4 Συλλογές Αντικειμένων

Πώς χειριζόμαστε αντικείμενα σε ομάδες με επανάληψη

Η Απαίτηση Συλλογών Αντικειμένων

- Πολλές εφαρμογές χρειάζονται πλήθος αντικειμένων:
  - Κατάλογος βιβλίων
  - Φοιτητολόγιο
  - Πελατολόγιο
- Το πλήθος των εγγραφών μεταβάλλεται *δυναμικά* από προσθήκες και τις διαγραφές.
- Δεν είναι λύση να ορίσουμε μια κλάση με πολλά πεδία -ένα για κάθε εγγραφή-, διότι
  - ο αριθμός τους μένει σταθερός κατά την εκτέλεση,
  - πρέπει να γνωρίζουμε το πλήθος κατά τη συγγραφή του προγράμματος.

96

Ένα Σημειωματάριο

- Αποθηκεύει σημειώσεις ως συμβολοσειρές.
- Προβάλλει τις καταγραμμένες σημειώσεις μια-μια.
- Δέχεται απεριόριστο πλήθος σημειώσεων.
- Ενημερώνει πόσες σημειώσεις έχει καταγράψει.
- Πώς θα υλοποιήσουμε μια συλλογή αντικειμένων (σημειώσεων);

97

Βιβλιοθήκες Κλάσεων

- **Επαναχρησιμοποίηση κώδικα** (code reuse)
  - Το καλό το παλικάρι, δεν προσπαθεί να ανακαλύψει τον τροχό ...
  - ... παίρνει έτοιμο, δοκιμασμένο κώδικα
- Είναι σημαντικό να προγραμματίζεις λίγο.
  - Λιγότερος κόπος, λιγότερα λάθη, καλύτερες επιδόσεις
- Η Java -όπως και άλλες αντικειμενοστρεφείς γλώσσες- έχει μια εκτεταμένη **βιβλιοθήκη κλάσεων** (class library).

98

Χρήση Βιβλιοθηκών

- Για να χρησιμοποιήσουμε μια κλάση από τη βιβλιοθήκη πρέπει να το δηλώσουμε με την εντολή **`import`**.

```
import java.util.Vector;
import java.util.*;
```

- Αυτές οι δηλώσεις τοποθετούνται στην αρχή της κλάσης μας.
- Μετά μπορούμε να χρησιμοποιούμε την κλάση σα να ήταν κομμάτι του προγράμματός μας.
  - Για τις κλάσεις των βιβλιοθηκών δημοσιεύεται ο τρόπος χρήσης τους, πχ οι υπογραφές των μεθόδων τους, αλλά *όχι* ο πηγαίος τους κώδικας.

99

Κλάση Βιβλιοθήκης `ArrayList`

- Υλοποιεί μια **συλλογή** (collection): αντικείμενο που αποθηκεύει ομάδα αντικειμένων.

```
import java.util.ArrayList;

public class Notebook {

    private ArrayList notes;

    /**
     * Κατασκευαστής σημειωματάριου
     */
    public Notebook() {
        notes = new ArrayList();
    }
    ...
}
```

δήλωση ότι θα χρησιμοποιηθεί κλάση βιβλιοθήκης

δήλωση μεταβλητής στιγμιότυπου

κατασκευή στιγμιότυπου

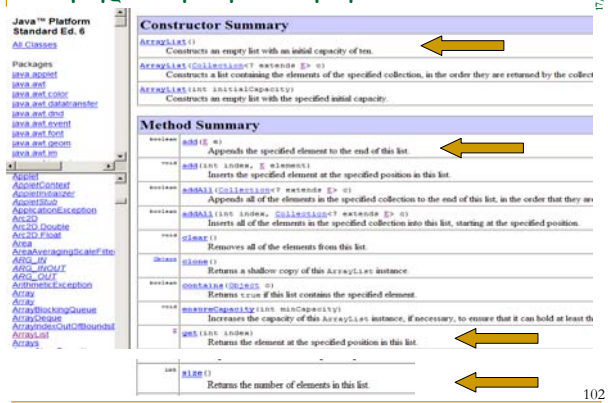
Project: **notebook1**: κατασκευή, inspect, επιθεώρηση κώδικα, αρίθμηση στοιχείων

