Ενότητα 1: Εισαγωγή στον Α/Σ Προγραμματισμό

Μάθημα 9 - 20/2/24

Σύνδεση μεθόδου με αντικείμενο
Στοιχεία Αφαιρετικότητας Α/Σ Π
Ενδοσκόπηση μεθόδων

University of Cyprus
Department of Computer Science

Περιεχόμενα

- Τυπικές Παράμετροι Μεθόδων
- Ορίσματα Μεθόδων
- Κλήση με τιμή (call-by-value)
- Παράμετροι κλάσης
- **Σύνδεση μεθόδου με αντικείμενο (this)**
- Στοιχεία Αφαιρετικότητας Α/Σ Πρ.
- Ενδοσκόπηση κλάσης

University of Cyprus
Department of Computer Science

Ενότητα 1: Εισαγωγή στον Α/Σ Προγραμματισμό

Σύνδεση σώματος μεθόδου με αντικείμενο κλήσης: «this»

University of Cyprus
Department of Computer Science

```
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 + ")";
    }
}
```

Η παράμετρος this

```
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);
    }
```

- Σε κάθε χρήση μεταβλητής στιγμιοτύπου (instance variable) υπονοείται ότι ενυπάρχει παραπομπή στο αντικείμενο που περιέχει τη μεταβλητή:

  `<object handle>.identifier`

- Αν χρειαστούμε να αναφερθούμε ρητά στο περικλείον αντικείμενο, μπορούμε να χρησιμοποιήσουμε την κλειδολέξη (keyword) **this**

Η παράμετρος this

- **myInstanceVariable** always means and is always interchangeable with **this.myInstanceVariable**

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


Η παράμετρος this

- Η **this πρέπει** να χρησιμοποιείται **υποχρεωτικά**, εάν στο σώμα της μεθόδου ορίζεται παράμετρος ή τοπική μεταβλητή με το ίδιο όνομα με μια μεταβλητή στιγμιοτύπου της κλάσης
  - Διαφορετικά, όλες οι εμφανίσεις του ονόματος της μεταβλητής θα ερμηνεύονται ως τοπικές

```
int someVariable = this.someVariable
```

Η παράμετρος this

```
class Banana {
  double param;

  Banana(int prm) {
    param = prm;
  }

  void f(int i) {
    System.out.println("Calc: " + i * param);
  }
}
```

```
Banana a = new Banana(5), b = new Banana(7);
a.f(1);
b.f(2);
```

Η χρήση του `this` (συνέχεια)

- Πως μπορεί η μέθοδος `f()` να γνωρίζει αν καλείται από το αντικείμενο **a** ή το αντικείμενο **b**;
  - `a.f(1)` ⇔ `Banana.f(a,1)`
  - `b.f(2)` ⇔ `Banana.f(b,2)`
- Η παράμετρος `this` είναι «κρυμμένη»
- Αν και δεν εμφανίζεται στη λίστα τυπικών παραμέτρων μιας μεθόδου, αυτομάτως και υπόρρητα περνιέται σαν όρισμα στο σώμα της μεθόδου:
  - Όταν κληθεί μια μέθοδος σε ένα αντικείμενο, το σώμα της μεθόδου συνδέεται αυτόματα με τη `this`, η οποία παραπέμπει στο αντικείμενο της κλήσης.

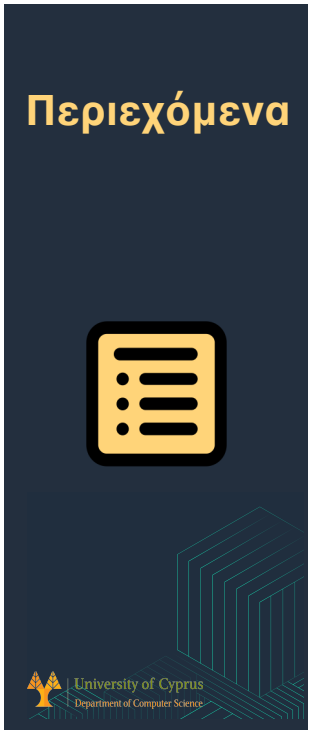
Σύνοψη: η παράμετρος `this`

- Αν θέλουμε, μέσα από το σώμα κάποιου αντικειμένου, να αποκτήσουμε πρόσβαση-χειριστήριο προς το ίδιο το αντικείμενο, μπορούμε να χρησιμοποιήσουμε την ειδική μεταβλητή `this`, η οποία είναι χειριστήριο για το αντικείμενο μας.
- Ανάθεση στην `this` δεν επιτρέπεται.
- Μέσω της `this` μπορούμε να περάσουμε το τρέχον αντικείμενο σαν παράμετρο σε μεθόδους άλλων αντικειμένων.

