Ενότητα 1: Εισαγωγή στην Αντικειμενοστρέφεια

Φροντιστήριο - 9/2/24

Προγραμματιστικές Συμβάσεις και Στυλ

University of Cyprus
Department of Computer Science

"Programs must be written for people to read, and only incidentally for machines to execute."

Abelson & Sussman, SICP, preface to the 1st edition

Λόγοι Υιοθέτησης Προγραμ. Συμβάσεων

- Οι προγραμματιστικές συμβάσεις είναι σημαντικές για μια σειρά λόγων:
  - 80% του χρόνου κωδικοποίησης ενός προγράμματος αφορά στην συντήρηση του.
  - Σπάνια κάποιο λογισμικό συντηρείται από τον αρχικό προγραμματιστή.
  - Προγραμματιστικές συμβάσεις βελτιώνουν την αναγνωσιμότητα του λογισμικού και διευκολύνουν τους προγραμματιστές να κατανοούν τα προγράμματα πιο γρήγορα και καλύτερα.
  - Αν ο πηγαίος κώδικα σας πρόκειται να κυκλοφορήσει σαν προϊόν, θα πρέπει να είναι οργανωμένος και γραμμένος καθαρά.

Απόσπασμα από σωστό πρόγραμμα που μεταγλωττίζεται χωρίς πρόβλημα και λειτουργεί πολύ καλά:

```
if ( (country == SING) || (country == BRNI) ||
     (country == POL) || (country == ITALY) )
{
        /*
         * If the country is Singapore, Brunei or Poland
         * then the current time is the answer time
         * rather than the off hook time.
         * Reset answer time and set day of week.
         */
        ...
```

Βλέπετε κάποιο πρόβλημα με τον κώδικα;

"typical of much real code: mostly well done, but with some things that could be improved."

Θέματα Προγραμματιστικού Στυλ

- Στυλ προγραμματισμού
- Ονόματα
- Συνέπεια και ιδιωματισμοί
- Εκφράσεις
- Μαγικοί αριθμοί
- Σχόλια
- Οργάνωση αρχείων Java
- Javadoc

University of Cyprus
Department of Computer Science

M. D. Dikaiakos

Προγραμματιστικό Στυλ

- Ο σκοπός του στυλ είναι να κάνει τον κώδικα εύκολο στην ανάγνωση για σας και τους άλλους: το καλό στυλ είναι ζωτικής σημασίας για τον καλό προγραμματισμό:
  - Δεν αφορά μόνο στη σωστή σύνταξη ενός προγράμματος, τη διόρθωση των σφαλμάτων, την καλή επίδοση (performance).
  - Ένα καλογραμμένο πρόγραμμα είναι ευκολότερο να κατανοηθεί και να τροποποιηθεί από το κακογραμμένο.
  - Η πειθαρχία της καλής γραφής οδηγεί σε κώδικα που είναι πιο πιθανό να είναι σωστός.

Αρχές Καλού Στυλ

- Οι αρχές του καλού στυλ βασίζονται στην κοινή λογική και εμπειρία - όχι σε αυθαίρετους κανόνες.
- Ο κώδικας μας πρέπει να είναι ξεκάθαρος και σαφής:
  - Βασισμένος σε απλή λογική
  - Κώδικας γραμμένος σαν αφήγηση σε φυσική γλώσσα
  - Συμβατική χρήση της σύνταξης της Γ/Π - με βάση καλά παραδείγματα
  - Ονόματα μεταβλητών με ξεκάθαρο νόημα
  - Προσεγμένη μορφοποίηση (formating)
  - Επεξηγηματικά, καλογραμμένα σχόλια
  - Αποφυγή έξυπνων/σύνθετων τεχνικών και

Αρχές Καλού Στυλ

- Ο τρόπος κωδικοποίησης πρέπει να είναι συνεπής και συστηματικός: ευκολύνει ανάγνωση/διαμοιρασμό/ επαναχρησιμοποίηση/εξέλιξη του κώδικα
- Συχνά πρέπει να ακολουθήσετε συμβάσεις του οργανισμού στον οποίο εργάζεστε, αν όχι ο κώδικας σας πρέπει να ακολουθεί τυπικές συμβάσεις και καλές πρακτικές που υπάρχουν για τη Γ/Π στην οποία προγραμματίζετε.

```
class UserQueue {
 int noofItemsInQ, frontofTheQueue, queueCapacity;
 public intnoofUsersInQueue(){...}
}
```

queue.queueCapacity

```
class UserQueue {
  int nitems, front, capacity;
  public intnusers() {…}

 }
```

```
queue.capacity++;
n = queue. nusers ();
```

University of Cyprus
Department of Computer Science

M. D. Dikaiakos

Θέματα Στύλ Προγραμματισμού

- Στυλ προγραμματισμού
- Ονόματα
- Συνέπεια και ιδιωματισμοί
- Εκφράσεις
- Μαγικοί αριθμοί
- Σχόλια
- Οργάνωση αρχείων Java
- Javadoc

University of Cyprus
Department of Computer Science

M. D. Dikaiakos

Ονόματα - πώς τα διαλέγουμε;

- Το **όνομα** μιας προγραμματιστικής οντότητας (μεταβλητής, κλάσης, μεθόδου, κλπ) περιγράφει την οντότητα και εκφράζει πληροφορίες για το σκοπό και τη σημασία της. Ένα όνομα πρέπει να είναι:
  - Πληροφοριακό
  - Περιεκτικό
  - Ευμνημόνευτο
  - Εύηχο.

Ονόματα - πώς τα διαλέγουμε;

- Πολλές πληροφορίες για τη σημασία μιας προγραμματιστικής οντότητας προέρχονται από το συγκείμενο (**context**) και το πεδίο εμβέλειας/εφαρμογής (**scope**) του ορισμού της οντότητας.
- Όσο **ευρύτερο** είναι το συγκείμενο/πεδίο εμβέλειας μιας οντότητας, τόσο πιο περιγραφικό πρέπει να είναι το όνομα της:
  - Χρησιμοποιείτε περιγραφικά ονόματα για οντότητες με μεγάλο πεδίο εμβέλειας και συντομότερα ονόματα για μεταβλητές με τοπική χρήση.
  - Η σαφήνεια συχνά πετυχαίνεται μέσω της συντομίας.