100

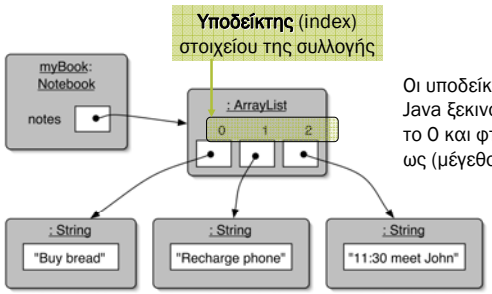
Υλοποίηση Σημειωματάριου

```
import java.util.ArrayList;

public class Notebook {

  private ArrayList notes;        // αποθηκευτικός χώρος για συλλογή

  public Notebook()  {            //κατασκευαστής σημειωματάριου
    notes = new ArrayList();      //κατασκευάζει συλλογή
  }

  public void storeNote(String note) {
    notes.add(note);              //προσθήκη αντικειμένου στη συλλογή
  }

  public int numberOfNotes() {
    return notes.size();          //το μέγεθος της συλλογής
  }

  public void showNote(int noteNumber) { //προβάλλει σημείωση με υποδείκτη
    if ((noteNumber >= 0) && (noteNumber < numberOfNotes())) {
      System.out.println(notes.get(noteNumber));
    }
  }
}
```


Τεκμηρίωση Βιβλιοθήκης



Διάγραμμα Αντικειμένων Συλλογής



Οι υποδείκτες στη Java ξεκινούν από το 0 και φτάνουν ως (μέγεθος – 1).

Χαρακτηριστικά της Συλλογής

- Αυξάνει τη χωρητικότητά της, όσο χρειάζεται.
- Διατηρεί τα αντικείμενα σε σειρά.
- Οι λεπτομέρειες της υλοποίησής της είναι κρυφές.
  - Ε, και; Μπορούμε να τη χρησιμοποιούμε ακόμη και χωρίς να γνωρίζουμε πώς λειτουργεί.
  - Αρκεί να ξέρουμε να τη χειριζόμαστε (να κατασκευάζουμε αντικείμενα και να καλούμε μεθόδους της).
  - (Μεταξύ μας, η `ArrayList` υλοποιείται με έναν πίνακα που έχει υποδείκτες 0..9 και εισάγει νέα στοιχεία στο τέλος. Όταν σβήσει κάποιο στοιχείο, συμπληρώνει το κενό. Το μέγεθός της αυξομειώνεται δυναμικά κατά τις απαιτήσεις προσθηκών / διαγραφών.)

Εκχώρηση Καθηκόντων

```
public class Notebook {
  ...
  public void storeNote(String note) {
    notes.add(note);
  }

  public int numberOfNotes() {
    return notes.size();
  }

  public void showNote(int noteNumber) {
    if ((noteNumber >= 0) && noteNumber < numberOfNotes())) {
      System.out.println( notes.get(noteNumber) );
    }
  }
  ...
}
```

Μέθοδος ανάγνωσης

Ουσιαστικά, η κλάση μας δεν κάνει και πολλά, απλώς καλεί τις μεθόδους της `ArrayList` για να κάνουν τη δουλειά.

