Προχωρημένος Προγραμματισμός

Γνωριμία με το μάθημα

ΕΛΕΥΘΕΡΙΟΣ ΚΟΣΜΑΣ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2022-2023 | ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Περίληψη

Σήμερα ...

- Θα περιγράψουμε τους στόχους για αυτό το μάθημα
- Θα παρουσιάσουμε ένα συνοπτικό σχέδιο των διαλέξεων του μαθήματος

- Θα εισαγάγουμε τη βασική έννοια του προγραμματισμού
- Θα δώσουμε παραδείγματα τομέων στους οποίους το λογισμικό είναι κρίσιμο για τον πολιτισμό μας

- Θα παρουσιάσουμε συνοπτικά την εξέλιξη των γλωσσών προγραμματισμού
- Θα ορίσουμε τον αντικειμενοστραφή προγραμματισμό

- Θα κάνουμε μία εισαγωγή στη Java

Στόχοι Μαθήματος

- Εξοικείωση
  - βασικές έννοιες και τενχικές Αντικειμενοστραφούς Προγραμματισμού
    - ☞Object -Oriented Programming
  - γλώσσα προγραμματισμού Java

- Απόκτηση πρακτικής εμπειρίας
  - δημιουργία διαδραστικών και διαδικτυακών εφαρμογών

Στόχοι Μαθήματος ΙΙ

- ✓ μετά το μάθημα, θα είστε σε θέση να
  - ✓ γράψετε μικρά προγράμματα Java
  - ✓ διαβάσετε πολύ μεγαλύτερα προγράμματα
  - ✓ μάθετε τα βασικά πολλών άλλων γλωσσών από μόνοι σας
  - ✓ από την περιγραφή ενός προβλήματος, να καταστρώνετε τον αλγόριθμο/λογική επίλυσής του (σύμφωνα με το πρότυπο του αντικειμενοστραφούς προγραμματισμού) και να τον υλοποιείτε σε Java
  - ✓ δοκιμάζετε, τεκμηριώνετε και να προετοιμάζετε ένα επαγγελματικό πρόγραμμα χρησιμοποιώντας
    - ✓ τα αντίστοιχα εργαλεία της Java: assertions, javadoc, packages
    - ✓ και ολοκληρωμένα περιβάλλοντα ανάπτυξης λογισμικού

- ✗ μετά το μάθημα, δε θα είστε (ακόμα)
  - ✗ ένας έμπειρος προγραμματιστής
  - ✗ έμπειρος με τη Java

Τα μέσα...

- Διαλέξεις
  - παρακολουθείστε όλες τις διαλέξεις!
- Εργαστήρια
  - παρακολουθείστε όλα τα εργαστήρια!
    - όχι ότι σας δίνεται και άλλη επιλογή ... ☺
  - ολοκληρώστε όλες τις ασκήσεις που συζητούμε στα εργαστήρια
    - θα πρέπει να τις παραδώσετε έως το τέλος της εκάστοτε εβδομάδας

Τα μέσα... II

- Εργασία
  - καταγράψτε τους (νέους) όρους που συζητάμε
  - ασκήσεις
  - project
    - το πιο ευχάριστο κομμάτι του μαθήματος
    - εδώ θα μάθετε τα περισσότερα
    - χαρείτε το!

Συνεργαστείτε στη μάθηση

επιπρόσθετα της ατομικής σας εργασίας, ενθαρρύνεστε να συνεργαστείτε και να βοηθήσετε ο ένας τον άλλον

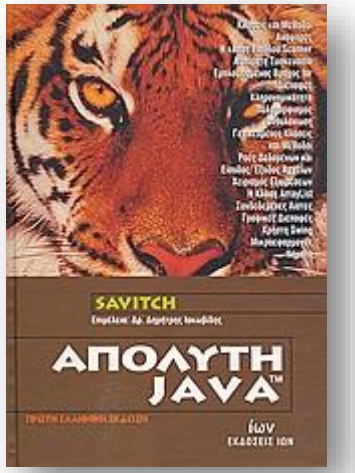
- σε περίπτωση αμφιβολίας εάν μια συνεργασία είναι νόμιμη: ρωτήστε!
  - μην ισχυρίζεστε ότι έχετε γράψει κώδικα που αντιγράψατε από άλλους
  - μη δίνετε σε κανέναν τον κώδικά σας
  - όταν βασίζεστε στη δουλειά κάποιου άλλου, αναφέρετε ρητά όλες τις πηγές σας
    - τιμήστε εκείνους που έκαναν το έργο
- μη μελετάτε μόνοι σας όταν δε χρειάζεται
  - συγκροτήστε ομάδες μελέτης
  - βοηθήστε ο ένας τον άλλον (χωρίς να κάνετε λογοκλοπή)
- εκμεταλευτείτε τις ώρες γραφείου του καθηγητή
  - πηγαίνετε προετοιμασμένοι με ερωτήσεις
    - τα μόνα ανόητα ερωτήματα είναι αυτά που θέλατε να ρωτήσετε, αλλά δεν το κάνατε

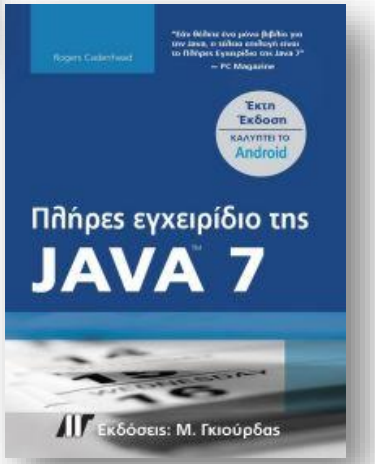
Αξιολόγηση

- **Εργαστήριο**
  - Ερωτήσεις πολλαπλής επιλογής (+5% bonus στον τελικό βαθμό)
- **Προγραμματιστικές Ασκήσεις**
  - 2-3 σειρές ασκήσεων
  - προφορική εξέταση
- **Project**
  - προφορική εξέταση
- **Τελική εξέταση**
  - προαιρετική **ενδιάμεση** εξέταση (→ συμμετέχει στην τελική βαθμολογία μόνο εάν τη βελτιώνει)
- **Παρακολούθηση Διαλέξεων**
  - +5% bonus στον τελικό βαθμό
  - το πολύ 2 απουσίες

Στοιχεία για το μάθημα

- Διαλέξεις
  - Θεωρία: Τρίτη, 17:00 – 20:00 @ Αίθουσα Β5
  - Εργαστήριο: Πέμπτη, 19:00 – 21:00 @ Εργαστήριο 3
- Ώρες Γραφείου
  - όποτε με χρειαστείστε στείλτε μου e-mail!
  - βολικές μέρες/ώρες Τρίτη και Παρασκευή, μετά τα μαθήματα
- Διδακτικό Υλικό
  - Διαφάνειες
  - Βιβλία
    - Savitch Walter, "Απόλυτη Java", ΣΤΕΛΛΑ ΠΑΡΙΚΟΥ & ΣΙΑ ΟΕ, 1η έκδοση, 2008.
    - Rogers Cadenhead, "Πλήρες Εγχειρίδιο της Java 7", Εκδόσεις Γκιούρδας, 2013.
- Ιστοσελίδα Μαθήματος: https://eclass.teicrete.gr/courses/TH142/
- e-mail καθηγητή: ekosmas@hmu.gr





Προγραμματισμός

Εισαγωγή

Προγραμματισμός ...

η τέχνη της έκφρασης λύσεων για προγράμματα έτσι ώστε ένας υπολογιστής να μπορεί να εκτελέσει αυτές τις λύσεις

- εξεύρεση και βελτίωση λύσεων
- καλύτερη κατανόηση του προβλήματος μέσω της διαδικασίας προγραμματισμού

Προγραμματισμός ... II

- είναι κάτι που μαθαίνεται γράφοντας προγράμματα

- μία μοναχική διαδικασία; → όχι!
  - συμμετοχή σε ομάδα με κοινό στόχο → εργαζόμαστε καλύτερα, μαθαίνουμε πιο γρήγορα
  - η ομαδική εργασία και η συζήτηση των προβλημάτων με φίλους
    - δεν είναι αντιγραφή
    - μας αναγκάζει να αρθρώνουμε τις ιδέες μας → βελτίωση κατανόησης και μνήμης

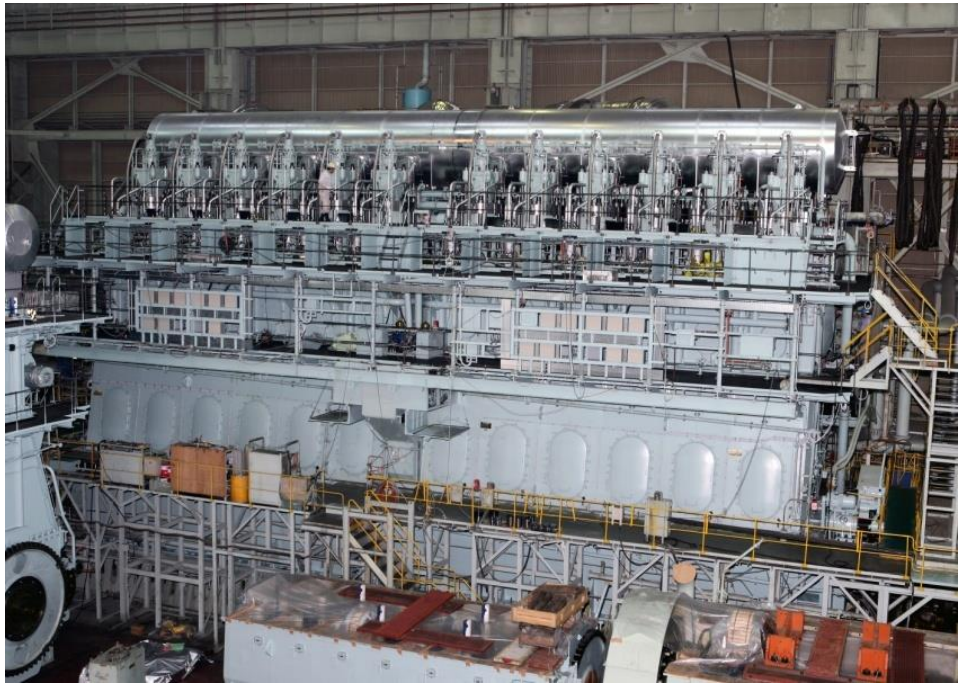
Γιατί να παρακολουθήσω το μάθημα;
Γιατί προγραμματισμός;

- ο πολιτισμός μας βασίζεται στο λογισμικό
  - εάν δεν κατανοώ το λογισμικό → νόμιζω ότι όλα γίνονται «μαγικά» → αποκλείομαι από τους πιο ενδιαφέροντες, κερδοφόρους και κοινωνικά χρήσιμους τεχνικούς τομείς εργασίας
  - προγράμματα υπολογιστών
    - εφαρμογές PC με GUI
    - υπολογισμούς και εφαρμογές ελέγχου σε διάφορα συστήματα
      - π.χ. κινητά, ψηφιακές φωτογραφικές μηχανές, αυτοκίνητα
    - εφαρμογές επεξεργασίας κειμένου
- πολύτιμη πνευματική άσκηση → οξύνει την ικανότητά μας να σκεφτόμαστε
- ένας τρόπο να αλλάξουμε τον κόσμο προς το καλύτερο

Γιατί Προγραμματισμός;

- σημείωση: τα περισσότερα προγράμματα δεν τρέχουν σε πράγματα που μοιάζουν με PC
  - μια οθόνη, ένα πληκτρολόγιο, ένα κουτί κάτω από το τραπέζι

Γιατί Προγραμματισμός;
Πλοία



- Σχεδίαση (σκελετού πλοίου)
- Κατασκευή (ναυπηγείο → ψηφιακό)
- Μηχανή (ηλεκτρονική ανάφλεξη)

- Παρακολούθηση (αυτονομία, gps)
- Διαχείριση (χρονοδιάγραμμα)

Γιατί Προγραμματισμός;
Αεροπλάνα

- Επικοινωνία
- Έλεγχος
- Κατασκευή

- Διαχείριση

Γιατί Προγραμματισμός;
Τηλεπικοινωνίες

- Ποιότητα φωνής
- Διεπαφές
- Χρεώσεις
- Φορητότητα

- Αξιοπιστία
- Μεταγωγή
- Εικόνες

Γιατί Προγραμματισμός;

Ενέργεια

- Έλεγχος
- Διαχείριση
- Ανάλυση
- Σχεδίαση

- Επικοινωνίες
- Οπτικοποίηση

Γιατί Προγραμματισμός;
PC, tablet, worksation

- Υπάρχουν πολλά περισσότερα στους υπολογιστές από παιχνίδια, επεξεργασία κειμένου, περιήγηση και υπολογιστικά φύλλα!

