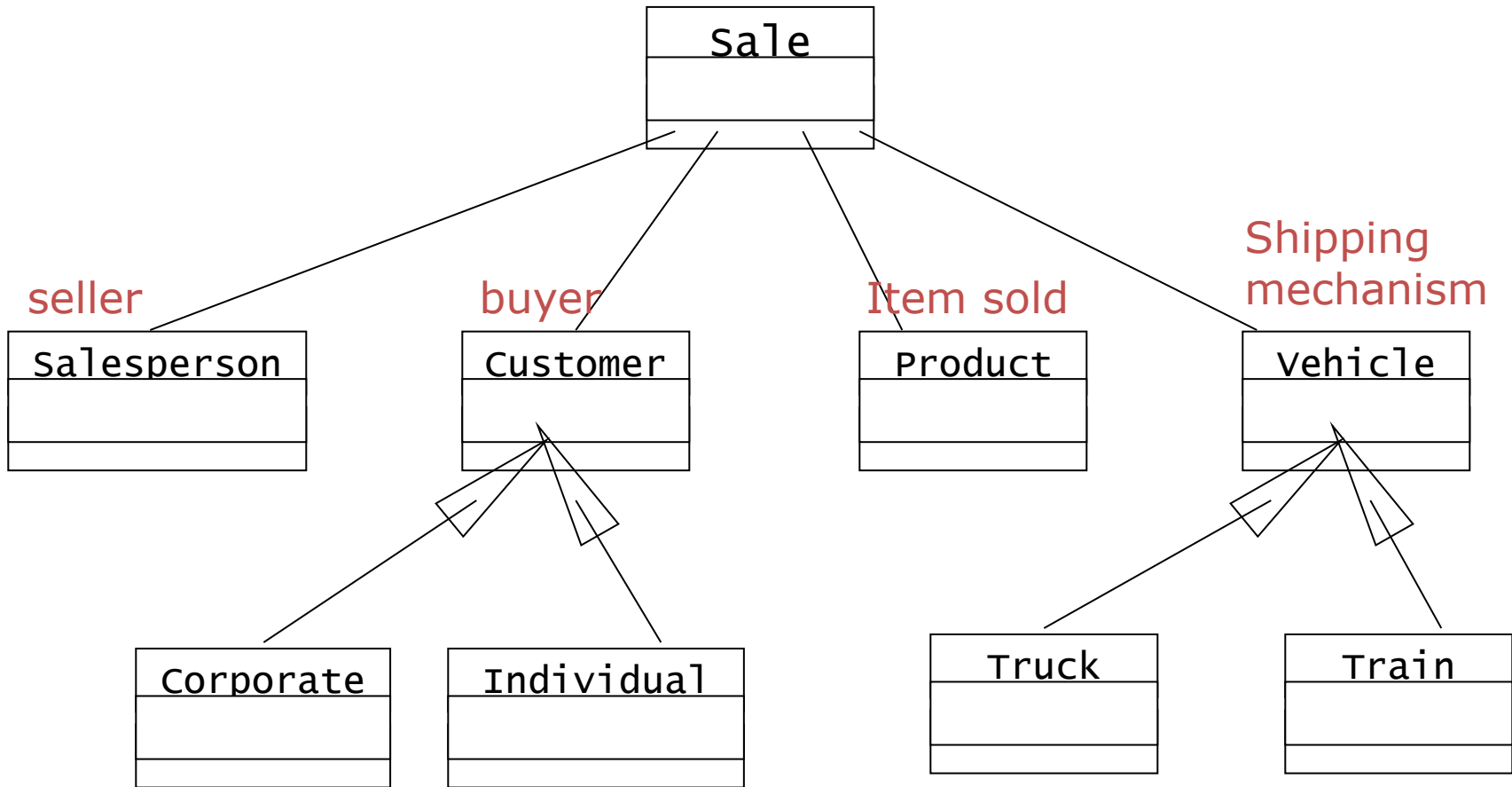
Παράδειγμα: Πωλήσεις

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