Διαγραφή Σημείωσης

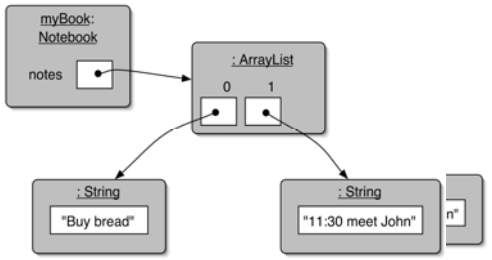
- Ελέγχεται η εγκυρότητα του αριθμού σημείωσης (υποδείκτη). Χρησιμοποιεί μέθοδο της συλλογής.

```
public void removeNote(int noteNumber) {
  if ((noteNumber >= 0) &&
      (noteNumber < numberOfNotes())) {
    notes.remove(noteNumber);
  } //end if
} //end method
```

- Αλλάζουν οι υποδείκτες των στοιχείων της συλλογής που ακολουθούν για να παραμείνουν συνεχόμενοι και να κλείσουν την ‘τρύπα’.

Ιωάννης Γαβιώτης
gaviotis@aegean.gr
http://www.syros.aegean.gr/users/gaviotis/tmp2

Αλλαγή Υποδεικτών Συλλογής



Ανασκόπηση

- Οι συλλογές διατηρούν αυθαίρετο πλήθος αντικειμένων.
- Οι βιβλιοθήκες κλάσεων περιέχουν ελεγμένο και δοκιμασμένο κώδικα που μπορούμε να χρησιμοποιούμε στα προγράμματά μας.
- Χρησιμοποιήσαμε την κλάση `ArrayList`:
  - Στοιχεία προστίθενται & διαγράφονται.
  - Κάθε στοιχείο έχει έναν υποδείκτη.
  - Οι υποδείκτες αλλάζουν με τη διαγραφή στοιχείων.
  - Οι κυριότερες μέθοδοι της `ArrayList` είναι οι: `add`, `get`, `remove` και `size`.

Επανάληψη

- Συχνά θέλουμε να εκτελούμε εντολές για αυθαίρετο αριθμό επαναλήψεων.
- Οι περισσότερες γλώσσες προγραμματισμού έχουν **εντολές επανάληψης** (loop).

```
/**
 * τύπωσε όλες τις σημειώσεις του σημειωματάριου.
 */
public void listNotes()
{
    int index = 0;
    while (index < notes.size()) {
        System.out.println(notes.get(index));
        index++ ;
    }
}
```

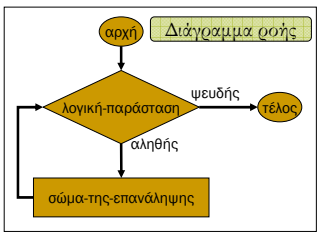
τελεστής αύξησης

Εντολή `while`

Σύνταξη

```
while ( λογική-παράσταση ) {
    σώμα-της-επανάληψης
}
```

Διάγραμμα ροής



Σημασιολογία

1. Εκτίμησε την παράσταση.
2. Αν είναι ψευδής, μετάφερε τον έλεγχο στην εντολή που έπεται της while.
3. Αν είναι αληθής, εκτέλεσε τον κώδικα στο σώμα της επανάληψης και αμέσως μετά πήγαινε στο βήμα 1.

Απαριθμητής

- **Απαριθμητής** (iterator) είναι ένα αντικείμενο μέσω του οποίου μπορούμε να διατρέχουμε όλα τα στοιχεία μιας συλλογής.
- Υλοποιείται από την κλάση βιβλιοθήκης `java.util.Iterator`
- Έχει δύο κύριες μεθόδους:
  - `hasNext()` που ελέγχει αν υπάρχουν κι άλλα στοιχεία
  - `next()` που φέρνει το επόμενο στοιχείο στον απαριθμητή
- Όλες οι συλλογές υποστηρίζουν απαριθμητές.
- Οι απαριθμητές επιτρέπουν επανάληψη χωρίς υποδείκτες.
- Δεν μπορείς να διατρέξεις όλες τις συλλογές με υποδείκτες.