Ύλη μαθήματος

1. Αρχές αντικειμενοστραφούς προγραμματισμού
   - Κλάσεις και αντικείμενα
   - Ενθυλάκωση και απόκρυψη
   - Πολυμορφισμός και Κληρονομικότητα
   - Αφηρημένες κλάσεις, Διεπαφές (Interfaces)
   - Γενικευμένες κλάσεις, συλλογές
2. Εισαγωγή στη Java
   - Βασικό συντακτικό και δομή προγράμματος
   - Είσοδος, έξοδος δεδομένων
   - Εξαιρέσεις
   - Γραφικά/Μικροεφαρμογές
3. Java Applets
4. Java Threads
5. Android

Υποσχέσεις

- Λεπτομέρειες: θα προσπαθήσουμε να εξηγήσουμε κάθε εργαλείο που χρησιμοποιείται σε αυτό το μάθημα με αρκετή λεπτομέρεια για πραγματική κατανόηση
  - δεν υπάρχει «μαγεία»
- Χρησιμότητα: θα προσπαθήσουμε να εξηγήσουμε μόνο χρήσιμες έννοιες, εργαλεία και τεχνικές
  - δε θα προσπαθήσουμε να εξηγήσουμε κάθε λεπτομέρεια
- Πληρότητα: Οι έννοιες, εργαλεία και τεχνικές μπορούν να χρησιμοποιηθούν σε συνδυασμό για την κατασκευή χρήσιμων προγραμμάτων
  - υπάρχουν, βεβαίως, πολλές χρήσιμες έννοιες, εργαλεία και τεχνικές πέρα από αυτά που διδάσκονται εδώ

Ευχαριστίες

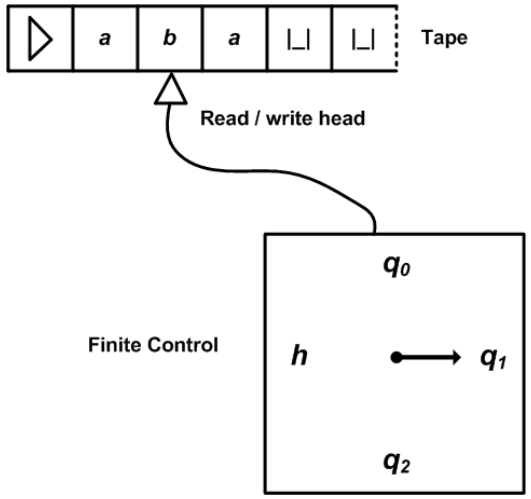
- το υλικό (διαφάνειες, εργασίες) που θα χρησιμοποιήσουμε στο μάθημα έχει παραχθεί βασιζόμενο
  - στο αντίστοιχο υλικό που έχει παραχθεί από τους καθηγητές
    - Παναγιώτη Τσαπάρα
    - Βασίλη Χριστοφίδη
  - και στη βιβλιογραφία του μαθήματος

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σύντομη αναφορά

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



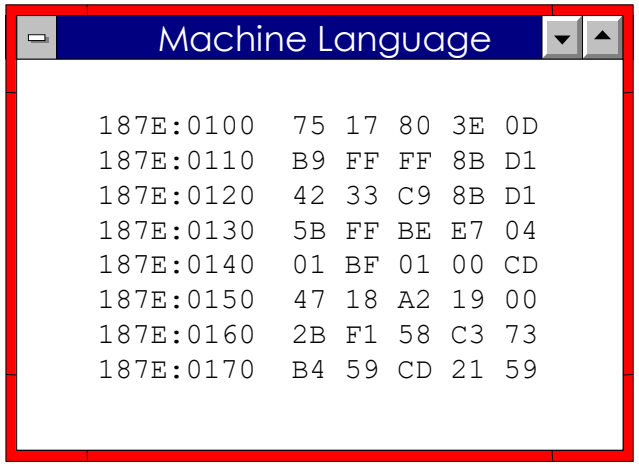
Γλώσσες Προγραμματισμού

1η γενιά: Γλώσσες Μηχανής

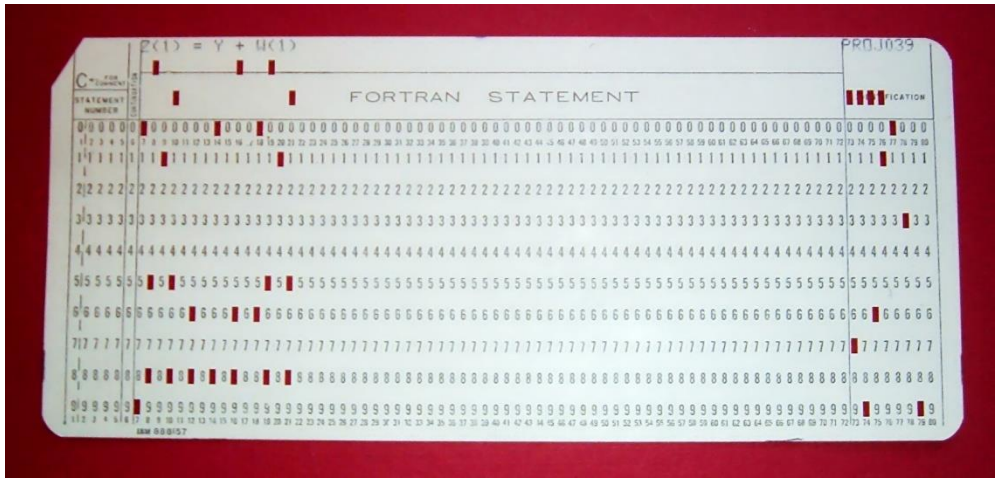
ο προγραμματιστής

1. μετατρέπει το πρόβλημα του σε ένα πρόγραμμα
   - π.χ. υπολογισμός μέγιστου κοινού διαιρέτη δύο αριθμών
2. γράφει ακριβώς τις εντολές που θα πρέπει να εκτελέσει ο υπολογιστής
   - πρέπει να γνωρίζει ακριβώς τη δυαδική αναπαράσταση των εντολών

☞ στους πρώτους υπολογιστές οι εντολές κωδικοποιούνταν σε διάτρητες κάρτες



Program entered and executed as machine language

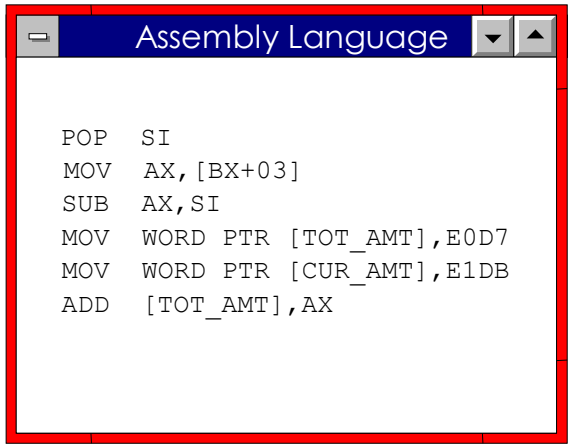


Punched card from a Fortran program: Z(1) = Y + W(1)

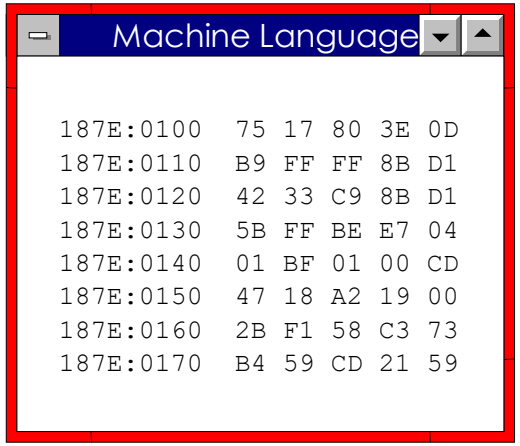
Γλώσσες Προγραμματισμού

2η γενιά: Συμβολική Γλώσσα (Assembly)

- ο προγραμματιστής
  - δε χρειάζεται να ξέρει ακριβώς την δυαδική αναπαράσταση των εντολών
  - χρησιμοποιεί πιο κατανοητούς μνημονικούς κανόνες
- ο συμβολομεταφραστής (assembler) μετατρέπει/μεταφράζει τα σύμβολα σε γλώσσα μηχανής (op-codes)
- οι γλώσσες εξαρτώνται από το hardware



program **entered** as **assembly** language

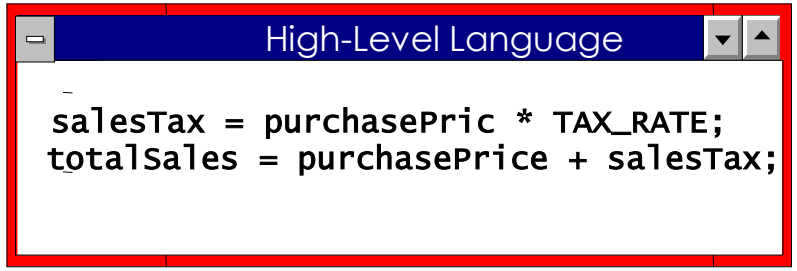


program is **translated** and **executed** as **machine** language

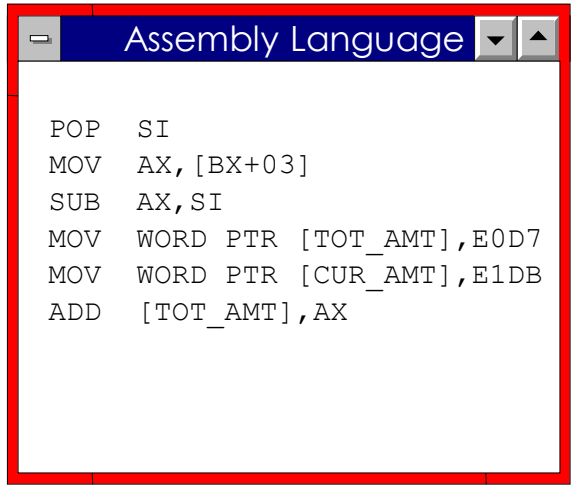
Γλώσσες Προγραμματισμού

3η γενιά: Υψηλού επιπέδου (high-level)

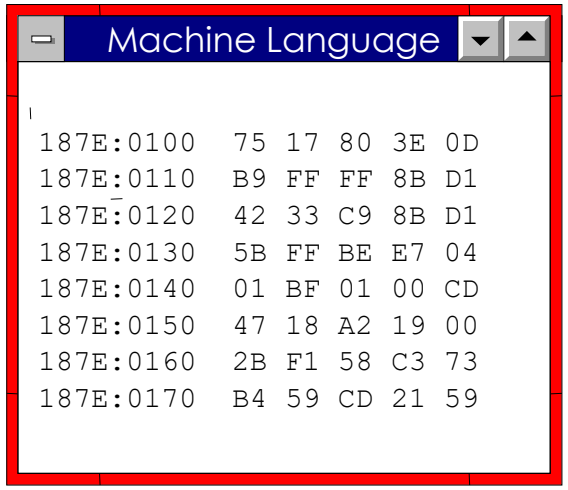
- ο προγραμματιστής δίνει εντολές στον υπολογιστή σε μια κατανοητή και καλά δομημένη γλώσσα (source code)
- ο μεταγλωτιστής (compiler) τις μετατρέπει σε ενδιάμεσο κώδικα (object code)
- ο ενδιάμεσος κώδικας μετατρέπεται σε γλώσσα μηχανής (machine code)