Camel-case σύνταξη ονομάτων

- Η Java ακολουθεί τη σύνταξη camel-case για την ονομασία των κλάσεων, των διεπαφών, των μεθόδων και των μεταβλητών.
- Εάν το όνομα συνδυάζεται με δύο λέξεις, η δεύτερη λέξη θα ξεκινά πάντα με κεφαλαία γράμματα, όπως maxMarks( ), lastName, ClassTest, αφαιρώντας όλα τα κενά.
- Εξαιρούνται τα ονόματα σταθερών (βλ. πιο κάτω)

Ποια ονόματα προγραμματιστικών οντοτήτων αναμένεται να είναι μεγαλύτερα και πιο περιγραφικά;

00:20 00:40

- Ονόματα τοπικών μεταβλητών
- Ονόματα ιδιωτικών μεθόδων
- Ονόματα γενικών μεταβλητών
- Ονόματα βιβλιοθηκών

Ονόματα Μεταβλητών (name of variables)

- Περιγραφικά και Ευμνημόνευτα
- Επιλογή γίνεται ώστε να υποδεικνύουν στον περιστασιακό αναγνώστη την πρόθεση χρήσης.
- Ονόματα μεταβλητών με 1-2 χαρακτήρες αποφεύγονται, εκτός από «προσωρινές» μεταβλητές με τοπική χρήση.
- Κοινά ονόματα για τις προσωρινές μεταβλητές είναι:
  - `i, j, k, m` και `n` για ακέραιους αριθμούς
  - `c, d` και `e` για χαρακτήρες.
- Τα ονόματα `i, j` για δείκτες βρόχων (loop indices), `p, q` για δείκτες/χειριστήρια και `s, t` για αλφαριθμητικά είναι συνήθης, ώστε δεν κερδίζουμε χρησιμοποιώντας πιο μεγάλα ονόματα για μεταβλητές με αυτό τον ρόλο.

Ονόματα Μεταβλητών: συμβάσεις Java

- **Ολες οι μεταβλητές** (στιγμιοτύπου, στατικές, τοπικές - εκτός των σταθερών) ξεκινούν πάντοτε με πεζό πρώτο γράμμα.
- Αν το όνομα μιας μεταβλητής είναι σύνθετη λέξη δύο ή περισσοτέρων λέξεων, οι εσωτερικές λέξεις πρέπει να ξεκινούν με κεφαλαίο γράμμα.

- Τα ονόματα των μεταβλητών **δεν** πρέπει να ξεκινούν με χαρακτήρες υπογράμμισης "_" ή σύμβολο δολαρίου "$", παρόλο που επιτρέπονται και τα δύο.
- **Σταθερές**:
  - Το όνομα μεταβλητής που δηλώνεται ως σταθερά πρέπει να χρησιμοποιεί αποκλειστικά κεφαλαία γράμματα.
  - Αν το όνομα είναι σύνθεση δύο ή περισσοτέρων λέξεων, οι λέξεις πρέπει να διαχωρίζονται με χαρακτήρα υπογράμμισης "_".

https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html

Ονόματα Κλάσεων/Διαπροσωπειών Java

- Κλάσεις (classes):
  - Το όνομα μιας κλάσης πρέπει να είναι ουσιαστικό
  - Ξεκινάει πάντοτε με κεφαλαίο γράμμα - τα υπόλοιπα γράμματα της λέξης είναι πεζά
  - Αν είναι σύνθετη λέξη, κάθε εσωτερική λέξη του ονόματος πρέπει επίσης να ξεκινάει με κεφαλαίο γράμμα, ενώ τα υπόλοιπα πρέπει να είναι πεζά.
  - Το όνομα πρέπει να είναι απλό και περιγραφικό.
  - Αποφύγετε ακρωνύμια και συντομογραφίες.
- Διαπροσωπείες (interfaces):
  - Ισχύουν ίδιοι κανόνες με τις κλάσεις.

Ονόματα Μεθόδων/Συναρτήσεων

- Περιγραφικά και Ευμνημόνευτα
- Ξεκινούν με πεζό γράμμα
- Επιλέγετε ονόματα που υποδηλώνουν ενέργεια και βασίζονται σε ενεργητικά ρήματα, ακολουθούμενα ίσως από ουσιαστικά:

```
now = date.getTime(); putchar('\n');
```

- Μέθοδος/συνάρτηση που επιστρέφει τιμή boolean (true or false) πρέπει να υποδηλώνει αυτή την ιδιότητα στο όνομα της για να μην υπάρχει σύγχιση. Π.χ.: `if (isoctal(c))`

Ποια από τις ακόλουθες επιλογές ονομάτων είναι προτιμότερη;

00:20 00:40

```
for (theElementIndex = 0; theElementIndex < numberofElements;
      theElementIndex++)
  elementArray[theElement:Index] = theElementIndex ;

for (i = 0; i < nelems; i++) uelem[i]= I;
```


Ονόματα Βιβλιοθηκών Java (packages)

- Το πρόθεμα ενός ονόματος πακέτου Java γράφεται πάντα με πεζούς ASCII χαρακτήρες και θα πρέπει να είναι ένα από τα ονόματα πεδίου (domain names) ανώτατου επιπέδου (com, edu, gov, mil, net, org ή ένας από τους αγγλικούς κωδικούς δύο γραμμάτων ταυτοποίηση χωρών ISO 3166).
- Τα επόμενα στοιχεία του ονόματος πακέτου ποικίλλουν ανάλογα με τις εσωτερικές συμβάσεις ονομασίας ενός οργανισμού. Τέτοιες συμβάσεις μπορεί να προσδιορίζουν ότι ορισμένα στοιχεία ονομάτων καταλόγου είναι ονόματα τμήματος, τμήματος, έργου, Η/Υ ή ονόματος πρόσβασης (login).
- Π.χ. `com.sun.eng`, `com.apple.quicktime.v2`, `edu.cmu.cs.bovik.cheese`

Θέματα Στύλ Προγραμματισμού

University of Cyprus
Department of Computer Science

- Στυλ προγραμματισμού
- Ονόματα
- Συνέπεια και ιδιωματισμοί
- Εκφράσεις
- Μαγικοί αριθμοί
- Σχόλια
- Οργάνωση αρχείων Java
- Javadoc

Συνέπεια

- Η συνέπεια στο προγραμματιστικό στυλ οδηγεί σε καλύτερα/πιο ευανάγνωστα και ευκολοτέρως συντηρούμενα προγράμματα.
- Εάν η μορφοποίηση ποικίλλει απρόβλεπτα, π.χ.
  - Ένας βρόχος πάνω σε έναν πίνακα κάποτε διατρέχει τα περιεχόμενα σε αυξανόμενη κατεύθυνση των δεικτών και κάποτε σε μειούμενη
  - Αντιγράφονται αλφαριθμητικά (strings) άλλοτε με βρόχο και άλλοτε με μέθοδο,

οι παραλλαγές καθιστούν πιο δύσκολο να δούμε τι πραγματικά κάνει ο κώδικας.