Απαρίθμηση Συλλογής

δήλωση της κλάσης βιβλιοθήκης του απαριθμητή

```
import java.util.ArrayList;
import java.util.Iterator;
...
  public void listNotes() {
    Iterator it = notes.iterator();

    while (it.hasNext()) {
      System.out.println(it.next());
    } //end while

  } //end method
...
```

δήλωση και κατασκευή του απαριθμητή

υπάρχει επόμενο στοιχείο στη συλλογή;

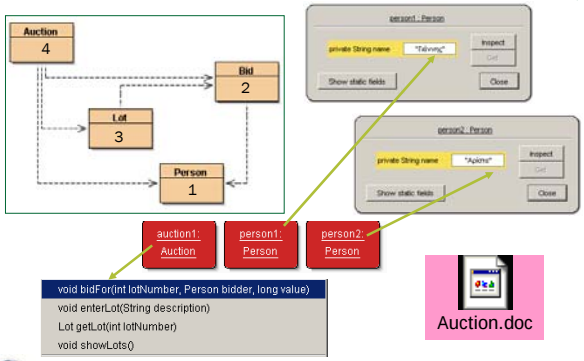
φέρε το επόμενο στοιχείο της συλλογής

Ιωάννης Γαβιώτης
gaviotis@aegean.gr
http://www.syros.aegean.gr/users/gaviotis/tmp2

Παράδειγμα: Πλειστηριασμός

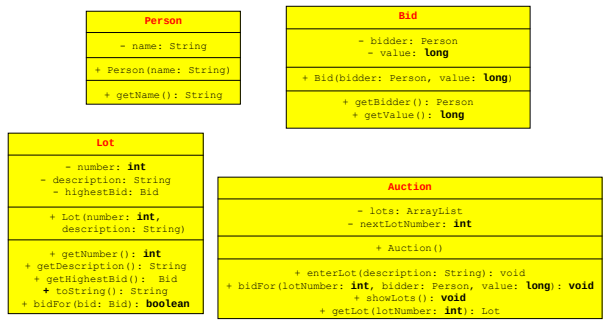
- Σε ένα πλειστηριασμό, αγαθά τίθενται προς πώληση.
- Άνθρωποι υποβάλουν προσφορές για κάποιο αγαθό.
- Όταν τελειώσει ο πλειστηριασμός, κάθε αγαθό για το οποίο έχουν γίνει προσφορές, πωλείται στον άνθρωπο που έχει προσφέρει τα περισσότερα.
- Κλάσεις / Ιδιότητες
  - Άνθρωπος (Person) έχει όνομα
  - Προσφορά (Bid) έχει άνθρωπο και αξία
  - Αγαθό (Lot) έχει κωδικό, περιγραφή, και καλύτερη προσφορά μέχρι στιγμής
  - Πλειστηριασμός (Auction) έχει συλλογή αγαθών

Πώς Λειτουργεί Ένας Πλειστηριασμός



Project: **auction**: κατασκευή Person (x2), Auction,

Διάγραμμα Κλάσεων Πλειστηριασμού



Project: auction: κατασκευή auction, 2 persons, inspect, λειτουργία

Κλάση για Ανθρώπους

```
/**
 * Τηρεί τα στοιχεία ανθρώπου που συμμετέχει σε πλειστηριασμό
 */
public class Person {

  private final String name; //το όνομα αυτού του ανθρώπου

  /**
   * Κατασκευάζει άνθρωπο με συγκεκριμένο όνομα.
   */
  public Person(String name) { //το this διακρίνει την ιδιότητα
    this.name = name;          //από την ομώνυμη παράμετρο
  }

  /**
   * Επιστρέφει το όνομα αυτού του ανθρώπου (μέθοδος ανάγνωσης)
   */
  public String getName() {
    return name;
  }

} //end class Person
```

δεν επιτρέπεται να αλλάξει τιμή (σταθερά)

Κλάση για Προσφορές

```
public class Bid {

  private final Person bidder; // ο άνθρωπος που κάνει την προσφορά
  private final long value; // η αξία αυτής της προσφοράς

  /**
   * Κατασκευάζει μια προσφορά.
   */
  public Bid(Person bidder, long value) {
    this.bidder = bidder;
    this.value = value;
  }

  public Person getBidder() {
    return this.bidder;
  }

  public long getValue() {
    return this.value;
  }

} //end class Bid
```

ένα αντικείμενο ως παράμετρος

εδώ η χρήση του **this** δεν είναι υποχρεωτική, αλλά είναι καλή συνήθεια, γιατί δείχνει ότι αναφερόμαστε σε μια ιδιότητα.