program **entered** as **high-level** language



program is **translated** into the **instruction set**

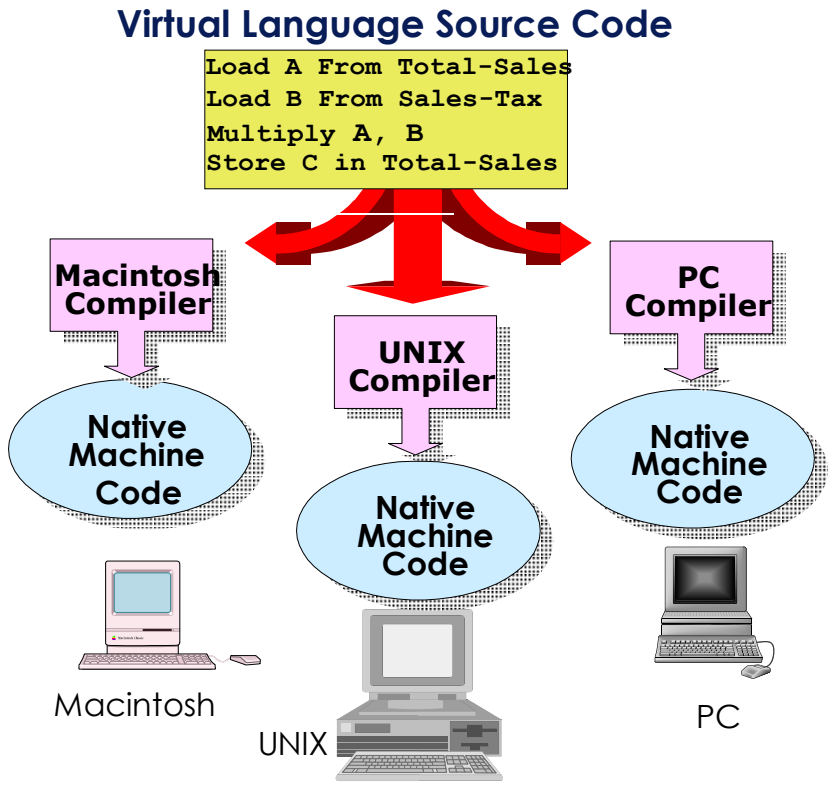


program is translated and **executed** as **machine** language

Γλώσσες Προγραμματισμού

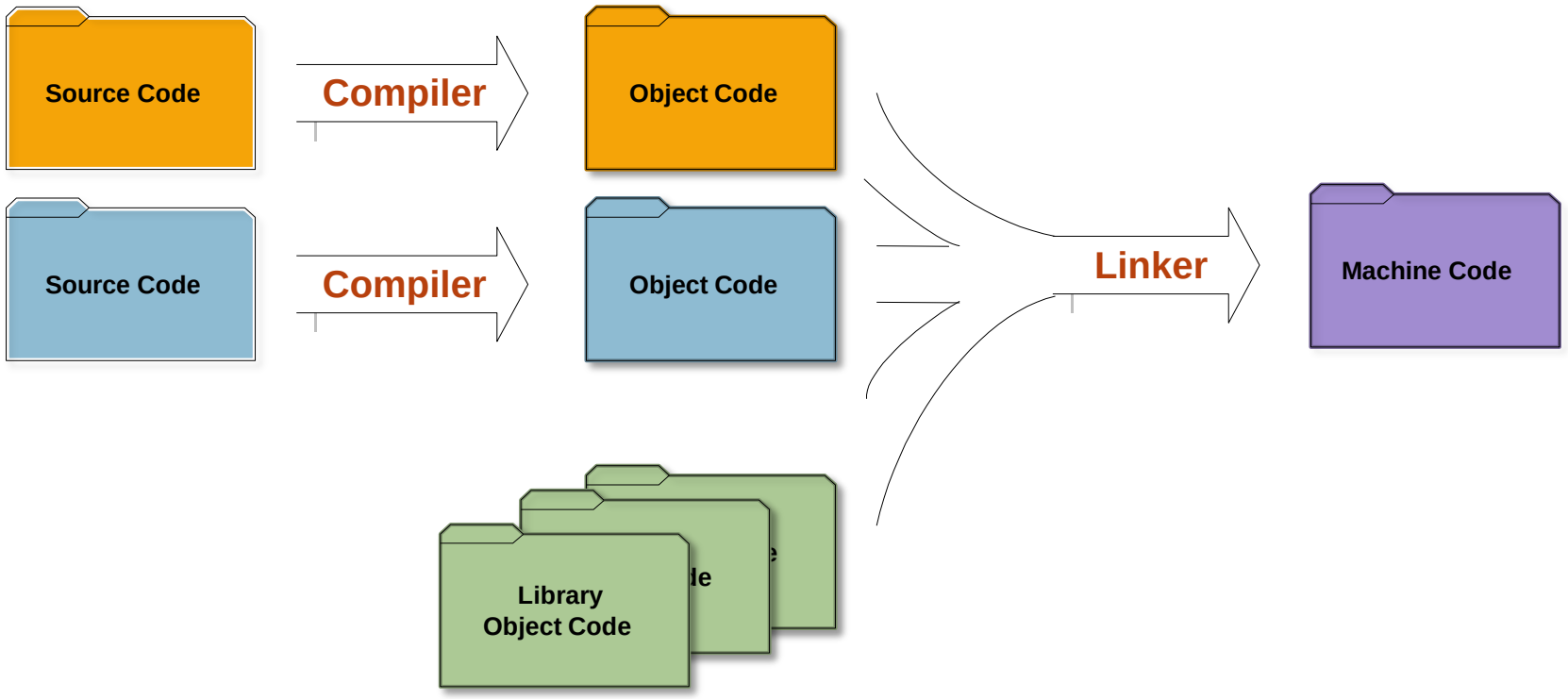
3η γενιά: Υψηλού επιπέδου (high-level) II

- ο προγραμματιστής δίνει εντολές στον υπολογιστή σε μια κατανοητή και καλά δομημένη γλώσσα (source code)
- ο μεταγλωτιστής (compiler) τις μετατρέπει σε ενδιάμεσο κώδικα (object code)
- ο ενδιάμεσος κώδικας μετατρέπεται σε γλώσσα μηχανής (machine code)
- το ίδιο πρόγραμμα μπορεί να μεταγλωττιστεί και να εκτελεστεί σε διαφορετικές μηχανές



Γλώσσες Προγραμματισμού

3η γενιά: Υψηλού επιπέδου (high-level) III



Compiling, **linking**, **and** **executing** **a program**

Γλώσσες Προγραμματισμού

Πέντε γενεές

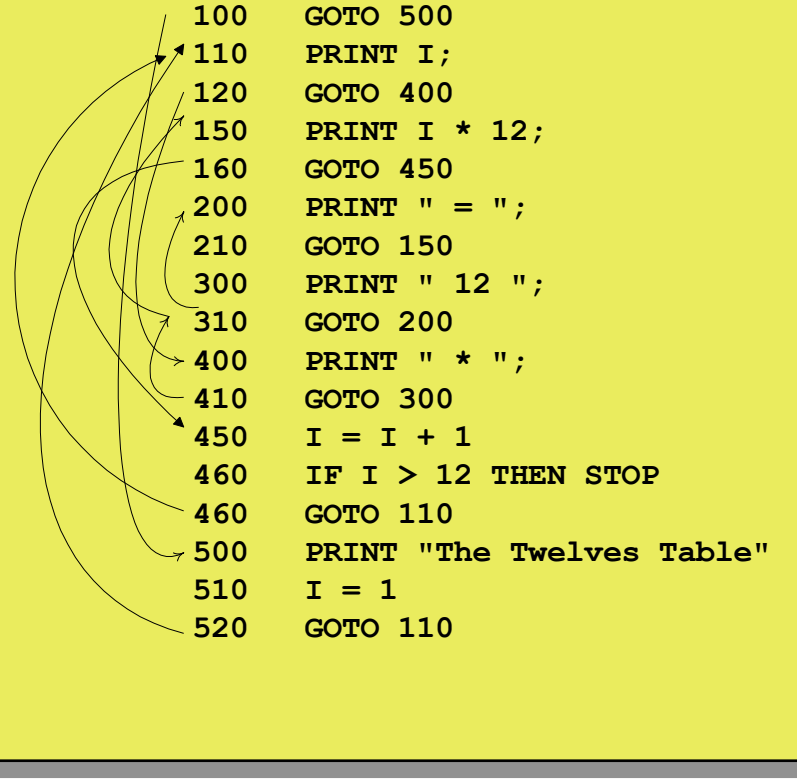
1. Γλώσσες μηχανής (machine language)
2. Συμβολικές γλώσσες (assembly language)
3. Γλώσσες υψηλού επιπέδου (high-level language)
4. Εξειδικευμένες γλώσσες
5. "Φυσικές" γλώσσες