Στοίχιση και Εσοχές (identation)

- Αξιοποιείστε κατάλληλη στοίχιση για παρουσίαση της λογικής δομής του κώδικα σας.
  - Ένα σταθερό στυλ στοίχιση είναι ο ευκολότερος τρόπος για να γίνεται αυτονόητη η δομή ενός προγράμματος.

```
for(n++;n<100;field[n++]='\0');
*i = '\0'; return('\n');
```

```
for (n++; n < 100; n++)
    field[n] = '\0';
*i = '\0';
return '\n';
```

```
for (n++; n < 100; field[n++] = '\0')
    ;
*i = '\0';
return('\n');
```


Ιδιωματισμοί (idioms)

- Όπως οι φυσικές γλώσσες, έτσι και οι Γ/Π έχουν ιδιωματισμούς:
  - Συμβάσεις που ακολουθούν οι πεπειραμένοι προγραμματιστές για να γράφουν συνήθη αποσπάσματα κώδικα.
- Αφορούν σε:
  - Σύνταξη βρόχων
  - Στοίχιση και εσοχές
  - Κενές γραμμές και διάκενα ανάμεσα σε εκφράσεις, τελεστές, εντολές
  - Οργάνωση και στοίχιση εντολών ελέγχου (if-then-else)

https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html

Idioms for Loops

```
i = 0;
while (i <= n-1)
    array[i++] = 1.0;
```

```
for (i = 0; i < n; )
    array[i++] = 1.0;
```

```
for (i = n; --i >= 0; )
    array[i] = 1.0;
```

```
for (i = 0; i < n; i++)
    array[i] = 1.0;
```

```
for (int i = 0; i < n; i++)
    array[i] = 1.0;
```

University of Cyprus
Department of Computer Science

Θέματα Στύλ Προγραμματισμού

- Στυλ προγραμματισμού
- Ονόματα
- Συνέπεια και ιδιωματισμοί
- Εκφράσεις
- Μαγικοί αριθμοί
- Σχόλια
- Οργάνωση αρχείων Java
- Javadoc

University of Cyprus
Department of Computer Science

M. D. Dikaiakos

Εκφράσεις (expressions)

- Τι είναι οι εκφράσεις (expressions) σε μια Γ/Π;
- Γράφετε τις εκφράσεις στη φυσική τους μορφή, όπως θα τις εκφωνούσατε μεγαλόφωνως πιο εύκολα.
- Οι λογικές εκφράσεις που περιλαμβάνουν αρνήσεις είναι πάντα δύσκολο να κατανοηθούν.

```
if (!(block_id < actblks) || !(block_id >= unblocks))
```

```
if ((block_id >= actblks) || (block_id < unblocks))
```


Εκφράσεις (expressions)

- Χρησιμοποιείστε παρενθέσεις για να αποφύγετε την ασάφεια, ακόμα και όταν η χρήση τους δεν απαιτείται λόγω των κανόνων προτεραιότητας.
- Διασπάστε τις σύνθετες/συμπαγείς εκφράσεις σε πιο απλές, ευανάγνωστες.

```
*x += (*xp=(2*k < (n-m) ? c[k+1] : d[k--]));
```

```
if (2*k < n-m)
    *xp = c[k+1];
else
    *xp = d[k--];
*x += *xp;
```

Παρενέργειες (side-effects)

- Τι είναι οι παρενέργειες (πηγή: stackoverflow):
  - Παρενέργεια είναι ο,τιδήποτε κάνει μια μέθοδος ή έκφραση, εκτός από τον υπολογισμό και την επιστροφή μιας τιμής.
- Π.χ., οποιαδήποτε αλλαγή των τιμών ενός πεδίου δεδομένων κάποιου αντικειμένου είναι παρενέργεια, όπως και η σχεδίαση στην οθόνη, η εγγραφή σε ένα αρχείο ή μια σύνδεση δικτύου.
- Οι παρενέργειες μπορούν επίσης να προκαλέσουν προβλήματα διότι συχνά οι προγραμματιστές παραγνωρίζουν την παρουσία τους. Π.χ.

```
array[i++] = i;
```


Μαγικοί Αριθμοί (magic numbers)

- Είναι σταθερές (constants), μήκη πινάκων (array sizes), θέσεις χαρακτήρων (character positions), παράμετροι μετατροπής (conversion factors) και άλλες αριθμητικές τιμές που χρησιμοποιούνται και εμφανίζονται στα προγράμματα
- Γενικός κανόνας: **οποιοσδήποτε αριθμός εκτός από το 0 και το 1, είναι *«μαγικός»* και πρέπει να του δίνεται κάποιο όνομα και να χρησιμοποιείται μέσω του ονόματος αυτού:**
  - Ένας αριθμός που εμφανίζεται σε κάποιο πηγαίο κώδικα δεν δίνει οποιαδήποτε ένδειξη για τη σημασία του ή για το πώς έχει προκύψει, καθιστώντας το πρόγραμμα πιο δύσκολο στην κατανόηση.
  - Η χρήση του 0 δεν πρέπει να γίνεται όταν π.χ. θέλουμε να ελέγξουμε αν μια τιμή (χειριστήριο αντικειμένου) είναι `null`.
  - Δεν χρησιμοποιούμε τις ακέραιες τιμές κωδικοποίησης χαρακτήρων, όταν π.χ. θέλουμε να κάνουμε ελέγχους για τιμές και ιδιότητες μεταβλητών που περιέχουν χαρακτήρες.

Έλεγξε αν ένας χαρακτήρας είναι κεφαλαίο γράμμα

```
if (c >= 65 && c <= 90)
```

```
if (c >= 'A' && c <= 'Z')
```

```
if (Character.isUpperCase(c))
```

University of Cyprus
Department of Computer Science

Ενότητα 1: Εισαγωγή στον Α/Σ Προγραμματισμό

Διάλεξη 6 - 9/2/24

Σχεδιασμός Κλάσεων

University of Cyprus
Department of Computer Science

Αντικείμενα και Κλάσεις στις Γ.Π.

- Βασικά συστατικά στοιχεία των προγραμμάτων Α/Σ Γ.Π. είναι οι **κλάσεις**: μια μορφή «τύπων», όπως τα struct της C, ενισχυμένων όμως με λειτουργικότητες («συμπεριφορές»), κώδικα, έλεγχο πρόσβασης σε πεδία δεδομένων κλπ.
- Οι κλάσεις, χρησιμοποιούνται κυρίως για να κατασκευάζουμε **αντικείμενα**, τα οποία αποτελούν και τη βασική δομή δεδομένων του Α/Σ Προγραμματισμού.
  - Κατά την εκτέλεση του, ένα αντικειμενοστρεφές προγράμμα μπορεί να ιδωθεί σαν ένα σύνολο αντικειμένων που αλληλεπιδρούν μεταξύ τους εξυπηρετώντας το ένα το άλλο.

Κλάσεις

- Στον πραγματικό κόσμο υπάρχουν αντικείμενα που ανήκουν στην ίδια ***κατηγορία*** - είναι του ίδιου **τύπου**.
  - Π.χ. έχουμε χιλιάδες ποδήλατα της ίδιας μάρκας και του ίδιου μοντέλου, τα οποία έχουν κατασκευαστεί με βάση το ίδιο σχέδιο κι έχουν τα ίδια συστατικά στοιχεία.
- Στον Α/Σ προγραμματισμό, λέμε ότι ένα αντικείμενο-ποδήλατο, αποτελεί στιγμιότυπο (instance) μια κλάσης αντικειμένων που είναι γνωστά ως ποδήλατα.
- Η **κλάση** είναι το σχεδιότυπο/σχεδιάγραμμα (ο τύπος) από το οποίο κατασκευάζονται τα αντικείμενα-ποδήλατα.
- Η κλάση καθορίζει την κατάσταση και τη συμπεριφορά των αντικειμένων της.