Παράδειγμα χρήσης this

```
// Simple use of the "this" keyword.
public class Leaf {
  private int i = 0;
  Leaf increment()
   i++;
   return this;
  }
  void print() { System.out.println("i = " + i); }
  public static void main(String[] args) {
   Leaf x = new Leaf();
   x.increment().increment().increment().print();
  }
}
```

- **Πολλαπλή κλήση της ίδιας μεθόδου πάνω στο ίδιο αντικείμενο.**



Ενότητα 1: Εισαγωγή στον Α/Σ Προγραμματισμό

Στοιχεία Αφαιρετικότητας Α/Σ Προγραμματισμού

University of Cyprus
Department of Computer Science

Στοιχεία αφαιρετικότητας στον Α/Σ.Π.

- ✓ Αντικείμενα: οι λύσεις σε υπολογιστικά προβλήματα που μοντελοποιούνται σαν αλληλεπιδράσεις μεταξύ αντικειμένων.
- ✓ Κλάσεις: είναι το «εργαλείο» δημιουργίας σχεδιοτύπων και παραγωγής αντικείμενων με ομοειδή χαρακτηριστικά.
3. Απόκρυψη πληροφοριών (information hiding).
4. Εμφώλευση (encapsulation).
5. Χαμηλή διασυνδετικότητα.

Απόκρυψη Πληροφορίας (Information Hiding)

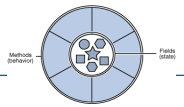
- Η πρακτική διαχωρισμού της **χρήσης στοιχείων μιας κλάσης** από τις **λεπτομέρειες υλοποίησης της κλάσης**.
- Μέσω της απάλειψης λεπτομερειών μειώνεται η πληροφοριακή υπερφόρτωση των προγραμματιστών:
  - Δεν μας ενδιαφέρει πώς ακριβώς ένα αντικείμενο εκπληρώνει μια υποχρέωση του.
  - Δεν χρειάζεται να γνωρίζουμε τις λεπτομέρειες διεκπεραίωσης μιας λειτουργίας (μεθόδου) από ένα αντικείμενο: τι ενέργειες θα κάνει και από ποια αντικείμενα θα ζητήσει εξυπηρέτηση για να πραγματοποιήσει τη λειτουργία.
  - Από την στιγμή που το αντικείμενο αποδεχθεί ένα «μήνυμα»-αίτημα, αποδέχεται και την ευθύνη για την διεκπεραίωση του.
- Βασική προσέγγιση Α/Σ. Π.: «αναζητούμε» κάποιο άλλο αντικείμενο για να **μεταβιβάσουμε** ολόκληρη ή μέρος της υποχρέωσης που έχουμε αναλάβει

Εμφώλευση (Encapsulation)

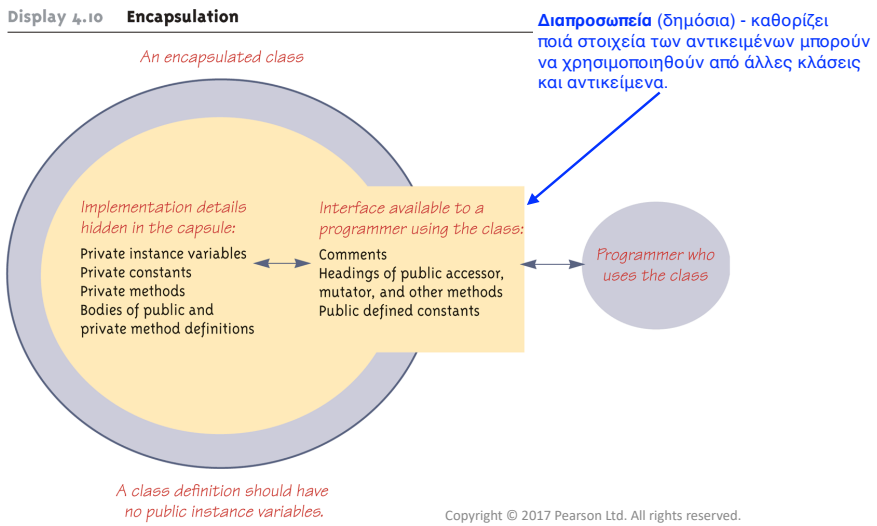
- **Εμφώλευση** ή **εγκόλπωση** σημαίνει ότι δεδομένα και μέθοδοι μιας κλάσης συνδυάζονται σε μια ενιαία οντότητα (αντικείμενο), η οποία:
  - Συγκεντρώνει λειτουργίες και χαρακτηριστικά
  - Δύναται να «αποκρύβει» από τον «έξω κόσμο» τις λεπτομέρειες της υλοποίησής του
- Η γνώση λεπτομερειών υλοποίησης δεν είναι απαραίτητη γιατί η αλληλεπίδραση με το αντικείμενο πραγματοποιείται μέσω μιας καλά καθορισμένης και απλής διεπαφής/διαπροσωπείας.