✍ κάθε γενιά προσθέτει και ένα επίπεδο αφαίρεσης

Προγραμματιστικά Παραδείγματα

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
```

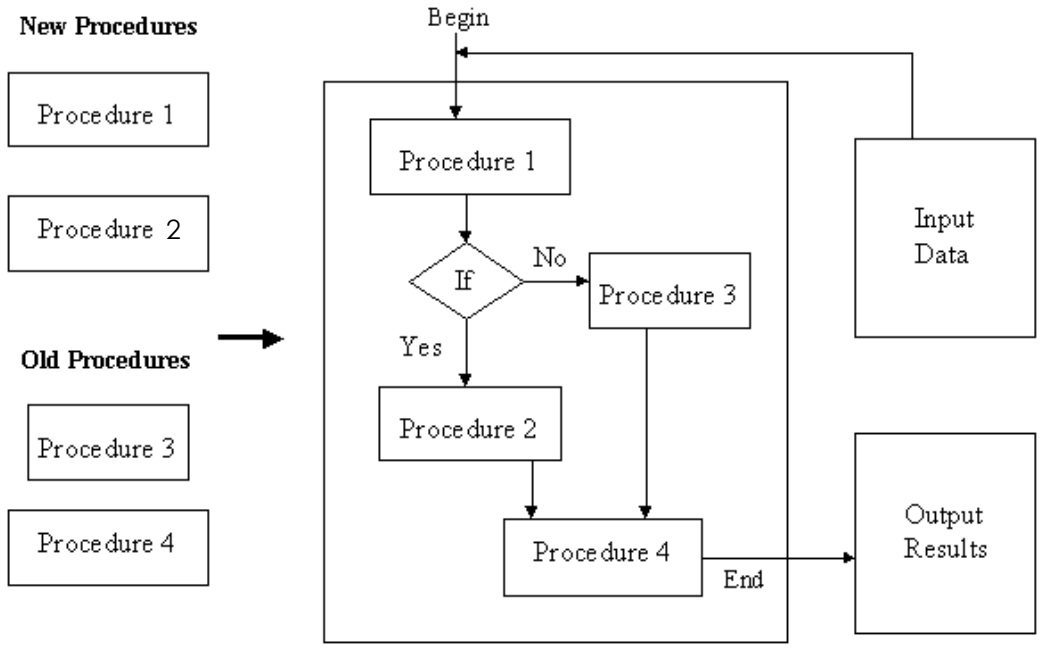
Προγραμματιστικά Παραδείγματα

Δομημένος προγραμματισμός

- τέσσερεις προγραμματιστικές δομές
  1. sequence: ακολουθιακές εντολές
  2. selection: επιλογή με if-then-else
  3. iteration: δημιουργία βρόχων
  4. 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)



Προγραμματιστικά Παραδείγματα
Διαδικασιακός προγραμματισμός (functional programming)

- 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

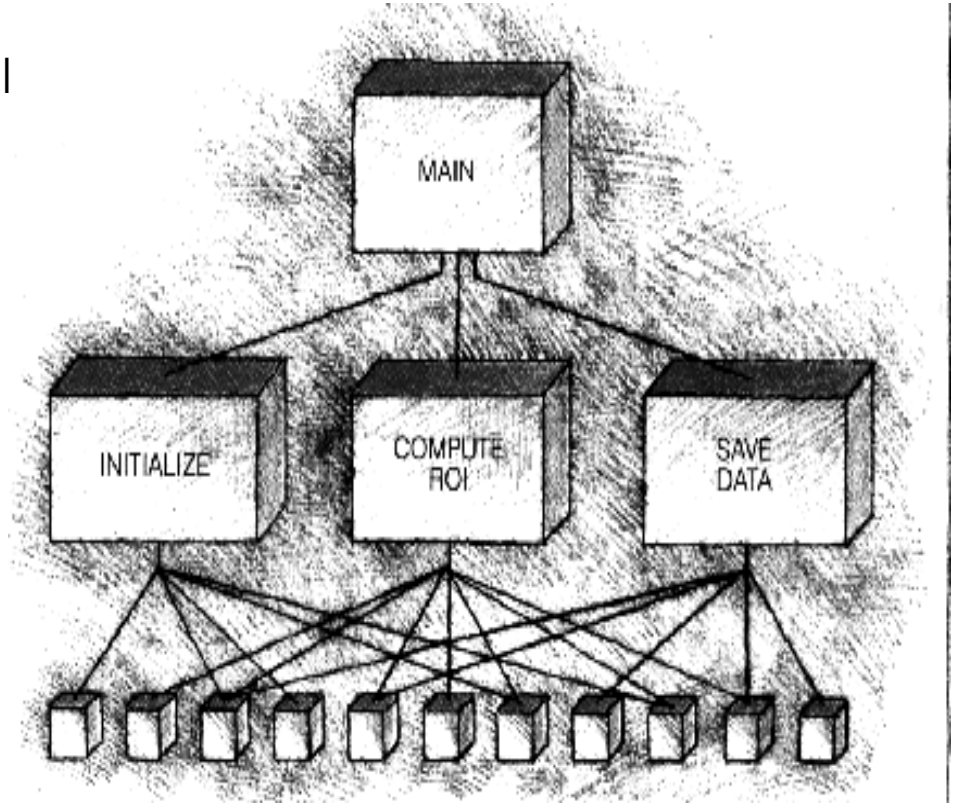
- το πρόγραμμα μας είναι τμηματοποιημένο (modular)

Προγραμματιστικά Παραδείγματα

Διαδικασιακός προγραμματισμός - 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 επικοινωνούν/εξαρτώνται μεταξύ τους

✍ πρέπει να αποφεύγουμε τη χρήση καθολικών μεταβλητών!

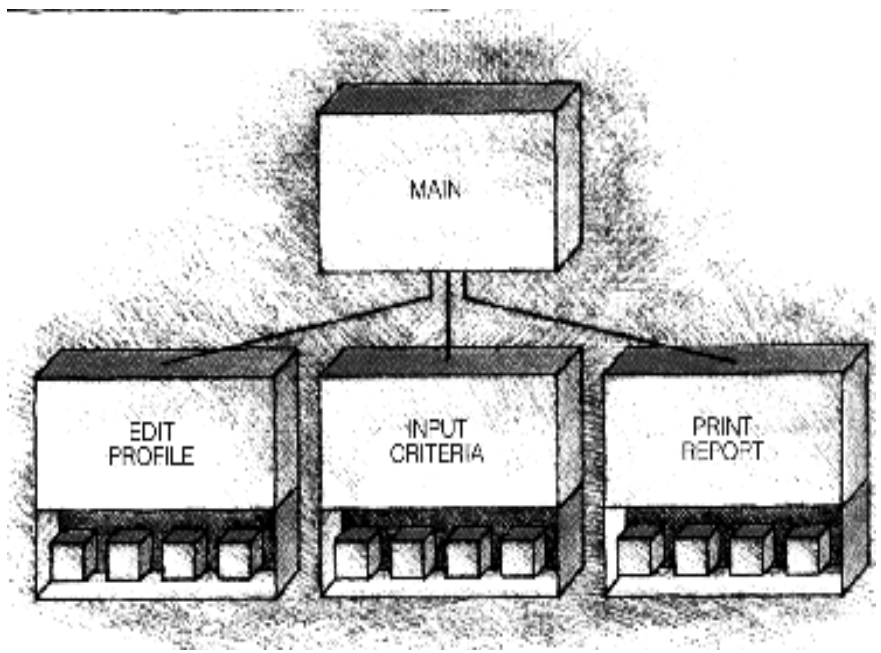


Προγραμματιστικά Παραδείγματα

Τμηματοποιημένος προγραμματισμός (modular programming)

απόκρυξη δεδομένων

- με τη δημιουργία τοπικών μεταβλητών μέσα στις διαδικασίες αποφεύγουμε την ύπαρξη κοινών δεδομένων
- η επικοινωνία μεταξύ των διαδικασιών γίνεται με ορίσματα και επιτρεφόμενες τιμές



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

Προγραμματιστικά Παραδείγματα
Τμηματοποιημένος προγραμματισμός - Περιορισμοί

ο τμηματοποιημένος προγραμματισμός

- δουλεύει για μικρά προγράμματα
- για μεγάλα συστήματα είναι δύσκολο να σχεδιάσουμε, να υλοποιήσουμε και να συντηρήσουμε τον κώδικα
  - δεν είναι εύκολο να προσαρμοστούμε σε αλλαγές
  - δε μπορούμε να προβλέψουμε όλες τις ανάγκες που θα έχουμε

Προγραμματιστικά Παραδείγματα

Τμηματοποιημένος προγραμματισμός - Περιορισμοί - Παράδειγμα

παράδειγμα: το ΕΛΜΕΠΑ έχει ένα σύστημα για να κρατάει πληροφορίες για φοιτητές και καθηγητές

- ✓ υπάρχει μια διαδικασία `print1` που τυπώνει στοιχεία και βαθμούς φοιτητών
- ☞ προκύπτει ανάγκη για μια διαδικασία που να τυπώνει τα μαθήματα των καθηγητών
  - ▶ χρειαζόμαστε μια `print2`

✍ τους περιορισμούς αυτούς προσπαθεί να αντιμετωπίσει ο αντικειμενοστραφής προγραμματισμός (object-oriented programming)

Προγραμματιστικά Παραδείγματα
Αντικειμενοστραφής προγραμματισμός (object-oriented programming)

τοποθετεί μαζί τα δεδομένα και τις διαδικασίες (μεθόδους) που είναι σχετικές με τα δεδομένα