Κατάσταση, Συμπεριφορά, Ταυτότητα

- Τα αντικείμενα στον Α/Σ προγραμματισμό έχουν τρία βασικά χαρακτηριστικά:
  - Κατάσταση (state)
    - Τα δεδομένα που αποθηκεύονται στο εσωτερικό του αντικειμένου
  - Συμπεριφορά (behavior)
    - Οι ενέργειες που μπορεί να διεκπεραιώσει το αντικείμενο: συναρτήσεις που διαθέτει το αντικείμενο
  - Ταυτότητα (identity)
    - Το “κλειδί” μέσω του οποίου μπορούμε να αποκτήσουμε πρόσβαση στο αντικείμενο

Ποιά η σχέση ιδιοτήτων και καταστάσεων ενός αντικειμένου;

00:20 00:40

- **Οι ιδιότητες του αντικειμένου είναι υποσύνολο των καταστάσεων στις οποίες μπορεί να βρεθεί το αντικείμενο**
- **Οι καταστάσεις είναι υποσύνολο των ιδιοτήτων του αντικειμένου**
- **Δεν υπάρχει σχέση μεταξύ ιδιοτήτων και καταστάσεων**
- **Μια κατάσταση αντιστοιχεί σε συνδυασμούς τιμών**

Ποιά από τις παρακάτω προτάσεις είναι σωστή;

00:40 00:20

- **Η κλάση είναι ένα σύνολο αντικειμένων με κοινές ιδιότητες και συμπεριφορές**
- **Η κλάση είναι ένα σχεδιότυπο απαράλλακτων αντικειμένων**
- **Το αντικείμενο είναι σχεδιότυπο· η κλάση είναι στιγμιότυπο**
- **Καμιά από τις πιο πάνω**

Ποιά από τις παρακάτω προτάσεις είναι σωστή;

00:40 00:20

- **Ένα αντικείμενο μπορεί να έχει ως ιδιότητα ένα άλλο αντικείμενο**
- **Η τιμή μιας ιδιότητας αντικειμένου μπορεί να είναι άλλο αντικείμενο**
- **Ένα αντικείμενο αποτελείται μόνο από ιδιότητες ή μόνο από συμπεριφορές**
- **Τίποτε από τα πιο πάνω**

Ποιά από τις παρακάτω προτάσεις είναι σωστή;

00:40 00:20

- **Ιδιότητα του αυτοκινήτου που βλέπετε είναι το «χρώμα πράσινο».**
- **Ιδιότητα του αυτοκινήτου που βλέπετε είναι το «χρώμα»**
- **Ιδιότητα του αυτοκινήτου που βλέπετε είναι: «μάρκα Citroen 2CV»**
- **Καμιά από τις πιο πάνω**

Πεδία Δεδομένων κ Μέθοδοι

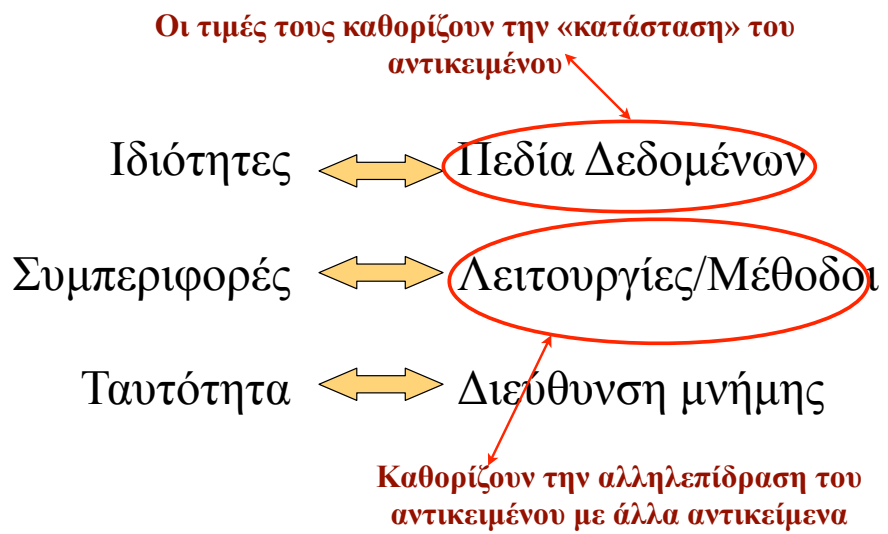
Ένα αντικείμενο λογισμικού:

- Αποθηκεύει στο εσωτερικό του πληροφορίες που είναι αποτέλεσμα των υπολογισμών-λειτουργιών του.
  - Η αποθήκευση γίνεται στα πεδία δεδομένων του αντικειμένου(fields/variables).
  - Οι αποθηκευμένες πληροφορίες αποτελούν την κατάσταση (state) του αντικειμένου.
- Δίνει πρόσβαση στη συμπεριφορά του μέσω μεθόδων (functions in some programming languages). Οι μέθοδοι:
  - λειτουργούν πάνω στην εσωτερική κατάσταση ενός αντικειμένου
  - αποτελούν τον βασικό μηχανισμό για επικοινωνία ανάμεσα σε αντικείμενα.

Ταυτότητα αντικειμένων

- Η ταυτότητα ενός αντικειμένου λογισμικού: identifier (προσδιοριστής) μέσω του οποίου αποκτούμε πρόσβαση στο αντικείμενο.
- Αντιστοιχεί στη διεύθυνση στην οποία βρίσκεται αποθηκευμένο το αντικείμενο
- Στη Java, ο προσδιοριστής ενός αντικείμενου λέγεται χειριστήριο (handle) και μέσω αυτού:
  - αποκτούμε πρόσβαση στα πεδία του αντικειμένου (κατάσταση)
  - μπορούμε να ενεργοποιήσουμε τις ενέργειες του