Εμφώλευση δεδομένων (data encapsulation)

- Στη Java, η «απόκρυψη» λεπτομερειών υλοποίησης επιτυγχάνεται με την δήλωση των λεπτομερειών αυτών ως **ιδιωτικών - private.**
- Απόκρυψη της κατάστασης (state) του αντικειμένου σε ιδιωτικά (private) πεδία δεδομένων.
- Περιορισμός της αλληλεπίδρασης/επικοινωνίας με τα αντικείμενα και δυνατότητα αλλαγής της κατάστασης των μόνο μέσω των μεθόδων τους.
- Τα αντικείμενα διατηρούν τον έλεγχο του πώς ο εξωτερικός κόσμος μπορεί να τα αξιοποιήσει.



Εμφώλευση



Προσδιοριστές public κ. private

- Ο προσδιοριστής πρόσβασης (modifier) **public** (δημόσιος)
  - καθορίζει ότι δεν υπάρχουν περιορισμοί στο ποια άλλη κλάση μπορεί να χρησιμοποιήσει μια δημόσια μεταβλητή στιγμιοτύπου ή μέθοδο της κλάσης μας.
- Ο προσδιοριστής πρόσβασης **private** (ιδιωτικός)
  - υποδηλώνει ότι μια μεταβλητή στιγμιοτύπου ή μέθοδος της κλάσης μας **δεν** μπορεί να προσπελασθεί μέσω του ονόματός της (δεν είναι «ορατή») από άλλες κλάσεις (*cannot be accessed by name outside of the class)*

Βέλτιστες Πρακτικές (best practices)

- **Όλες** οι μεταβλητές στιγμιοτύπου πρέπει να είναι **private**
- Οι περισσότερες μέθοδοι είναι **public**
  - Μέσω αυτών δίνουμε *ελεγχόμενη πρόσβαση* στις ιδιωτικές μεταβλητές στιγμιοτύπου
- Συνήθως, μέθοδοι που δηλώνονται **private** χρησιμοποιούνται σαν υποβοηθητικές (helper methods) σε άλλες μεθόδους της κλάσης τους.

Πρόσβαση στα ιδιωτικά μέλη κλάσης από αντικείμενα της κλάσης

- Μια κλάση έχει πρόσβαση σε ιδιωτικά μέλη όλων των αντικειμένων της.
- Μέσα από το σώμα μιας κλάσης, μπορούμε να έχουμε πρόσβαση σε ιδιωτικά μέλη οποιουδήποτε αντικειμένου της κλάσης, όχι μόνο ιδιωτικά μέλη του καλούντος αντικειμένου.

Μέθοδοι Προσπέλασης και Μεταλλαγής

- *Accessor ή getter* methods (μέθοδοι **προσπέλασης**) καθορίζονται για την ανάγνωση (λήψη) της τιμής των μεταβλητών στιγμιότυπου ενός αντικειμένου.
  - Τα δεδομένα είναι προσβάσιμα αλλά δεν πρέπει να αλλάξουν από τη μέθοδο προσπέλασης
  - Το όνομα μιας μεθόδου accessor συνήθως ξεκινά με τη λέξη ***get***
- *Mutator or setter* methods (μέθοδοι **μεταλλαγής**) ορίζονται για την αλλαγή της τιμής των μεταβλητών παρουσίας ενός αντικειμένου με ελεγχόμενο τρόπο
  - Τα εισερχόμενα δεδομένα συνήθως ελέγχονται και/ή φιλτράρονται
  - Το όνομα μιας μεθόδου μετάλλαξης ξεκινά συνήθως με τη λέξη ***set***


```
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 + ")";
    }
    public boolean equals(Charge c) {
          return (c.q == q);
        }

    public getXCoord()       { return this.rx; }
    public getYCoord()       { return this.ry; }
    public getChargeValue() { return this.q;  }
    public setPosition(double x, double y) {
           this.rx = x;
           this.ry = y;
    }
}
```