- π.χ., ο φοιτητής ή ο καθηγητής έρχεται με μια δικιά του διαδικασία `print`

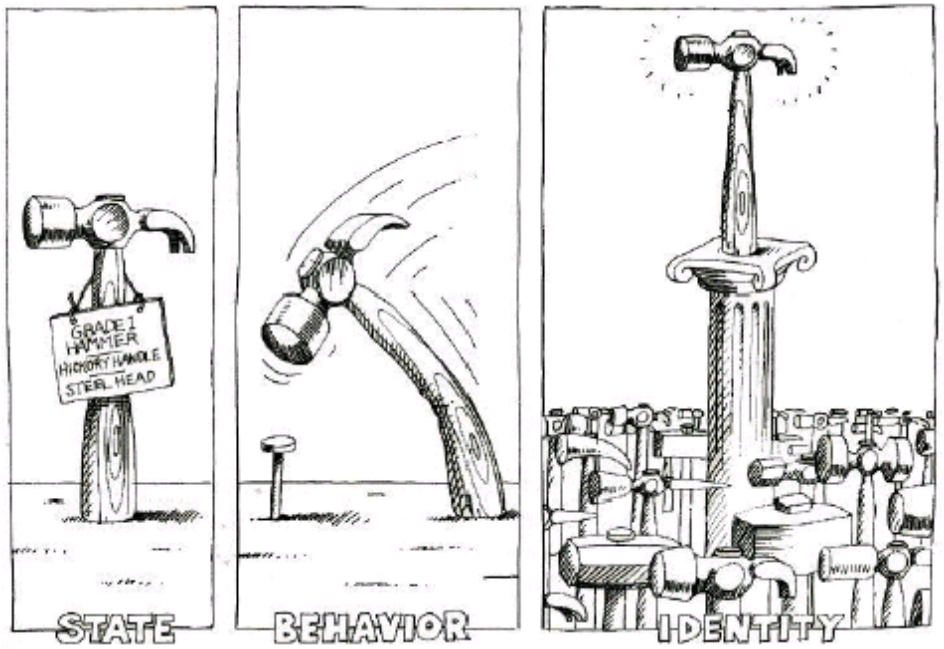
☞ αυτό επιτυγχάνεται με αντικείμενα και κλάσεις

Προγραμματιστικά Παραδείγματα

Αντικειμενοστραφής προγραμματισμός - 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 την ξεχωρίζει από τις υπόλοιπες μονάδες/οντότητες/έννοιες ίδιου τύπου

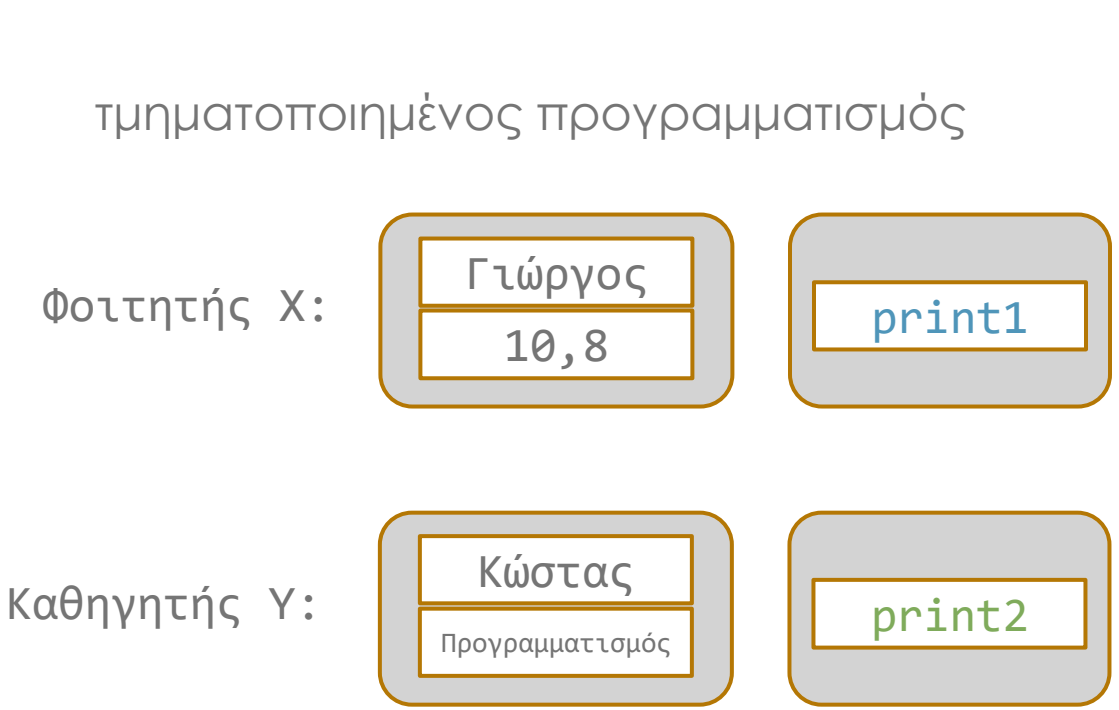


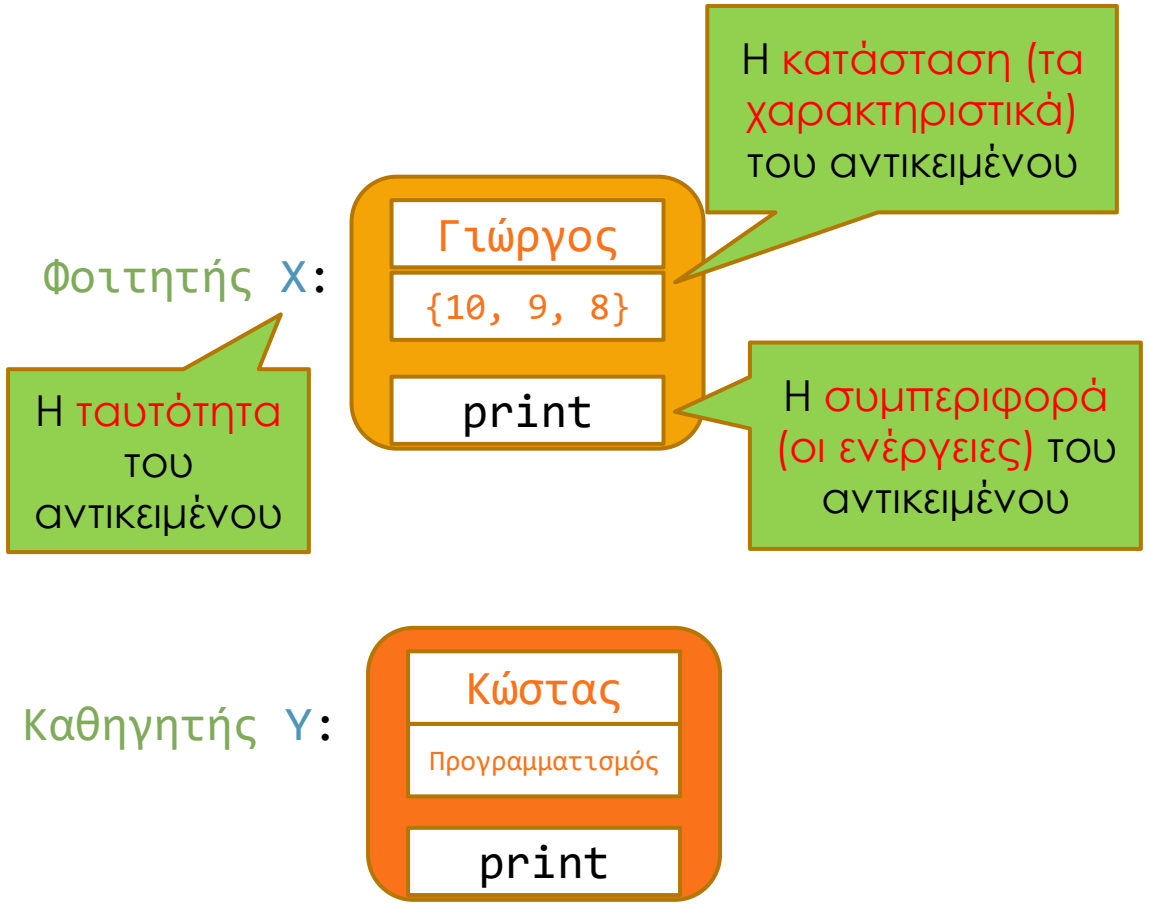
- παραδείγματα αντικειμένων: ένας άνθρωπος, ένα πράγμα, ένα μέρος, μία υπηρεσία

Προγραμματιστικά Παραδείγματα

Αντικειμενοστραφής προγραμματισμός - Αντικείμενο - Παράδειγμα

τμηματοποιημένος προγραμματισμός





Προγραμματιστικά Παραδείγματα
Αντικειμενοστραφής προγραμματισμός - Κλάσεις

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

  Παραδείγματα:

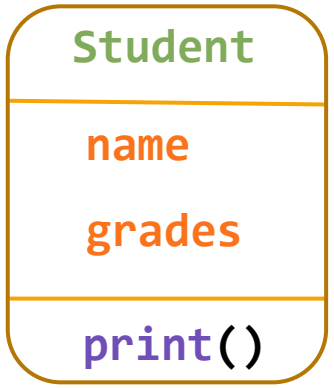
  1. η κλάση `Φοιτητής` έχει τα γενικά χαρακτηριστικά (όνομα, βαθμοί) και τη συμπεριφορά `print`
     - ο φοιτητής `X` είναι ένα αντικείμενο της κλάσης `Φοιτητής` με κατάσταση τα χαρακτηριστικά (Γιώργος, 10,8)
  2. η κλάση `Car` έχει τα χαρακτηριστικά (brand, color) και τη συμπεριφορά (drive, stop)
     - το αυτοκίνητο `INI2013` είναι ένα αντικείμενο της κλάσης `Car` με κατάσταση τα χαρακτηριστικά (BMW, red)

Προγραμματιστικά Παραδείγματα

Αντικειμενοστραφής προγραμματισμός - Κλάσεις και αντικείμενα - Σύνοψη

Κλάση

Μια αφηρημένη περιγραφή ενός φοιτητή



Όνομα κλάσης

Πεδία κλάσης: Ιδιότητες/Χαρακτηριστικά

Μέθοδοι κλάσης: λειτουργίες

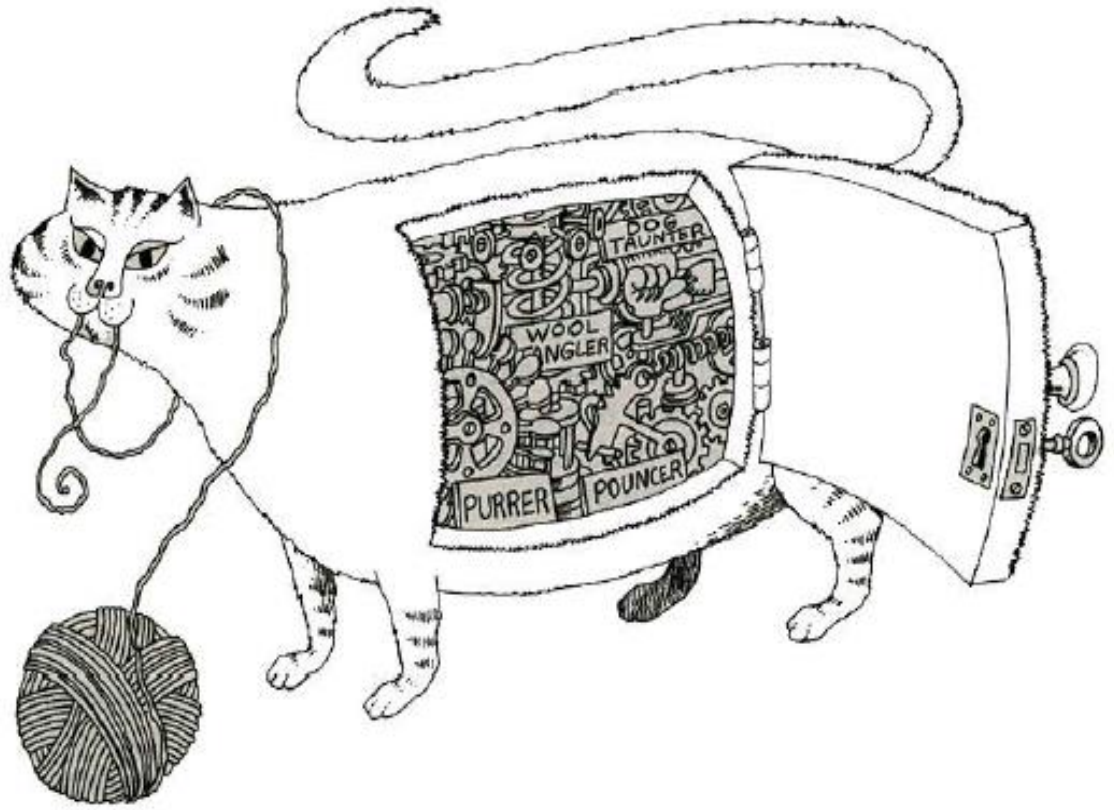
Αντικείμενα

Το κάθε αντικείμενο έχει
- μια κατάσταση (name, grade)
- ενέργειες (print)
- ταυτότητα (X,Y,Z)

Προγραμματιστικά Παραδείγματα

Αντικειμενοστραφής προγραμματισμός - Ενθυλάκωση

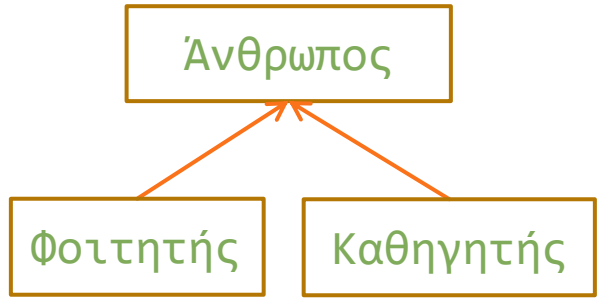
- η στεγανοποίηση της κατάστασης και της συμπεριφοράς ώστε οι λεπτομέρειες της υλοποίησης να είναι κρυμμένες από το χρήστη του αντικειμένου



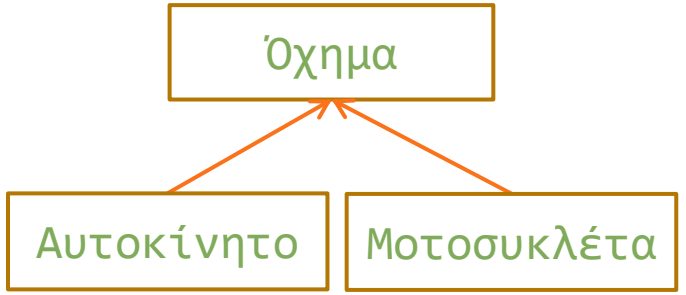
Προγραμματιστικά Παραδείγματα

Αντικειμενοστραφής προγραμματισμός - Κληρονομικότητα

- οι κλάσεις μας επιτρέπουν να ορίσουμε μια ιεραρχία
  - 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



Προγραμματιστικά Παραδείγματα

Αντικειμενοστραφής προγραμματισμός - Πολυμορφισμός

- κλάσεις με κοινό πρόγονο έχουν κοινά χαρακτηριστικά, αλλά έχουν και διαφορές
  - π.χ., είναι διαφορετικό το παρκάρισμα για ένα αυτοκίνητο και μια μοτοσυκλέτα