Κλάση για Αγαθά: Υποβολή Προσφοράς

```
public class Lot {
   private final int number;   // κωδικός του αγαθού
   private String description; // περιγραφή
   private Bid highestBid;     // μεγαλύτερη προσφορά μέχρι στιγμής
   ...

   /**
    * Υποβλήθηκε προσφορά. Δες αν είναι η καλύτερη μέχρι στιγμής.
    */
   public boolean bidFor(Bid bid) {
     if( ( highestBid == null ) ||   // δεν έχει γίνει άλλη προσφορά
         // έχει γίνει άλλη προσφορά και είναι μικρότερη
         ( highestBid.getValue() < bid.getValue() ) )
     {
       highestBid = bid; // θέσε την ως καλύτερη προσφορά για το αγαθό
       return true;
     }
     else
       return false;
   }

   ...
}
```

σημαίνει κενό σε μια θέση που μπαίνει αντικείμενο

Ιωάννης Γαβιώτης
gaviotis@aegean.gr
http://www.syros.aegean.gr/users/gaviotis/tmp2

Κλάση Πλειστηριασμού: Προσφορά

```
public class Auction {
  private ArrayList lots; // συλλογή των αγαθών
  private int nextLotNumber; // αριθμός επόμενου αγαθού
  ...
  public void bidFor(int lotNumber, Person bidder, long value) {
    // ανακάλεσε την προσφορά γι αυτό το υλικό
    Lot selectedLot = getLot(lotNumber);
    if (selectedLot != null) { // δεν έχει υποβληθεί προσφορά
      if (selectedLot.bidFor( new Bid(bidder, value) ))
        System.out.println("Νέα προσφορά για " + lotNumber);
      else
        System.out.println("Το αγαθό με αριθμό " + lotNumber +
          " έχει ήδη προσφορά " +
          selectedLot.getHighestBid().getValue() );
    } // end if
  } //end bidFor
  ...
}
```

Καλεί την `getValue` επί του αντικειμένου που επιστρέφει η `getHighestBid`, που καλείται επί του αντικειμένου `selectedLot`.

Κατασκευάζει νέο αντικείμενο για προσφορά και το θέτει ως καλύτερη προσφορά για το αγαθό (**ανώνυμο αντικείμενο**).

Κλάση Πλειστηριασμού: Αποτέλεσμα

Μετατρέπει το αντικείμενο που επιστρέφει η `next` σε `Lot`.

```
public void showLots() {
  Iterator it = lots.iterator();
  while ( it.hasNext() ) {
    Lot lot = (Lot) it.next();
    System.out.print(lot.getNumber() + ": " +
                     lot.getDescription());
    // καλύτερη προσφορά για το εμπόρευμα
    Bid highestBid = lot.getHighestBid();
    if ( highestBid != null )
      System.out.println(" Προσφορά: " +
                         highestBid.getValue());
    else
      System.out.println(" (Καμιά προσφορά)");
  }
}
```


Εκμαγείο

- Όπου γίνεται ανάθεση, ή αντιστοίχιση πραγματικής σε τυπική παράμετρο, οι τύποι δεδομένων πρέπει να είναι συμβατοί.
- Μετατροπές τύπων μπορούμε να επιβάλουμε με ένα **εκμαγείο** (cast).
- Σύνταξη
  (*τύπος-δεδομένων* **)** *παράσταση*
- Σαν αποτέλεσμα η παράσταση αλλάζει τύπο και μετατρέπεται στην «αντίστοιχη» τιμή του τύπου δεδομένων μέσα στο εκμαγείο.
- Πχ `int i=1, j=2;`
  `i/j` → `0`
  `(float) i/j` → `0.5` - γιατί;
  `(float) (i/j)` → ;

Επεκτάσεις στον Πλειστηριασμό

E1: Προσθέστε μέθοδο `close` στην κλάση `Auction`. Θα διαπερνά τη συλλογή και θα τυπώνει λεπτομέρειες για τα αγαθά. Κάθε αγαθό για το οποίο έχει γίνει μια τουλάχιστον προσφορά, θα πωλείται. Θα τυπώνεται το όνομα του πλειοδότη και η αξία της προσφοράς.