Συστατικά στοιχεία Αντικειμένου



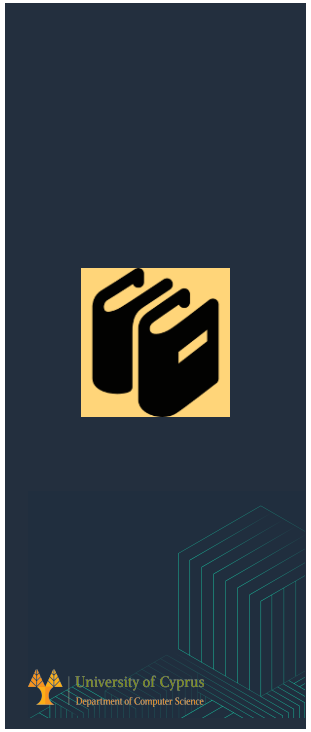
Πολυπλοκότητα αντικειμένων

- Η πολυπλοκότητα της κατάστασης και της συμπεριφοράς αντικειμένων διαφορετικού τύπου μπορεί να διαφέρει. Π.χ.:
  - Ένας σκύλος έχει κατάσταση (name, color, breed, hungry) και συμπεριφορά (barking, fetching, wagging tail).
  - Ένα ποδήλατο επίσης έχει κατάσταση (current gear, current pedal cadence, current speed) και συμπεριφορά (changing gear, changing pedal cadence, applying brakes).
- Ο καθορισμός της κατάστασης και της συμπεριφοράς αντικειμένων του πραγματικού κόσμου είναι ένας πολύ καλός τρόπος για να ξεκινήσετε να σκέφτεστε με αντικειμενοστρεφή

Ενότητα 1: Εισαγωγή στον Α/Σ Προγραμματισμό

Ορισμοί Κλάσεων Java

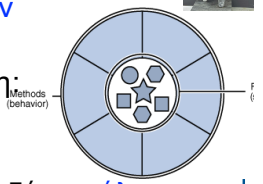
University of Cyprus
Department of Computer Science



• κεφ. 4.1, Savitch

Αντικείμενο

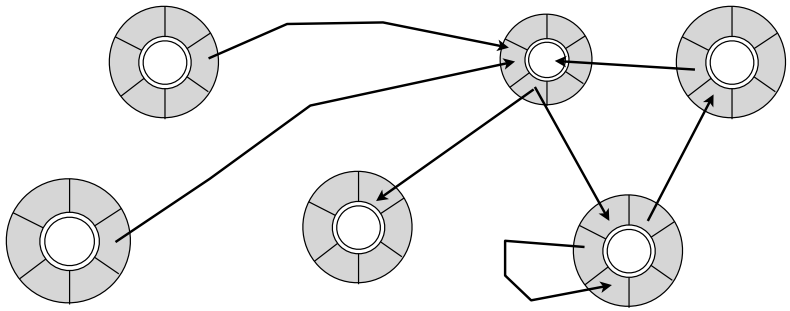
- Η βασική αφαιρετική αναπαράσταση δεδομένων (data abstraction) του Α/Σ Προγραμματισμού.
  - "Κάθε δεδομένο είναι αντικείμενο" (εξαίρεση: οι μεταβλητές αρχέγονου τύπου)



- Κάθε αντικείμενο περιλαμβάνει/περικλείει τα εξής «μέλη»:
  - Πεδία Δεδομένων ή Μεταβλητές Στιγμιοτύπου (fields or instance variables)
  - Μεθόδους (methods)
- Η πρόσβαση στις μεθόδους και στα πεδία δεδομένων ενός αντικειμένου γίνεται μέσω:
  - Χειριστηρίου (handle), το οποίο αντιστοιχεί στη διεύθυνση μνήμης στην οποία αποθηκεύεται το αντικείμενο

Α/Σ επίλυση προβλημάτων

- Ένα πρόγραμμα JAVA κωδικοποιεί την επίλυση ενός υπολογιστικού προβλήματος σαν σύνολο από αντικείμενα που *αλληλεπιδρούν*.



Ποιά από τις παρακάτω προτάσεις είναι σωστή;

- **Η κλάση είναι ένα σύνολο αντικειμένων με κοινές ιδιότητες και συμπεριφορές**
- **Η κλάση είναι ένα σχεδιότυπο πανομοιότυπων αντικειμένων**
- **Το αντικείμενο είναι σχεδιότυπο· η κλάση είναι στιγμιότυπο**
- **Καμιά από τις πιο πάνω**

Ποιά η σχέση ιδιοτήτων και καταστάσεων ενός αντικειμένου;

- **Οι ιδιότητες του αντικειμένου είναι υποσύνολο των καταστάσεων στις οποίες μπορεί να βρεθεί το αντικείμενο**
- **Οι καταστάσεις είναι υποσύνολο των ιδιοτήτων του αντικειμένου**
- **Δεν υπάρχει σχέση μεταξύ ιδιοτήτων και καταστάσεων**
- **Οι καταστάσεις αντιστοιχούν στο σύνολο των**

Περίγραμμα

University of Cyprus
Department of Computer Science

Κλάση: A Java Class Is a Type (τύπος δεδομένων)

- Η **κλάση** είναι τύπος δεδομένου (*programmer-defined type*), και συγκεκριμένα το σχεδιότυπο/σχεδιάγραμμα από το οποίο κατασκευάζονται τα αντικείμενα (*instances of the class*).
- Μια κλάση ορίζει τα *πεδία δεδομένων* και τις *μεθόδους* των αντικειμένων που προκύπτουν από αυτή.
- Μηχανισμός για σχεδιασμό και παραγωγή αντικειμένων
- Αντικείμενα που ανήκουν στην ίδια κλάση:
  - Υποστηρίζουν την ίδια συλλογή λειτουργιών (συμπεριφορών)
  - Έχουν ένα κοινό σύνολο από πιθανές καταστάσεις

Δήλωση κλάσεων στη Java

- Ο τύπος των αντικειμένων στην JAVA καθορίζεται με τον ορισμό κλάσεων.
  - Π.χ: **class** ATypeName { /* class body */}
- Η δήλωση μιας κλάσης περιλαμβάνει τους ορισμούς των **μελών** της:
  - Πεδία δεδομένων της κλάσης (data items) ή fields ή μεταβλητές στιγμιοτύπου (instance variables).
  - Μεθόδους (methods)

Παρατηρήσεις

- Όλα τα αντικείμενα της ίδιας κλάσης έχουν τις **ίδιες μεθόδους**
- Όλα τα αντικείμενα της ίδιας κλάσης έχουν τα **ίδια πεδία δεδομένων** (όνομα, τύπος, αριθμός) αλλά πιθανόν με διαφορετικές τιμές.
- Κάθε αντικείμενο διατηρεί την δική του ξεχωριστή μνήμη για τα δικά του πεδία δεδομένων.
- Instance variable declarations and method definitions *can be placed in any order within the Java class definition* (*η σειρά εμφάνισης δεν παίζει ρόλο*)
- Εκτός από τις κλάσεις (τύποι που ορίζονται από τον προγραμματιστή), η Java υποστηρίζει και αρχέγονους τύπους (primitive types).