Προσπέλασης - Accessor

Μεταλλαγής - Mutator

Επιστροφή boolean από μέθοδο μεταλλαγής

- Κάποιες μέθοδοι μεταλλαγής εκτυπώνουν μήνυμα σφάλματος και τερματίζουν το πρόγραμμα αν τους δοθούν τιμές που δεν είναι έγκυρες/αποδεκτές
- Εναλλακτική προσέγγιση: ελέγξετε την εγκυρότητα των τιμών στο σώμα της μεθόδου, αλλά:
  - Μην τερματίσετε το πρόγραμμα
  - Επιστρέψετε μια boolean τιμή και αφήστε το καλούν πρόγραμμα να χειριστεί τις περιπτώσεις όπου οι επιδιωκώμενη μεταλλαγή δεν έχουν νόημα

Προϋποθέσεις and Μετασυνθήκες

- Η προϋπόθεση/ συνθήκη πριν την κλήση (*precondition*) μιας μεθόδου δηλώνει τι θεωρείται αληθές όταν καλείται η μέθοδος.
- Η μετά-συνθήκη / συνθήκη μετά την κλήση (*postcondition*) μιας μεθόδου δηλώνει τι θα ισχύει μετά την εκτέλεση της μεθόδου, εφόσον ισχύει η προϋπόθεση.

Καλές πρακτικές

- Είναι καλή πρακτική να σκέφτεστε πάντα με όρους και προϋποθέσεις όταν σχεδιάζετε μια μέθοδο.
- Είναι καλή πρακτική να περιγράφεται η προϋπόθεση και η μετασυνθήκη στα σχόλια της μεθόδου.

A Couple of Important Acronyms: API and ADT

- The API or *application programming interface* for a class is a description of how to use the class
  - A programmer need only read the API in order to use a well designed class
- An ADT or *abstract data type* is a data type that is written using good information-hiding techniques

Διασυνδετικότητα

- Με την απόκρυψη πληροφοριών, μπορούμε να πετύχουμε χαμηλή διασυνδετικότητα στα προγράμματά μας
- Ορισμός διασυνδετικότητας:
  - Ο βαθμός εξάρτησης ενός τμήματος του κώδικα κάποιου λογισμικού συστήματος από τα υπόλοιπα τμήματα.
- Σημασία διασυνδετικότητας
  - Θεωρείστε κάποιον κώδικα που υλοποιεί ουσιώδη λειτουργία ενός πολύπλοκου συστήματος λογισμικού.
  - Έφ' όσον ο κώδικας αυτός χρησιμεύει σε άλλα κομμάτια του συστήματος, στοιχεία του (μεταβλητές, αποτελέσματα, εντολές) θα χρησιμοποιούνται και πρέπει να είναι γνωστά και προσβάσιμα για την υλοποίηση των υπολοίπων κομματιών του λογισμικού.
  - Άρα, η ανάπτυξη ενός νέου υποσυστήματος προϋποθέτει την κατανόηση και γνώση του κώδικα μας (καλό ή κακό;)

Απουσία Παρεμβολών (non-interference)

- Εφόσον ένα αντικείμενο παρέχει τις υπηρεσίες που θέλουμε, δεν μας ενδιαφέρει **το πως** διεκπεραιώνει τα αιτήματα μας.
- Τα **εξαρτήματα (δομοστοιχεία) λογισμικού** (software components) σε Α/Σ συστήματα χαρακτηρίζονται από τις υπηρεσίες που παρέχουν και όχι από τις λεπτομέρειες της υλοποίησής τους.
- Η προσέγγιση αυτή διευκολύνει την συνεργασία πολλών προγραμματιστών πάνω στο ίδιο σύστημα με την ελάχιστη παρεμβολή του ενός στη δουλειά του άλλου.
- Ask not what you can do *to* your data structures; ask what your data structures can do *for* you.

Ποιά από τις πιο κάτω προτάσεις είναι η σωστή;

- **Η χαμηλή διασυνδετικότητα είναι ιδιότητα του Α/Σ. Π.**
- **Η υποστήριξη της εμφώλευσης επιτρέπει την απόκρυψη πληροφορίας, η οποία διευκολύνει τη χαμηλή διασυνδετικότητα.**
- **Η χρήση αντικειμένων διασφαλίζει την απόκρυψη πληροφορίας.**
- **Η απουσία απόκρυψης πληροφορίας συνεπάγεται την υψηλή διασυνδετικότητα.**

Περιεχόμενα

University of Cyprus
Department of Computer Science

Ενότητα 1: Εισαγωγή στον Α/Σ Προγραμματισμό

Ενδοσκόπηση - 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.