- 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

Προγραμματιστικά Παραδείγματα
Αντικειμενοστραφής προγραμματισμός - Σύνοψη

- χρησιμοποιώντας τις κλάσεις μπορούμε να ορίσουμε τους δικούς μας τύπους δεδομένων
  - έτσι μπορούμε να φτιάξουμε αντικείμενα με συγκεκριμένα χαρακτηριστικά και συμπεριφορά (μεθόδους)

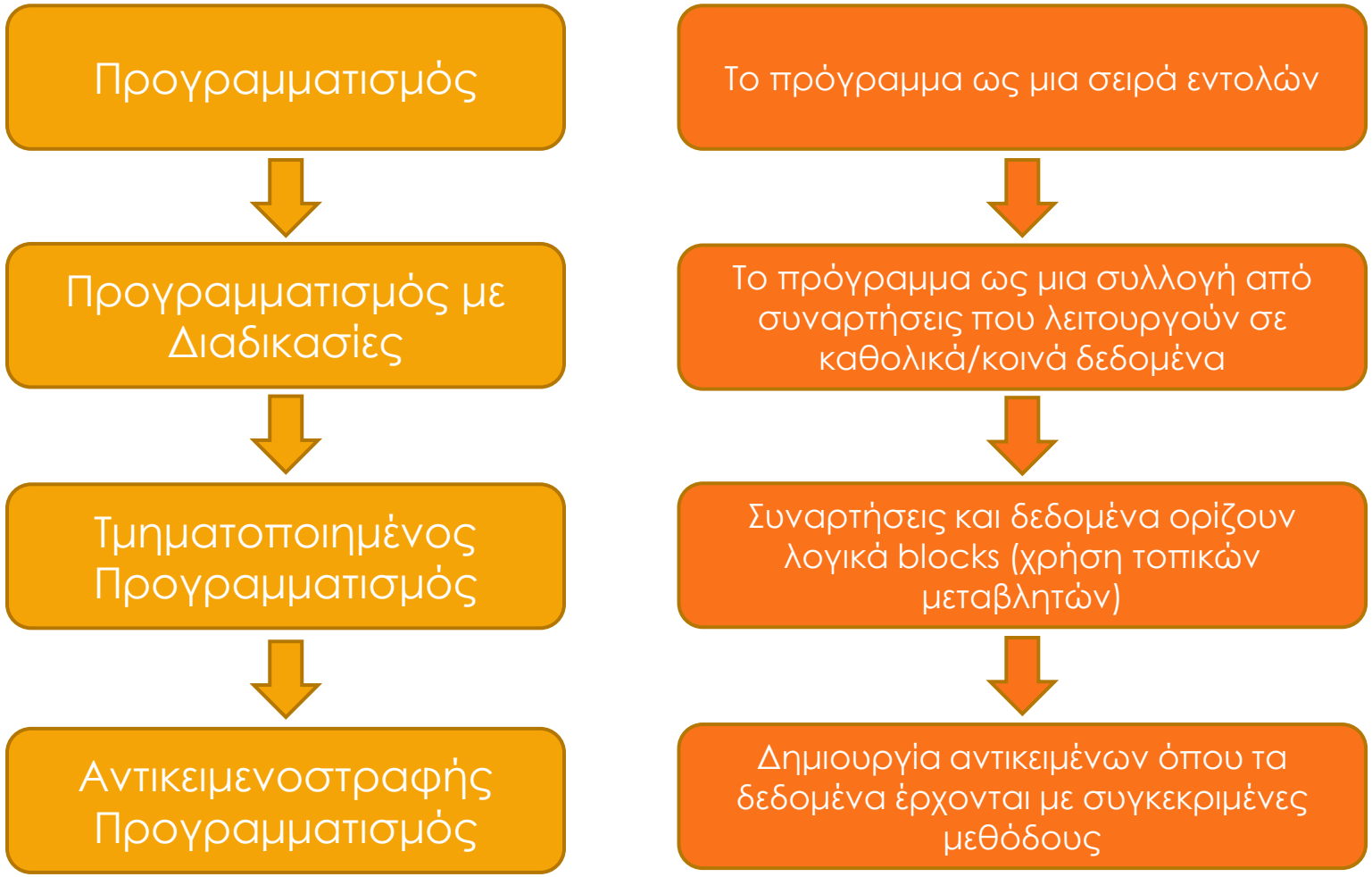
- χρησιμοποιώντας την κληρονομικότητα και τον πολυμορφισμό, μπορούμε να επαναχρησιμοποιήσουμε υπάρχοντα χαρακτηριστικά και μεθόδους

✍ ο πιο δημοφιλής ορισμός του αντικειμενοστραφούς προγραμματισμού:

κληρονομικότητα + πολυμορφισμός + ενθυλάκωση

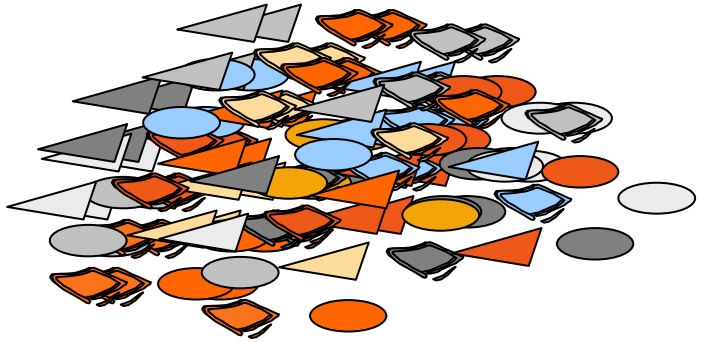
Προγραμματιστικά Παραδείγματα

Η εξέλιξη του προγραμματισμού - Σύνοψη

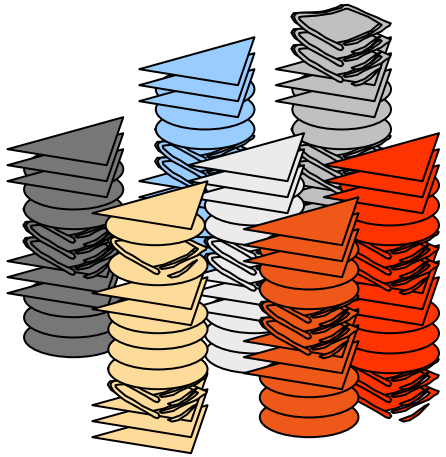


Προγραμματιστικά Παραδείγματα

Η εξέλιξη του προγραμματισμού - Σύνοψη II



Spaghetti-Code



functional decomposition

object decomposition

Λογισμικό =

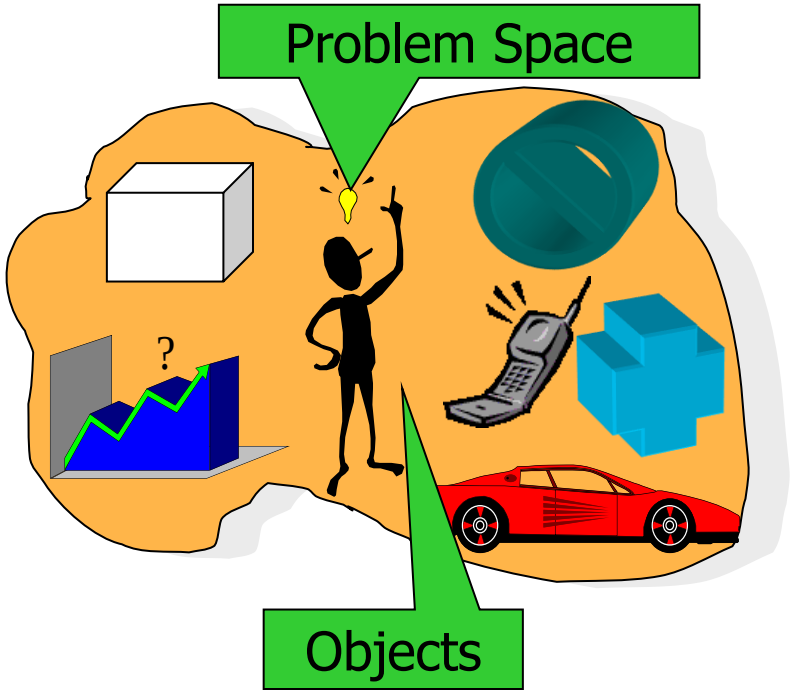
Δεδομένα (Σχήματα)

+

Λειτουργίες (Χρώματα)

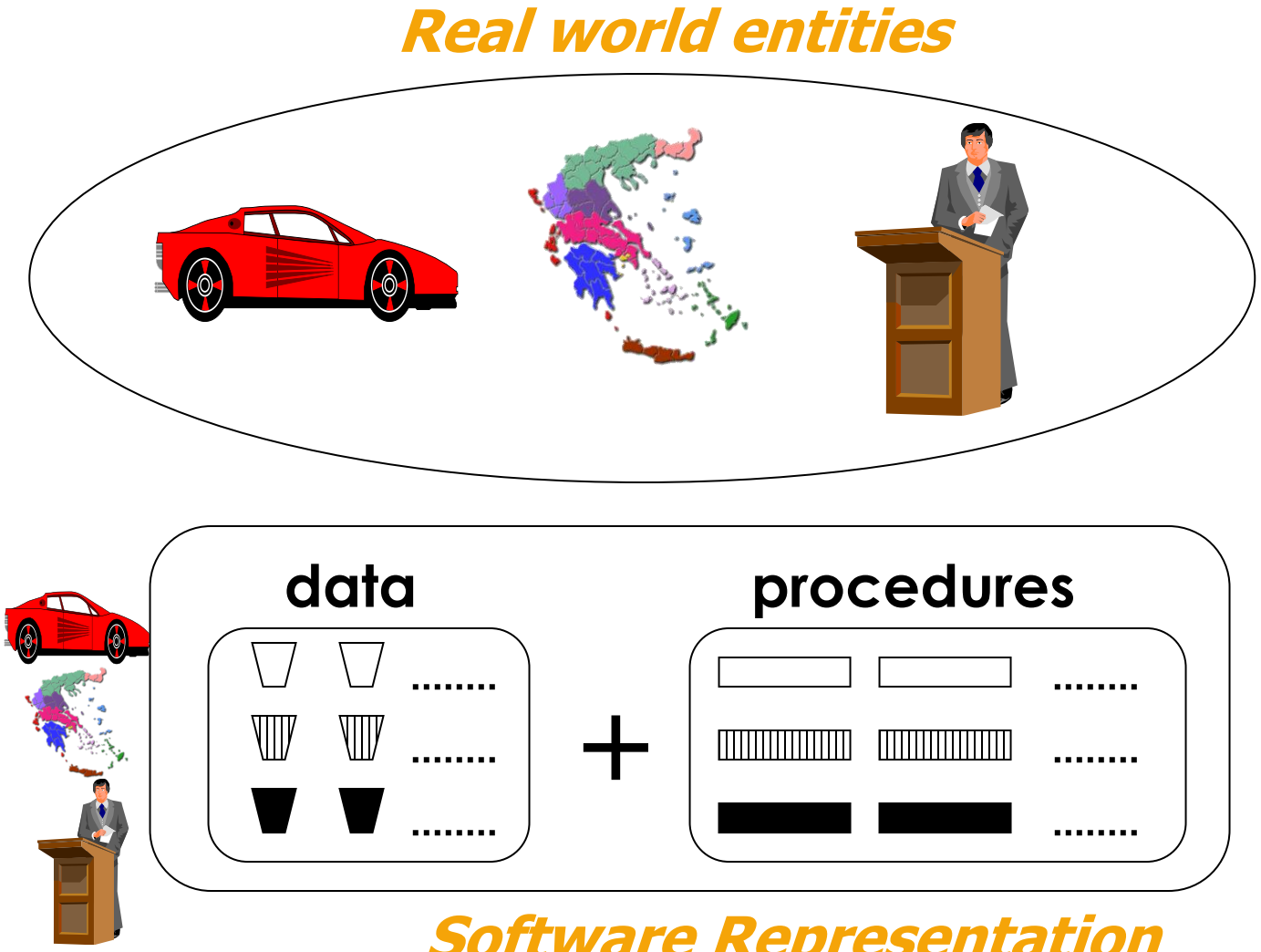
Διαδικασιακός vs Αντικειμενοστραφής προγραμματισμός

- 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



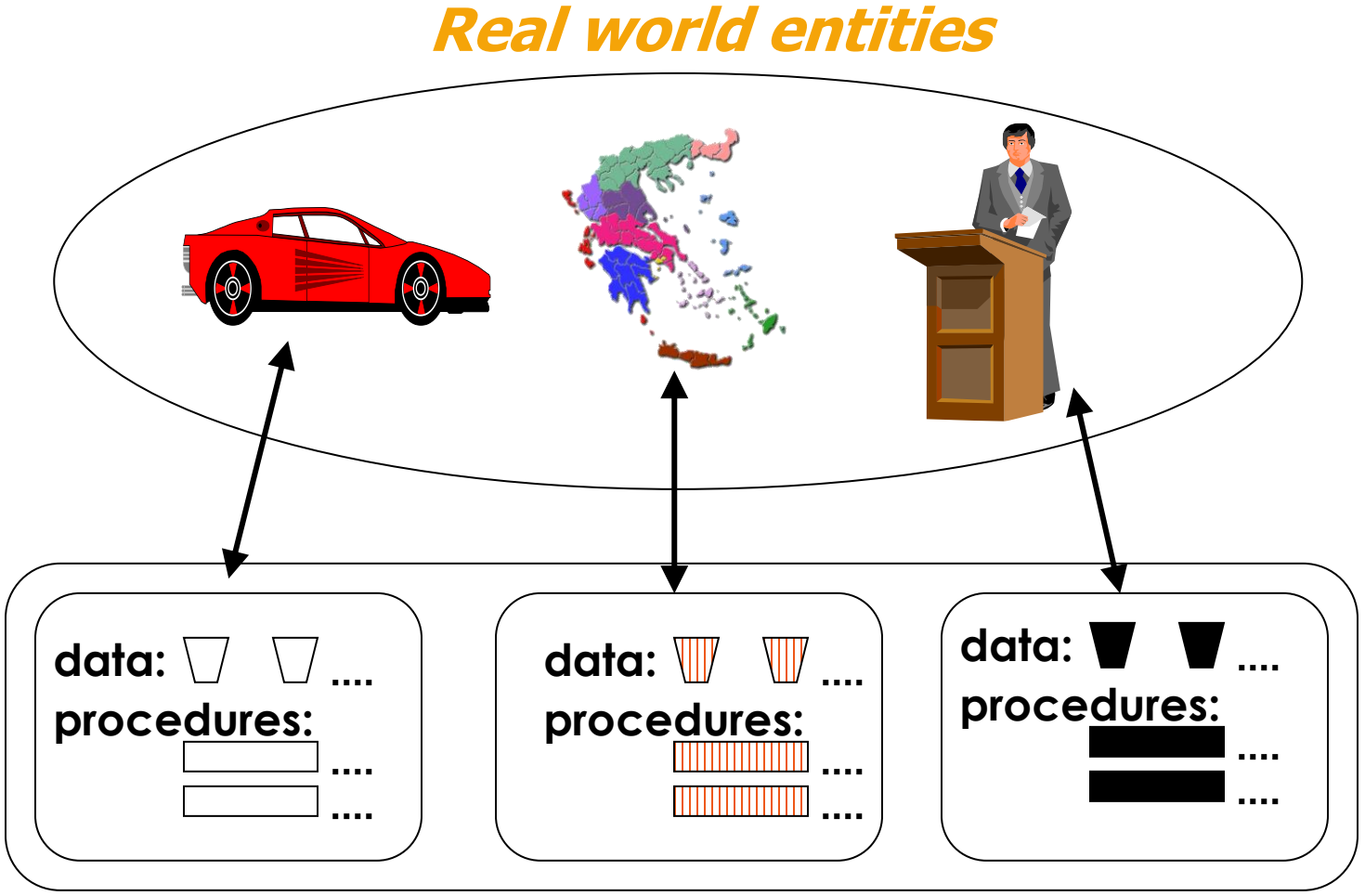
Διαδικασιακός vs Αντικειμενοστραφής προγραμματισμός

Παράδειγμά - Διαδικασιακή αναπαράσταση



Διαδικασιακός vs Αντικειμενοστραφής προγραμματισμός

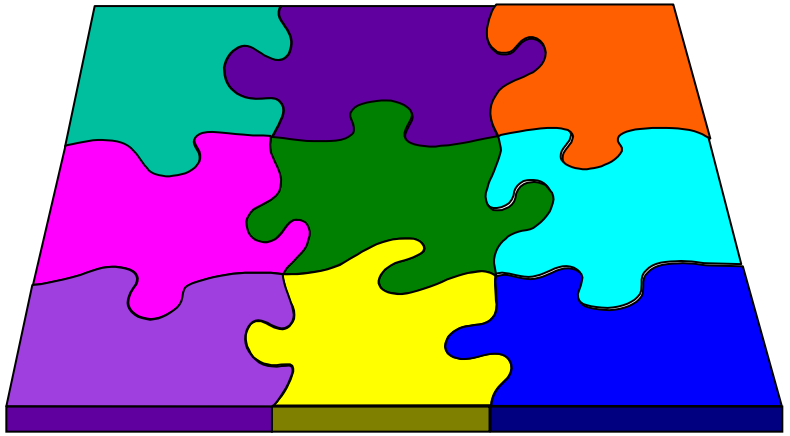
Παράδειγμά - Αντικειμενοστραφής αναπαράσταση



Διαδικασιακός vs Αντικειμενοστραφής προγραμματισμός

Πλεονεκτήματα αντικειμενοστραφούς προγραμματισμού

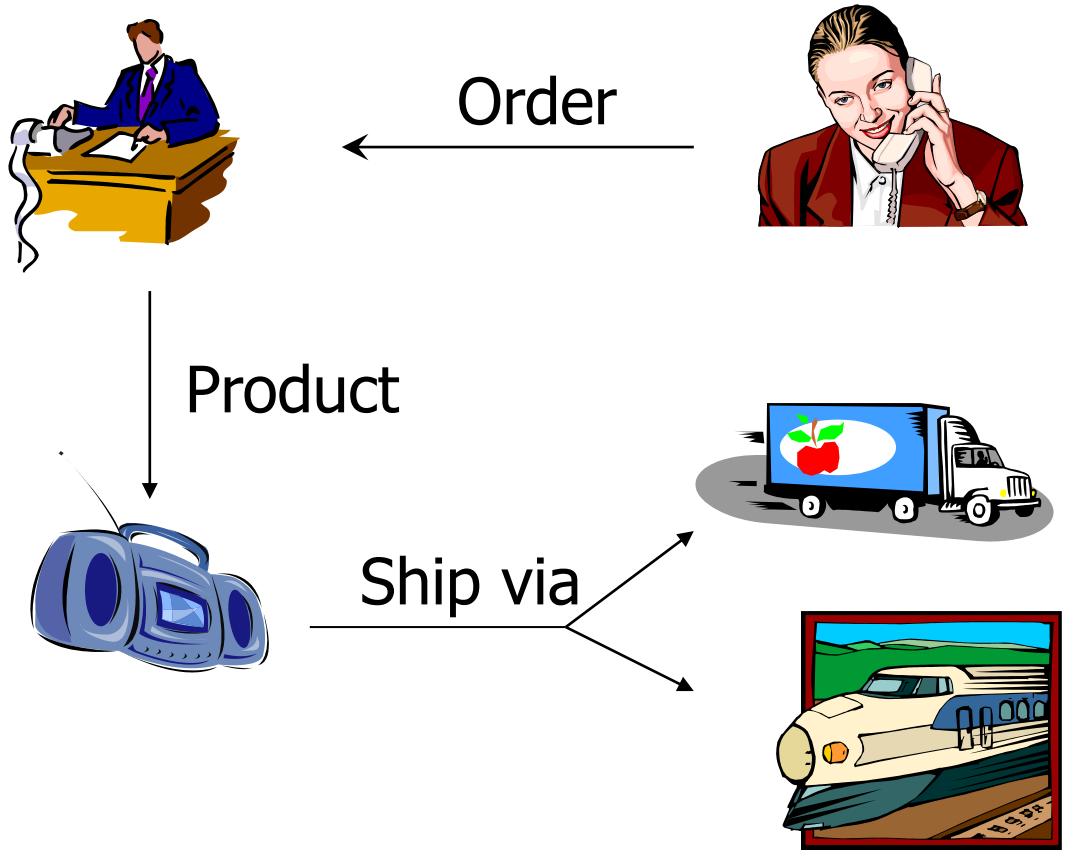
- επειδή προσπαθεί να μοντελοποιήσει τον πραγματικό κόσμο, ο αντικειμενοστραφής κώδικας είναι πιο κατανοητός
- τα δομικά κομμάτια που δημιουργεί είναι πιο εύκολο να επαναχρησιμοποιηθούν και να συνδυαστούν
- ο κώδικας είναι πιο εύκολο να συντηρηθεί λόγω της ενθυλάκωσης



Διαδικασιακός vs Αντικειμενοστραφής προγραμματισμός

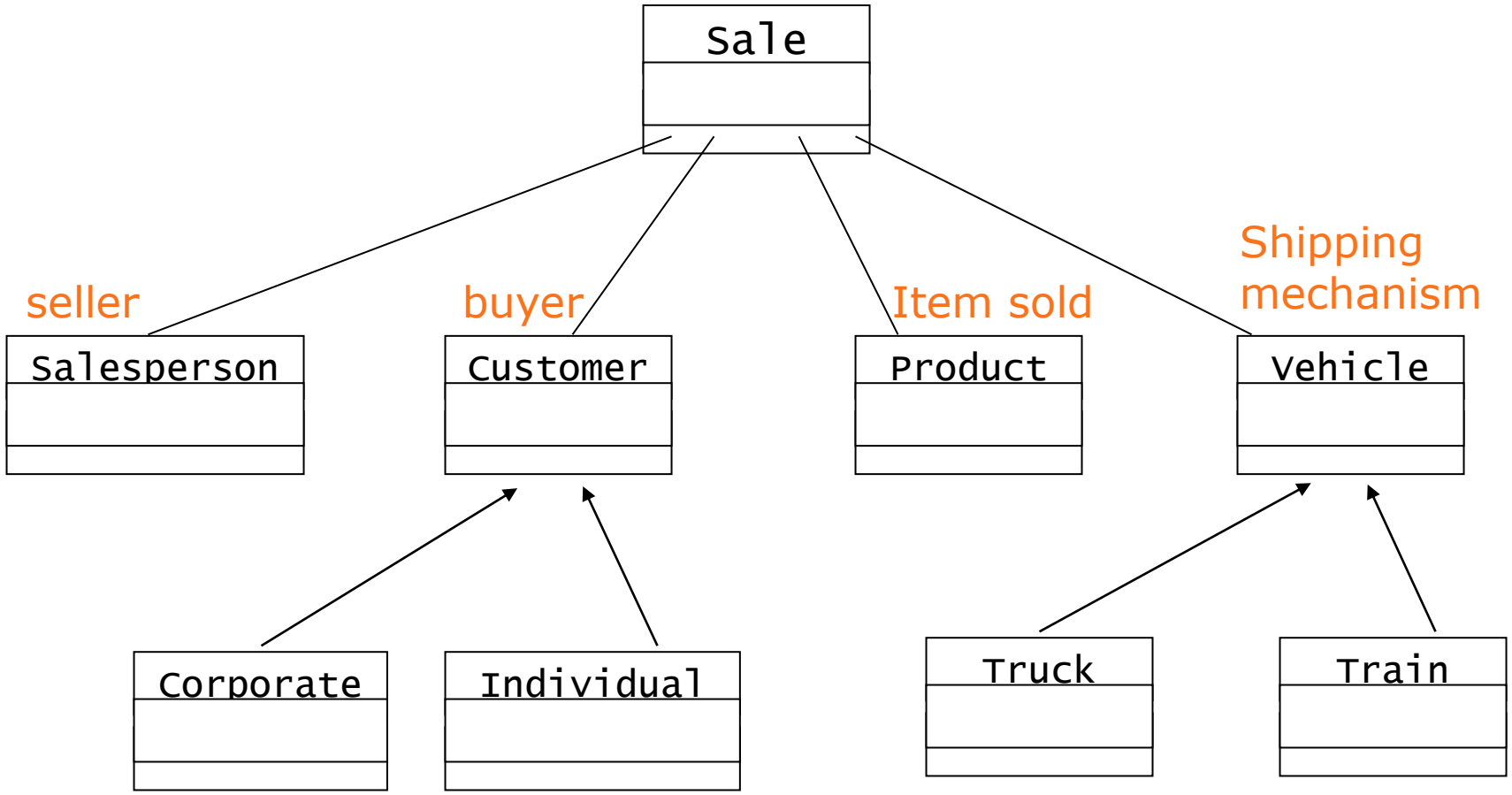
Πλεονεκτήματα αντικειμενοστραφούς προγραμματισμού - Παράδειγμα: Πωλήσεις

- Θέλουμε να δημιουργήσουμε λογισμικό 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



Διαδικασιακός vs Αντικειμενοστραφής προγραμματισμός

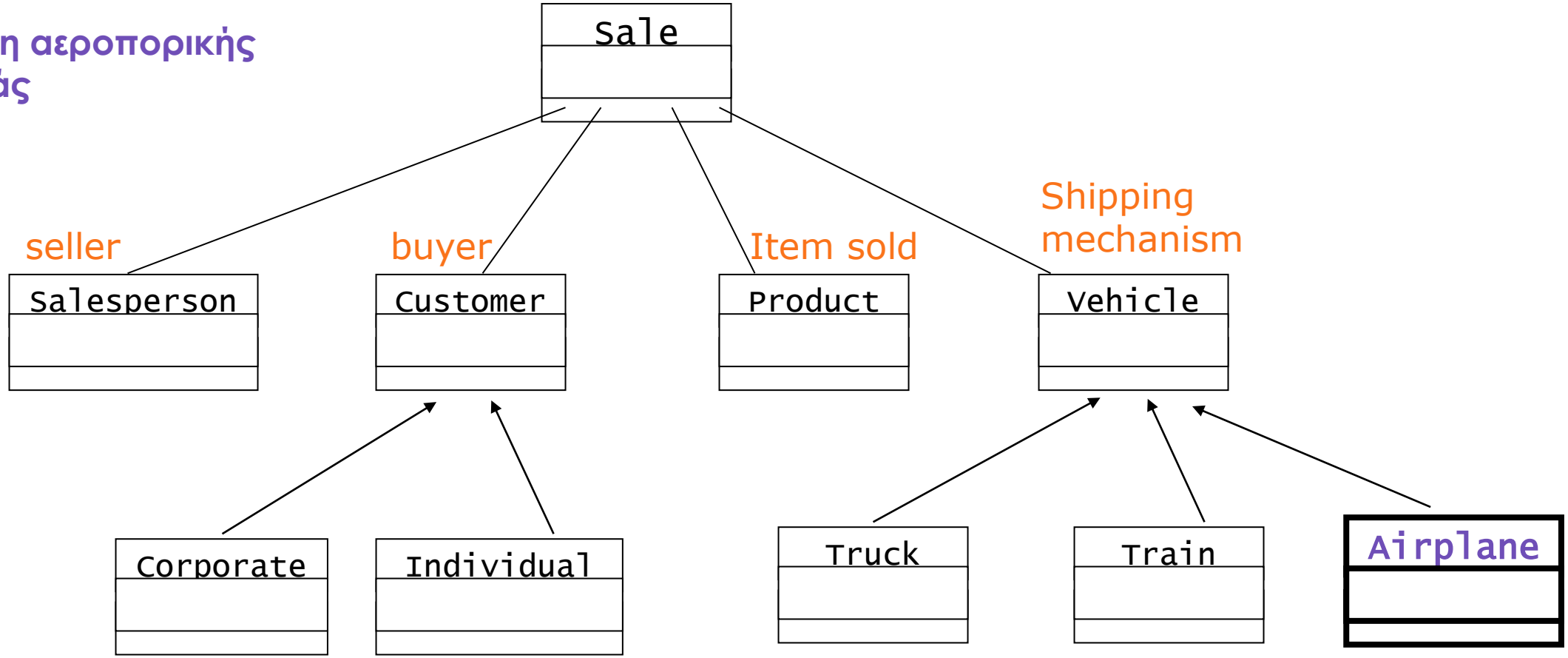
Πλεονεκτήματα αντικειμενοστραφούς προγραμματισμού - Παράδειγμα: Πωλήσεις



Διαδικασιακός vs Αντικειμενοστραφής προγραμματισμός