Αρχέγονοι τύποι

Type	Description	Default	Size	Example Literals
boolean	true or false	FALSE	1 bit	`true, false`
byte	twos complement integer	0	8 bits	whole numbers from -128 to 127
char	Unicode character	`\u0000`	16 bits	`'a', '\u0041', '\101', '\\', '\'', '\n', 'ß'`
short	twos complement integer	0	16 bits	-32,768 to 32,767
int	twos complement integer	0	32 bits	`-2, -1, 0, 1, 2`
long	twos complement integer	0	64 bits	`-2L, -1L, 0L, 1L, 2L`
float	IEEE 754 floating point	0.0	32 bits	`1.23e100f, -1.23e-100f, .3f, 3.14F`
double	IEEE 754 floating point	0.0	64 bits	`1.23456e300d, -1.23456e-300d, 1e1d`

Δημιουργία αντικειμένων

- Αντικείμενα μιας κλάσης δημιουργούνται με χρήση της εντολής new σε συνδυασμό με τον κατασκευαστή (constructor) της κλάσης.
- Μετά την δήλωση ενός χειριστηρίου, μπορεί να γίνει η αρχικοποίηση/διασύνδεση του με κάποιο δημιουργηθέν αντικείμενο:

```
ATypeName a = new ATypeName();
String s = new String("asdf");
String s = "asdf";
BankAcct b1 = new BankAcct(21338, 0.0);
ATypeName b = a;
```


Πρόσβαση στα πεδία δεδομένων

- Πεδία δεδομένων αντικειμένων DataOnly:

```
class DataOnly {
  int i;
  float f;
  GasContainer s;
}
```

- Δήλωση χειριστηρίου, δημιουργία αντικειμένου, αρχικοποίηση χειριστηρίου:

```
DataOnly d = new DataOnly();
```

- Αρχικοποίηση πεδίων αντικειμένου:

```
d.i = 47;
d.f = 1.1f;
d.s = new GasTanks("Hello");
d.s.leftTank.capacity = 100;
```

Ενότητα 1: Εισαγωγή στον Α/Σ Προγραμματισμό

Μέθοδοι

University of Cyprus
Department of Computer Science

Μέθοδοι στη JAVA

- Οι μέθοδοι της JAVA είναι οι γνωστές συναρτήσεις-διαδικασίες του διαδικασιακού προγραμματισμού.
- Καθορίζουν τις υπευθυνότητες/συμπεριφορές/ πιθανές ενέργειες των αντικειμένων μιας κλάσης.
- Μια μέθοδος JAVA δηλώνεται *μόνο σαν στοιχείο κάποιας κλάσης* - δεν υπάρχουν «γενικές» συναρτήσεις.
  - Μια μέθοδος μπορεί να κληθεί μόνο μέσω κάποιου αντικειμένου *[ή κλάσης]*, το οποίο να την εμπεριέχει.
  - Αν προσπαθήσετε να καλέσετε κάποια μέθοδο που δεν ενυπάρχει σε ένα αντικείμενο, θα πάρετε μήνυμα λάθους από τον μεταφραστή.

Κατηγορίες Μεθόδων

- void methods: Μέθοδοι που επιτελούν μια ενέργεια (perform an action): δεν επιστρέφουν μια τιμή. Τις χρησιμοποιούμε για τις παρενέργειές τους (side-effects).
- Μέθοδοι που υπολογίζουν και επιστρέφουν μια τιμή καποιου τύπου (compute and return a **value** of a certain type)
- Η κλήση μιας μεθόδου (invocation) που επιστρέφει ιοποιηθεί σαν ρεί να

```
System.out.println(c1);
```

```
public int random(){..}
```

```
public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }
```



```
public class ChargeClient {
   public static void main(String[] args) {
          double x = Double.parseDouble(args[0]);
          double y = Double.parseDouble(args[1]);

          Charge c1 = new Charge(.51, .63, 21.3);
          Charge c2 = new Charge(.13, .94, 81.9);
          System.out.println(c1);
          System.out.println(c2);

          double v1 = c1.potentialAt(x, y);
          double v2 = c2.potentialAt(x, y);
          System.out.println(v1+v2);
      }
   }
```

Η κλήση μεθόδων γίνεται διαμέσου των αντικειμένων).
Η κλήση μεθόδου προκαλεί την εκτέλεση των εντολών στο σώμα της μεθόδου.

Τιμές επιστροφής μεθόδων

- Μια μέθοδος που **επιστρέφει τιμή** πρέπει να προσδιορίζει τον τύπο αυτής της τιμής στην επικεφαλίδα της:

```
public typeReturned methodName(paramList)
```

- Μια μέθοδος που **δεν επιστρέφει τιμή** χρησιμοποιεί τη λέξη-κλειδί **`void`** στην επικεφαλίδα της για να δείξει ότι:

```
public void methodName(paramList)
```


Περιεχόμενο (σώμα) μεθόδου

- Το σώμα των μεθόδων περιλαμβάνει κατάλογο:
  - Δηλώσεων (declarations) και
  - εντολών (statements)

που περικλείονται από ένα ζευγάρι αγκύλες.

```
public <void or typeReturned> myMethod(){
    declarations (δηλώσεις)
    statements (εντολές)
}
```

Body (σώμα μεθόδου)

Copyright © 2017 Pearson Ltd. All rights reserved.

Η εντολή `return`

- Το σώμα μεθόδου που επιστρέφει τιμή πρέπει να περιέχει **μία ή περισσότερες** εντολές επιστροφής `return`
- Η εντολή `return` καθορίζει την τιμή που επιστρέφεται και τερματίζει την κλήση της μεθόδου: **`return Expression;`**
  - Expression (έκφραση): μπορεί να είναι οποιαδήποτε έκφραση που αξιολογείται σε τιμή του τύπου που αναφέρεται στην επικεφαλίδα της μεθόδου ότι επιστρέφεται

Η εντολή `return`

- Μια μέθοδος **`void`** δεν χρειάζεται να περιέχει **`return`**, **εκτός εάν**
  - υπάρχει μια κατάσταση που απαιτεί *τη λήξη της μεθόδου πριν εκτελεστεί όλος ο κώδικας της*
- Σε αυτή την περίπτωση, δεδομένου ότι δεν επιστρέφεται τιμή, η εντολή **`return`** χρησιμοποιείται *χωρίς έκφραση*:
  - **`return;`**

Κάθε μέθοδος μπορεί να χρησιμοποιηθεί σαν `void`

- Μια μέθοδος που επιστρέφει τιμή μπορεί επίσης να εκτελέσει και διάφορες άλλες λειτουργίες για τις παρενέργειες τους
- Εάν δεν σας ενδιαφέρει η επιστρεφόμενη τιμή, μπορείτε να καλέσετε τη μέθοδο σαν να ήταν `void` και να αγνοήσετε την επιστρεφόμενη τιμή:
  - `objectName.returnedValueMethod();`

Copyright © 2017 Pearson Ltd. All rights reserved.

`main` is a `void` Method

- Ένα πρόγραμμα Java είναι απλώς μια κλάση που έχει μια μέθοδο `main`
- Όταν δίνετε εντολή εκτέλεσης ενός προγράμματος Java, το σύστημα εκτέλεσης καλεί τη μέθοδο `main`
- Σημειώστε ότι η `main` είναι μια μέθοδος `void`:

```
public static void main(String[] args)
```

Copyright © 2017 Pearson Ltd. All rights reserved.