E2: Προσθέστε ελέγχους ορθότητας τιμών στις παραμέτρους.

E3: Προσθέστε στην κλάση `Auction` μέθοδο με υπογραφή **`public`** `ArrayList getUnsold()`
Θα διαπερνά τη συλλογή των αγαθών και θα αποθηκεύει τα απούλητα σε μια άλλη συλλογή. Στο τέλος της μεθόδου θα επιστρέφει τη συλλογή αυτή.

E4: Τροποποιήστε το πρόγραμμα, ώστε να μπορεί να τεθεί τιμή εκκίνησης για κάθε αγαθό.

Συλλογές Σταθερού Μεγέθους

- Όταν το μέγεθος μιας συλλογής είναι γνωστό εκ των προτέρων και δεν αλλάζει κατά την εκτέλεση του προγράμματος, χρησιμοποιούμε πίνακες (array).
- Οι πίνακες στη Java αποθηκεύουν τιμές από τους βασικούς τύπους ή αντικείμενα.
  - Οι συλλογές αποθηκεύουν μόνο αντικείμενα.
- Σύγκριση πινάκων – συλλογών
  - συλλογές έχουν μεταβαλλόμενο μέγεθος
  - πίνακες έχουν καλύτερες επιδόσεις στην ανάκληση
  - πίνακες αποθηκεύουν πιο οικονομικά τους βασικούς τύπους δεδομένων

Σύνταξη Δήλωσης / Κατασκευής Πίνακα

- *τύπος-βάσης* **[]** *όνομα-πίνακα* **;**
  δηλώνει μια μεταβλητή που μπορεί να δεχθεί αντικείμενα πίνακα με συγκεκριμένο τύπο βάσης. Παραδείγματα:
  ```
  double[] mesesThermokrasies;
  String[] onomateponyma;
  TicketMachine[] ekdotes;
  ```
- *όνομα-πίνακα* **= new** *τύπος-βάσης* **[** *διάσταση* **];**
  κατασκευάζει πίνακα συγκεκριμένης διάστασης (ακέραιος αριθμός) και τύπου βάσης
  ```
  mesesThermokrasies = new double[12];
  onomateponyma = new String[100];
  ekdotes = new TicketMachine[5];
  ```


Ιωάννης Γαβιώτης
gaviotis@aegean.gr
http://www.syros.aegean.gr/users/gaviotis/tmp2

Λειτουργίες σε Πίνακες

- Δήλωση μεταβλητής και κατασκευή
```
String[] nhsia;
nhsia = new String[20];
```
- Αποθήκευση τιμής σε θέση
```
nhsia[0] = "Σύρα";
nhsia[7] = "Χίος";
```
- Ανάσυρση τιμής από θέση
```
String enaNhsi = nhsia[0];
```
- Μέγεθος πίνακα
```
int meg = nhsia.length ;
```

Οι πίνακες είναι αντικείμενα.

ιδιότητα του αντικειμένου

Παράδειγμα: Ανάλυση Χρήσης Ανά Ώρα

- Σε ένα αρχείο καταγράφονται χρονοσφραγίδες επισκέψεων σε ιστοσελίδες.
- Η κλάση `LogAnalyzer` διαβάζει τις καταγραφές και υπολογίζει το πλήθος των επισκέψεων ανά ώρα.