Πλεονεκτήματα αντικειμενοστραφούς προγραμματισμού - Παράδειγμα: Πωλήσεις - Αλλαγή των απαιτήσεων

Προσθήκη αεροπορικής μεταφοράς



Εισαγωγή στη Java

Ιστορία της Java

- ο Patrick Naughton απειλεί την Sun ότι θα φύγει
- τον βάζουν σε μία ομάδα αποτελούμενη από τους James Gosling και Mike Sheridan για να σχεδιάσουν τον προγραμματισμό τον έξυπνων συσκευών της επόμενης γενιάς
  - The Green project
- ο Gosling συνειδητοποιεί ότι η C++ δεν είναι αρκετά αξιόπιστη για να δουλεύει σε συσκευές περιορισμένων δυνατοτήτων και με διάφορες αρχιτεκτονικές
  - Δημιουργεί τη γλώσσα Oak
- το 1992 η ομάδα κάνει ένα demo μιας συσκευής PDA, *7 (star 7)
  - Δημιουργείται η θυγατρική εταιρία FirstPerson Inc
- η δημιουργία των έξυπνων συσκευών αποτυγχάνει και η ομάδα (μαζί με τον Eric Schmidt) επικεντρώνεται στην εφαρμογή της πλατφόρμας στο Internet
  - ο Naughton φτιάχνει τον WebRunner browser (μετά HotJava)
  - η γλώσσα μετονομάζεται σε Java και το ενδιαφέρον επικεντρώνεται σε εφαρμογές που τρέχουν μέσα στον browser
- Ο Marc Andersen ανακοινώνει ότι ο Netscape browser θα υποστηρίζει Java μικροεφαρμογές (applets)

Java

- Η Java είχε τους εξής στόχους:
  - "simple, object-oriented, and familiar"
  - "robust and secure"
  - "architecture-neutral and portable"
  - "high performance"
  - "interpreted, threaded, and dynamic"

Java

- Η Java είχε τους εξής στόχους:
  - **"simple, object-oriented, and familiar"**
  - "robust and secure"
  - **"architecture-neutral and portable"**
  - "high performance"
  - "interpreted, threaded, and dynamic"

Java

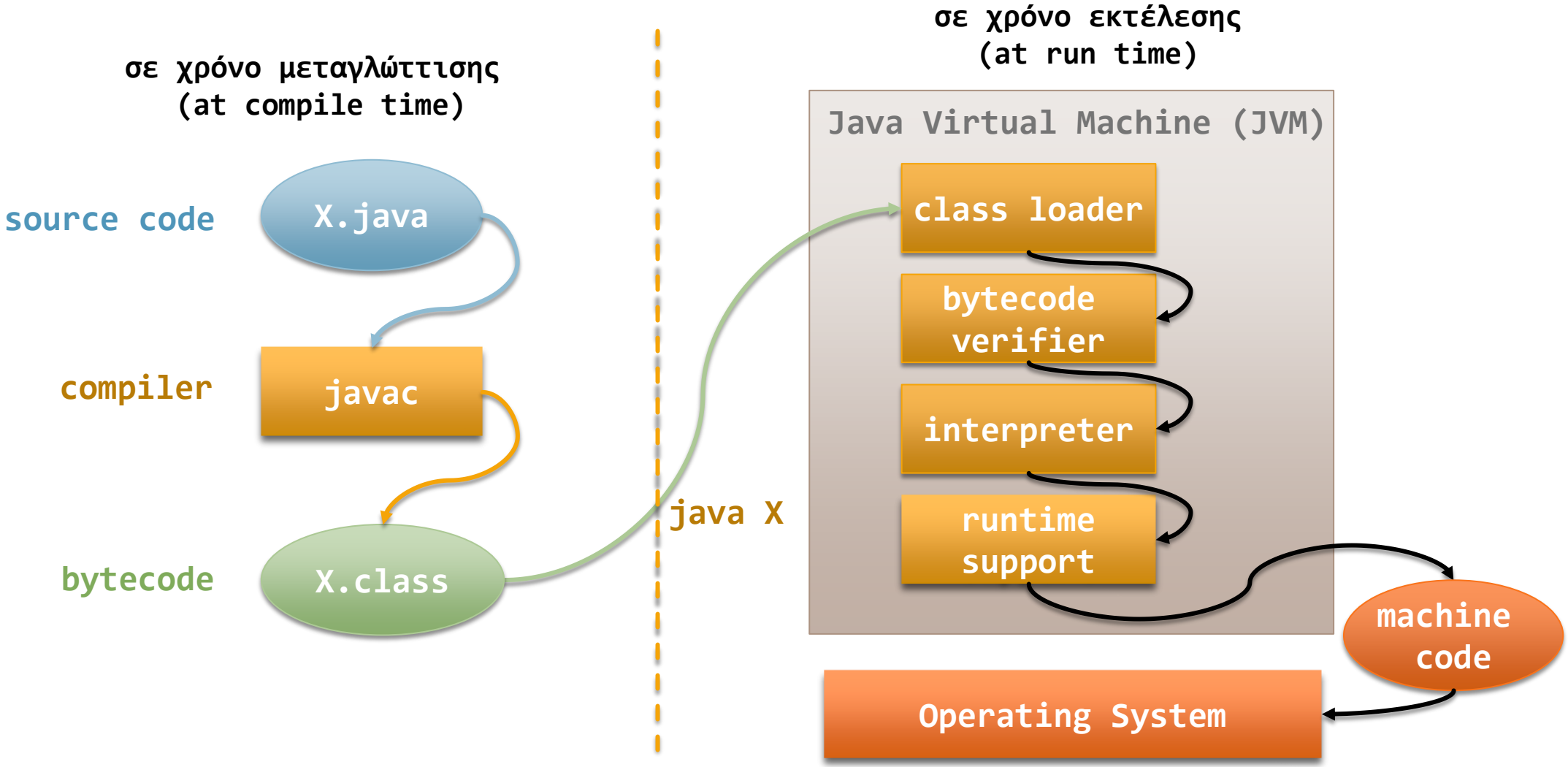
“architecture-neutral and portable”

- το μεγαλύτερο πλεονέκτημα της Java είναι η μεταφερσιμότητα (portability): ο κώδικας μπορεί να τρέξει πάνω σε οποιαδήποτε πλατφόρμα
  - write-once-run-anywhere μοντέλο
  - σε αντίθεση με το σύνηθες write-once-compile-anywhere μοντέλο

- αυτό επιτυγχάνεται δημιουργώντας ένα ενδιάμεσο κώδικα (bytecode) ο οποίος μετά τρέχει πάνω σε μια εικονική μηχανή (Java Virtual Machine) η οποία το μεταφράζει σε γλώσσα μηχανής
  - οι προγραμματιστές πλέον γράφουν κώδικα για την εικονική μηχανή, η οποία εκτελείται σε οποιαδήποτε πλατφόρμα

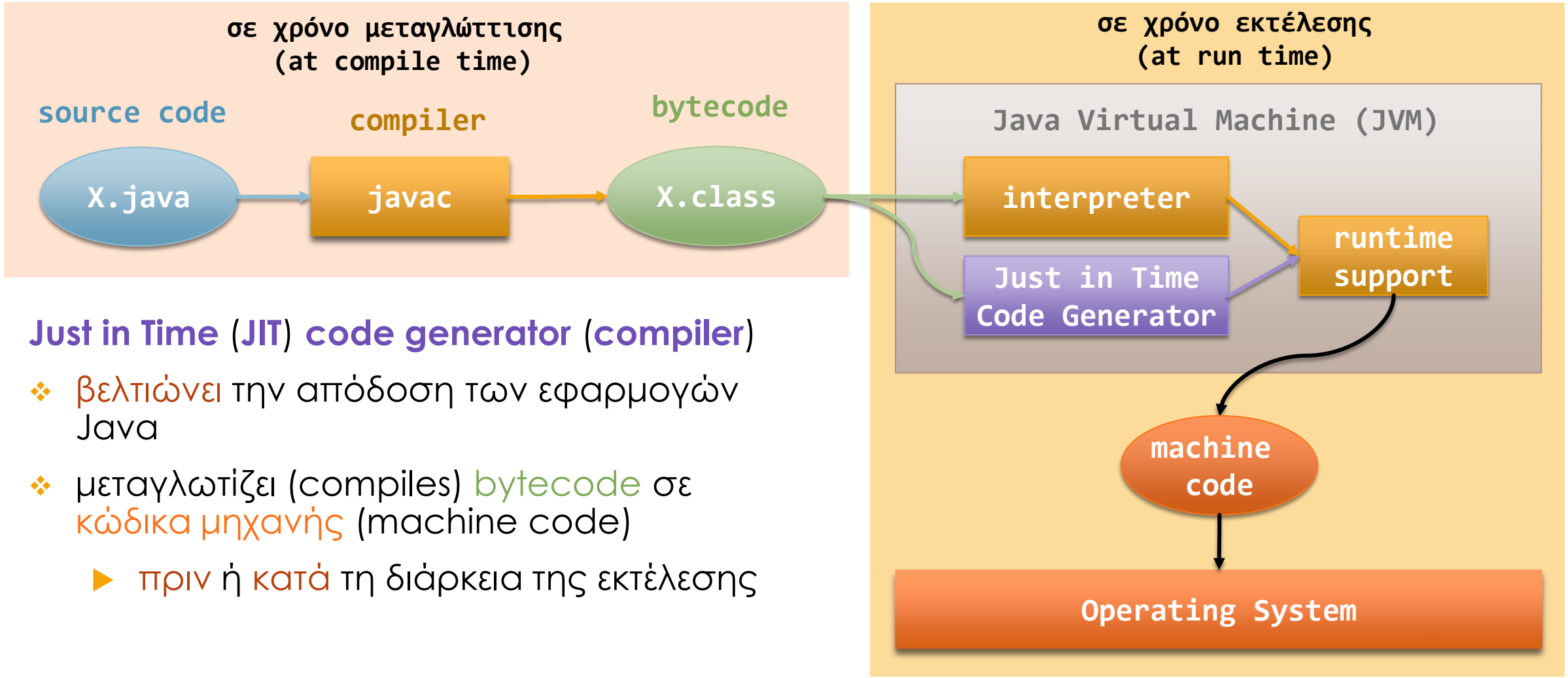
Java

"architecture-neutral and portable" - Διερμηνεία (interpretation)



Java

"architecture-neutral and portable" - Μεταγλώττιση (compiling)



Just in Time (**JIT**) **code generator** (**compiler**)