Μεταβλητό πλήθος ορισμάτων: varargs

- Τα varargs είναι μια δυνατότητα της Java (μετά την έκδοση 5) που επιτρέπει σε μια μέθοδο να δέχεται έναν απροσδιόριστο αριθμό ορισμάτων του **ίδιου τύπου**.
- Σύνταξη: Χρήση των τριών τελείων (...) μετά τον τύπο της τυπικής παραμέτρου.
- Επιτρέπεται μόνο ένα vararg ανά μέθοδο, και πρέπει να είναι η τελευταία τυπική παράμετρος στον κατάλογο τυπικών παραμέτρων της μεθόδου.
- Τα Varargs απλοποιούν τον κώδικα και τον κάνουν πιο ευέλικτο.

Υλοποίηση varargs

- Τα varargs μετατρέπονται αυτόματα σε πίνακα του αντίστοιχου τύπου.
- Μπορούμε να έχουμε πρόσβαση στα στοιχεία του πίνακα με τον ίδιο τρόπο όπως και σε έναν κανονικό πίνακα.
- Τα varargs είναι χρήσιμα όταν δεν γνωρίζουμε εκ των προτέρων πόσα ορίσματα θα περάσουμε στη μέθοδο.
- Παράδειγμα:

```
public static double calculateAverage(double... numbers) {
        double sum = 0.0;
        for (double number : numbers) {
            sum += number;
        }
        return sum / numbers.length;
    }
```

Ενότητα 1: Εισαγωγή στον Α/Σ Προγραμματισμό

Παράμετροι, Ορίσματα, Τοπικές Μεταβλητές, Κλήση με Τιμή

University of Cyprus
Department of Computer Science

Τυπικές Παράμετροι (formal parameters)

- Οι **τυπικές παράμετροι** (**formal parameters**) μιας μεθόδου καθορίζουν τις πληροφορίες που πρέπει να περάσουμε στην μέθοδο όταν την καλέσουμε.

```
public class Charge {
    private double rx, ry;   // position
    private double q;        // charge

    public Charge(double x0, double y0, double q0) {
        rx = x0;
        ry = y0;
        q  = q0;
    }
    public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }

    public String toString() {
        return q + " at " + "(" + rx + ", " + ry + ")";
    } }
```

Τυπικές παράμετροι (formal parameters)

Πραγματικές Παράμετροι ή Ορίσματα

- Κατά την **κλήση μεθόδου**, στη θέση της κάθε τυπικής παραμέτρου εισάγουμε ένα ***όρισμα*** (argument), το οποίο μπορεί να είναι:
  - Χειριστήριο αντικειμένου του ίδιου τύπου με την τυπική παράμετρο
  - Μεταβλητή ή Τιμή αρχέγονου/πρωταρχικού τύπου (primitive value), συμβατού τύπου με την τυπική παράμετρο
- Τα ορίσματα αποκαλούνται επίσης πραγματικές παράμετροι (actual parameters).
- Ο αριθμός και η σειρά των ορισμάτων *πρέπει να ταιριάζει ακριβώς σε αυτά της λίστας παραμέτρων*
- Ο τύπος κάθε ορίσματος πρέπει να είναι **συμβατός** (*compatible*) με τον τύπο της αντίστοιχης παραμέτρου

```
int a=1,b=2,c=3;
double result = myMethod(a,b,c);
```


Copyright © 2017 Pearson Ltd. All rights reserved.

Συμβατότητα και μετασχηματισμός (αρχέγονων) τύπων

- Εάν οι τύποι ορισμάτων και παραμέτρων δεν ταιριάζουν ακριβώς, η Java θα προσπαθήσει να πραγματοποιήσει αυτόματο μετασχηματισμό τύπου (**automatic type conversion**)
- Ένα αρχέγονο όρισμα μπορεί να υποστεί αυτόματα μετασχηματισμό τύπου (**type casting**) από οποιονδήποτε από τους παρακάτω τύπους, σε οποιονδήποτε από τους τύπους που εμφανίζονται στα δεξιά:

```
byte→short→int→long→float→double
char ──────────↗
```

Τυπικές παράμετροι
(formal parameters)

```
public class Charge {
    private double rx, ry;    // position
    private double q;         // charge

    public Charge(double x0, double y0, double q0) {
        rx = x0;
        ry = y0;
        q  = q0;
    }
    public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }

    public String toString() {
        return q + " at " + "(" + rx + ", " + ry + ")";
    } }
```

Ορίσματα ή **πραγματικές παράμετροι**
(arguments ή actual parameters)

Χρήση όρων «Παράμετρος» και «Όρισμα»: Προσοχή!

- Παράμετρος (parameter) είναι η μεταβλητή
- Το όρισμα (argument) είναι η τιμή που περνιέται στην παράμετρο
- Οι όροι **παράμετρος** και **όρισμα** συχνά χρησιμοποιούνται εναλλακτικά
- Όταν τους συναντήσετε, ίσως χρειαστεί να προσδιορίσετε την ακριβή τους σημασία από τα συμφραζόμενα.

Copyright © 2017 Pearson Ltd. All rights reserved.

Ποιά από τις πιο κάτω προτάσεις είναι λάθος;

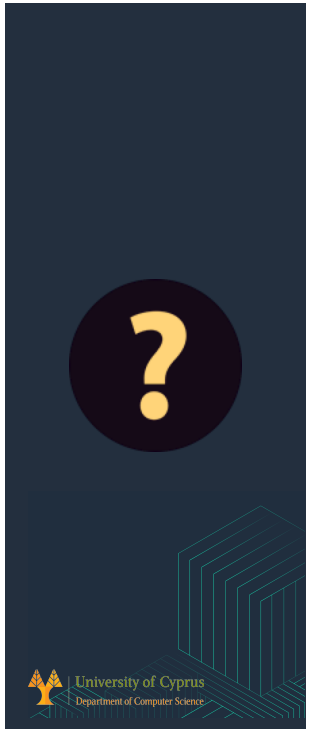
00:40 00:20

- **Η `main` είναι `void` μέθοδος της Java**
- **Κάθε μέθοδος Java μπορεί να χρησιμοποιηθεί σαν να ήταν `void`**
- **Στη `main` δεν μπορούμε να κάνουμε χρήση εντολής «`return;`»**
- **Η `main` δεν χρειάζεται να υπάρχει σε κάθε κλάση Java**

Κλήση με τιμή (call by value)

```
int a=1,b=2,c=3;
double result = myMethod(a,b,c);
```

- Στο παράδειγμα, είναι η τιμή του κάθε ορίσματος και όχι το όνομα της μεταβλητής, η οποία περνιέται στην αντίστοιχη παράμετρο μεθόδου.
- Αυτή η μέθοδος σύνδεσης ορισμάτων για τυπικές παραμέτρους είναι γνωστή ως μηχανισμός *κλήσης με τιμή* (*call-by-value mechanism*)