Project: weblog-analyzer: κατασκευή, κλήση μεθόδων, inspect hourCounts

Δήλωση Μεταβλητών Τύπου Πίνακα

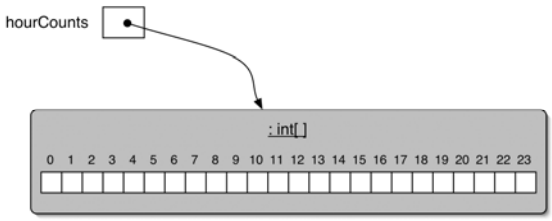
η μεταβλητή μπορεί να δεχθεί αντικείμενα τύπου πίνακας από ακέραιους

```
public class LogAnalyzer
{
    private int[] hourCounts ;
    private LogfileReader reader;

    public LogAnalyzer()
    {
        hourCounts = new int[24] ;
        reader = new LogfileReader();
    }
    ...
}
```

κατασκευάζει πίνακα 24 θέσεων [0..23] και τον ονομάζει hourCounts

Ο Πίνακας `hourCounts`

Σε όλους τους πίνακες οι υποδείκτες ξεκινούν από το 0 και φτάνουν ως το μέγεθός τους -1.



Χρήση Πινάκων

- Αφού κατασκευάσουμε και ονοματίσουμε ένα πίνακα, προσπελαύνουμε ένα στοιχείο του μέσω του υποδείκτη.

```
int[] hourCounts = new int[24];
hourCounts[0] = 1; //αναθέτει τιμή σε στοιχείο
double midday;
midday = (hourCounts[11] + hourCounts[12]) / 2.0;
```

- Το πλήθος των στοιχείων του πίνακα επιστρέφεται μέσω `hourCounts.length`

Εντολή Επανάληψης `for`

- Χρησιμοποιείται για συγκεκριμένο αριθμό επαναλήψεων
- Ταιριάζει σε επαναλήψεις επί στοιχείων πινάκων.

Σύνταξη

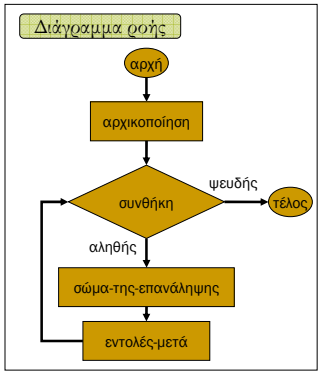
```
for (αρχικοποίηση; συνθήκη; εντολές-μετά ) {
    σώμα-της-επανάληψης
}
```


Ιωάννης Γαβιώτης
gaviotis@aegean.gr
http://www.syros.aegean.gr/users/gaviotis/tmp2

Εντολή **for**

17/3/2009

Διάγραμμα ροής



Σημασιολογία

1. Εκτέλεσε τις εντολές στην αρχικοποίηση
2. Εκτίμησε τη συνθήκη.
3. Αν είναι ψευδής, μετάφερε τον έλεγχο στην εντολή που έπεται της for.
4. Αν είναι αληθής, εκτέλεσε τον κώδικα στο σώμα της επανάληψης, τις εντολές-μετά και πήγαινε στο βήμα 1.

131

Παράδειγμα Επανάληψης

17/3/2009

Έκδοση με for

```
for ( int hour = 0 ;
      hour < hourCounts.length ;
      hour++ ) {
  System.out.println(hour + ": " + hourCounts[hour]);
}
```

Έκδοση με while

```
int hour = 0;
while (hour < hourCounts.length) {
  System.out.println(hour + ": " + hourCounts[hour]);
  hour++;
}
```

132

Όροι Ενότητας 4

17/3/2009

- Βιβλιοθήκη κλάσεων
- Επαναχρησιμοποίηση κώδικα
- Συλλογή αντικειμένων
- Υποδείκτης
- Απαριθμητής
- Εντολή επανάληψης
- Σταθερά
- Εκμαγείο
- Πίνακες
- Τύπος βάσης πίνακα

Τεκμήρια Ενότητας

- `import`
- `while`
- `null`
- `[ ]`
- `for`

133