- βελτιώνει την απόδοση των εφαρμογών Java
- μεταγλωτίζει (compiles) bytecode σε κώδικα μηχανής (machine code)
  - πριν ή κατά τη διάρκεια της εκτέλεσης

Java - "architecture-neutral and portable" - Διαδίκτυο (Internet)

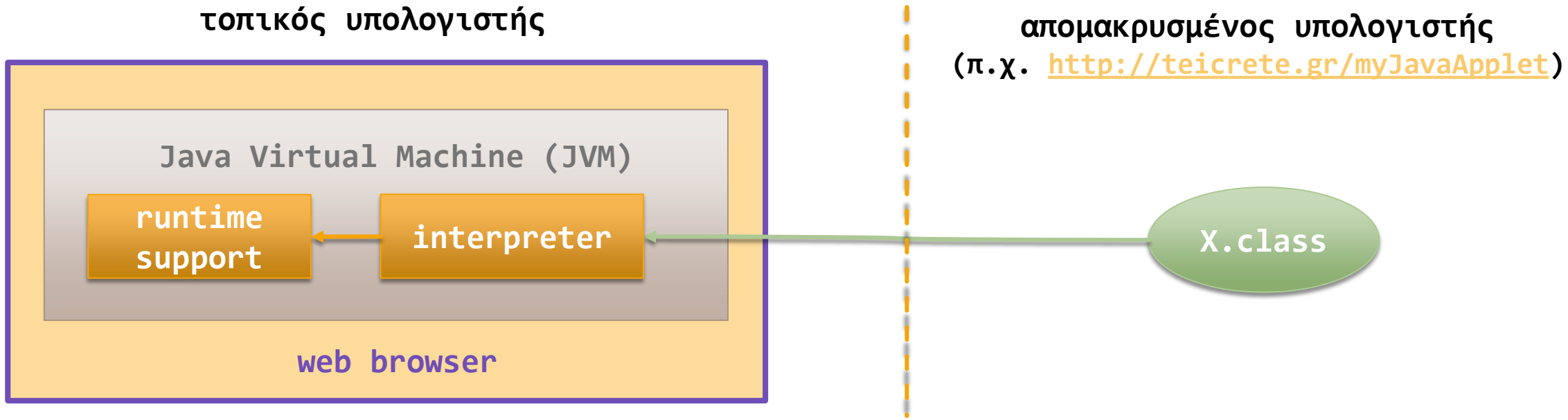
- η προσέγγιση της Java είχε μεγάλη επιτυχία για εφαρμογές διαδικτύου, όπου έχουμε ένα τεράστιο κατανεμημένο μοντέλο πελάτη-εξυπηρετητή (client-server) με πολλές διαφορετικές αρχιτεκτονικές

1. προγραμματισμός στο μέρος του πελάτη (client-side programming): αντί να κάνει όλη τη δουλειά ο εξυπηρετητής για τη δημιουργία της σελίδας κάποια από την επεξεργασία των δεδομένων γίνεται στη μηχανή του πελάτη
   - web Applets: κώδικας ο οποίος κατεβαίνει μαζί με την ιστοσελίδα και τρέχει στη μηχανή του πελάτη
     - πολύ σημαντικό: ο κώδικας είναι μεταφέρσιμος (portable)
     - Java Applets : η λύση της Java
2. προγραμματισμός στο μέρος του εξυπηρετητή (server-side programming): μία ιστοσελίδα μπορεί να είναι το αποτέλεσμα ενός προγράμματος που συνδυάζει δυναμικά δεδομένα και είσοδο του χρήστη
   - Java Service Pages (JSPs): η λύση της Java
     - μεταγλωττίζεται (compiled) σε servlets και τρέχει στη μεριά του εξυπηρετητή

Java - "architecture-neutral and portable" - Διαδίκτυο (Internet)

Java Applets

- ο **φυλλομετρητής** (web browser) στην πλευρά του χρήστη περιλαμβάνει ένα **JVM**
  - φορτώνει το Java **bytecode** από τον απομακρυσμένο υπολογιστή
  - εκτελεί τοπικά το πρόγραμμα Java μέσα στο παράθυρο του φυλλομετρητή



Java

"simple, object-oriented, and familiar"

- familiar
  - η Java είχε ως έμπνευση της την C++,
  - δανείζεται αρκετά από τα χαρακτηριστικά της C++
- object-oriented
  - η Java είναι «πιο αντικειμενοστραφής» από τη C++
    - καθώς, η C++ οποία προσπαθεί να μείνει συμβατή με τη C
  - στη Java τα πάντα είναι αντικείμενα
- simple : η Java δίνει λιγότερο έλεγχο στο χρήστη → αλλά κάνει τη ζωή του πιο εύκολη
  - 👍 φροντίζει να κάνει πιο γρήγορο και πιο σταθερό (robust) τον προγραμματισμό
    - η διαχείριση της μνήμης γίνεται αυτόματα
  - 👎 αυτό μπορεί να έχει ως αποτέλεσμα τα προγράμματα να εκτελούνται πιο αργά

Σύνοψη

- Στόχοι και ύλη μαθήματος
- Προγραμματισμός
- 5 γενεές γλωσσών προγραμματισμού
- Προγραμματιστικά παραδείγματα
  - Αντικειμενοστραφής προγραμματισμός
- Εισαγωγή στη Java
  - Μεταγλώττιση, ενδιάμεσος κώδικας, διερμηνεία