MOODLE ΚΟΥΙΖ ΓΙΑ ΤΟΠΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ, ΠΑΡΑΜΕΤΡΟΥΣ ΚΑΠ

M. D. Dikaiakos

Τοπικές μεταβλητές (local variables)

```
public class ChargeClient {

   public static void main(String[] args) {
          double x = Double.parseDouble(args[0]);
          double y = Double.parseDouble(args[1]);

          Charge c1 = new Charge(.51, .63, 21.3);
          Charge c2 = new Charge(.13, .94, 81.9);
          System.out.println(c1);
          System.out.println(c2);

          double v1 = c1.potentialAt(x, y);
          double v2 = c2.potentialAt(x, y);
          System.out.println(v1+v2);
      }
   }
```

Μια μεταβλητή που δηλώνεται στο **εσωτερικό μεθόδου** αποκαλείται *τοπική μεταβλητή*

Οι παράμετροι μεθόδων χρησιμοποιούνται σαν να ήταν τοπικές μεταβλητές.

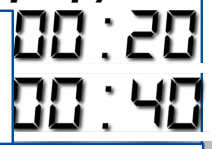
```
public class Charge {
    private double rx, ry;    // position
    private double q;         // charge

    public Charge(double x0, double y0, double q0) {
        rx = x0;
        ry = y0;
        q  = q0;
    }
    public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }

    public String toString() {
        return q + " at " + "(" + rx + ", " + ry + ")";
    } }
```


Παράμετροι μεθόδων κ. Τοπικές Μεταβλητές

- Οι παράμετροι συχνά θεωρούνται ως σύμβολα αναπλήρωσης (placeholders) που αντικαθίστανται από την τιμή του αντίστοιχου ορίσματος.
- Ωστόσο, στην πραγματικότητα μια παράμετρος είναι κάτι περισσότερο: έχει θέση και χρήση τοπικής μεταβλητής.
- Όταν κληθεί μια μέθοδος, υπολογίζονται οι τιμές των ορισμάτων της και οι τιμές αυτές χρησιμοποιούνται για να αρχικοποιηθούν οι αντίστοιχες παράμετροι (που είναι σαν τοπικές μεταβλητές).
- Ακόμα κι αν η τιμή μιας τυπικής παραμέτρου αλλάξει μέσα στο σώμα μιας μεθόδου, **η τιμή του ορίσματος δεν μπορεί να αλλάξει.**

Call-by-value semantics

Τι συμβαίνει κατά την εκτέλεση της



```
public class Silly {
 public double dummySum(double x, double y, int z) {
     z = 288;
     return x + y;
 }}
```

```
public class TestSilly {
 public static void main(String args[]){
   int zop = 133;
   Silly sillyObj = new Silly();
   System.out.println(sillyObj.dummySum(1, 2, zop) + " "+ zop);
}}
```

- **Εκτυπώνει: 0 0**
- **Εκτυπώνει: 3.0 zop**
- **Εκτυπώνει: 3.0 133**
- **Εκτυπώνει: 3.0 288**

Ορισμός Κλάσεων Java

Java Reflection

University of Cyprus
Department of Computer Science

Αντικείμενα τύπου class

- Κάθε αντικείμενο στη Java συνδέεται με ένα αντικείμενο τύπου Class το οποίο περιέχει όλες τις πληροφορίες για τον τύπο (κλάση) του αντικειμένου.
- Μπορείτε να αποκτήσετε πρόσβαση στο class object χρησιμοποιώντας διάφορες μεθόδους, όπως:

`Class.forName("MyClass")`

  - Φορτώνει την κλάση MyClass από το classpath και επιστρέφει το class object της `MyClass`.

`object.getClass()`

  - Επιστρέφει το class object της κλάσης από την οποία παρήχθει το `object`.

- Ένα class object μπορεί να χρησιμοποιηθεί για:
  - Ενδοσκόπηση (reflection): άντληση πληροφοριών για την κλάση.
  - Δημιουργία νέων αντικειμένων της κλάσης.
  - Εξέταση και να τροποποίηση της συμπεριφοράς της κλάσης.

Ενδοσκόπηση - Reflection

- Το reflection (ενδοσκόπηση) είναι δυνατότητα που παρέχει η Java (έκδοση 5 και μετά) και επιτρέπει σε ένα εκτελούμενο πρόγραμμα να εξετάζει δυναμικά και να χειρίζεται εσωτερικές ιδιότητες των στοιχείων του προγράμματος.
  - Για παράδειγμα, είναι δυνατό, για κάποια κλάση, το πρόγραμμα να λάβει ένα κατάλογο με τα ονόματα των μελών της.
- Μόλις έχετε μια Class object, μπορείτε να χρησιμοποιήσετε τις μεθόδους της για να λάβετε πληροφορίες για την κλάση, όπως τα πεδία, τις μεθόδους, και τους κατασκευαστές της.
- Μπορείτε επίσης να χρησιμοποιήσετε τις μεθόδους της Class object για να τροποποιήσετε την συμπεριφορά της κλάσης, όπως να καλέσετε private μεθόδους ή να τροποποιήσετε τα πεδία της.

```
import java.lang.reflect.*;

   public class DumpMethods {
      public static void main(String args[])
      {
         try {
            Class c = Class.forName(args[0]);
            Method m[] = c.getDeclaredMethods();
            for (int i = 0; i < m.length; i++)
            System.out.println(m[i].toString());
         }
         catch (Throwable e) {
            System.err.println(e);
         }
      }
   }
```



```
> java DumpMethods java.util.Stack


public java.lang.Object java.util.Stack.push(
 java.lang.Object)

public synchronized
     java.lang.Object java.util.Stack.pop()

public synchronized
      java.lang.Object java.util.Stack.peek()

public boolean java.util.Stack.empty()

public synchronized
     int java.util.Stack.search(java.lang.Object)
```


Χρήσεις Ενδοσκόπησης

- Δυναμική φόρτωση και δημιουργία κλάσεων: Μπορείτε να φορτώσετε κλάσεις από διάφορες πηγές (π.χ., αρχεία JAR) και να δημιουργήσετε αντικείμενα δυναμικά.
- Επιθεώρηση και τροποποίηση κλάσεων: Μπορείτε να λάβετε λεπτομερείς πληροφορίες για μια κλάση, όπως τα πεδία, τις μεθόδους, και τους κατασκευαστές της. Μπορείτε επίσης να τροποποιήσετε την συμπεριφορά της κλάσης, όπως να καλέσετε private μεθόδους ή να τροποποιήσετε τα πεδία της.
- Ανάπτυξη frameworks και βιβλιοθηκών: Η ενδοσκόπηση αποτελεί βασικό συστατικό για την ανάπτυξη frameworks και βιβλιοθηκών που προσφέρουν δυναμικές δυνατότητες.