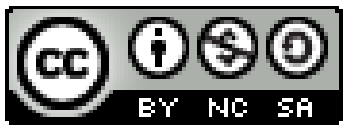
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Αντικειμενοστρεφής Προγραμματισμός

Ενότητα 4: Ομαδοποίηση Αντικειμένων

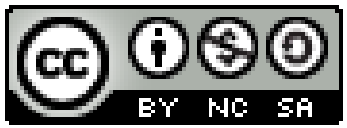
Γρηγόρης Τσουμάκας, Επικ. Καθηγητής
Τμήμα Πληροφορικής



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ομαδοποίηση Αντικειμένων

Τα παραδείγματα κώδικα που χρησιμοποιούνται σε κάποιες από τις ακόλουθες διαφάνειες μπορούν να βρεθούν στον παρακάτω σύνδεσμο:
http://users.auth.gr/greg/oop.zip

Πακέτα

- Πακέτο (package).
  - Μια ομάδα συναφών κλάσεων.
- Δήλωση πακέτου.
  - Με λέξη κλειδί *package* στην κορυφή του αρχείου.
    - Π.χ. package utilities;
  - Στο σύστημα αρχείων αντιστοιχεί σε φάκελο.
  - Ιεραρχία φακέλων αντιστοιχεί σε όνομα πακέτου που συνδέει τα ονόματα φακέλων με "."
    - Π.χ. package utilities.devices.printer;

Τι Εξυπηρετούν τα Πακέτα;

- Οργάνωση κλάσεων.
  - Ευκολία εύρεσης συγκεκριμένων κλάσεων εντός ενός μεγάλου πλήθους κλάσεων.
- Ορισμός χώρου ονομάτων (namespace).
  - Επίλυση θεμάτων συνωνυμίας μεταξύ κλάσεων.
- Έλεγχος πρόσβασης σε κλάσεις.
  - Πρόσβαση σε μια κλάση μόνο από κλάσεις του ίδιου πακέτου, εκτός και αν δηλωθεί ως public.

Ονόματα Πακέτων

- Χρησιμοποιούμε πεζούς χαρακτήρες.
- Όνομα του project για ενδοεταιρικό κώδικα.
  - Π.χ. package eclass;
- Χρήση του διαδικτυακού domain name της εταιρίας/του οργανισμού για ανοιχτό κώδικα.
  - Π.χ. package gr.auth.csd.eclass;
- Πακέτα γλώσσας Java (βιβλιοθήκες κλάσεων).
  - Ξεκινάνε από *java.* ή *javax.*
  - Οι κλάσεις *String* και *System* που έχουμε ήδη χρησιμοποιήσει βρίσκονται στο *java.lang*

Χρήση Κλάσεων από Πακέτα

- Με το όνομα του πακέτου (qualified name).
  - Π.χ. eclass.Student s = new eclass.Student();
- Με χρήση της λέξης κλειδί *import* στην κορυφή του αρχείου (μετά από *package*).
  - Π.χ. import eclass.Student;
  - Π.χ. import eclass.*;
- Κλάσεις με ίδιο όνομα;
  - Απαιτείται αναφορά με το όνομα του πακέτου.
- Το *java.lang* γίνεται import αυτόματα.

Δεν περιλαμβάνει κλάσεις υποπακέτων!

Έλεγχος Πρόσβασης (1/3)

```
package pkg1;
class DefaultClass {}
-----------------------------------------
package pkg1;
public class PublicClass {}
-----------------------------------------
package pkg1;
public class ClassUsingDefault {
    public static void main(String[] args) {
        DefaultClass x = new DefaultClass();
    }
}
```

Έλεγχος Πρόσβασης (2/3)

```
package pkg1;
class DefaultClass {}
-----------------------------------------
package pkg1;
public class PublicClass {}
-----------------------------------------
package pkg2;
import pkg1.*;
public class ClassUsingDefault {
    public static void main(String[] args) {
        DefaultClass x = new DefaultClass();
    }
}
```

Έλεγχος Πρόσβασης (3/3)

```
package pkg1;
class DefaultClass {}
-----------------------------------------
package pkg1;
public class PublicClass {}
-----------------------------------------
package pkg2;
import pkg1.*;
public class ClassUsingPublic {
    public static void main(String[] args) {
        PublicClass x = new PublicClass();
    }
}
```

Έλεγχος Πρόσβασης σε Μέλη

- Public.
  - Προσβάσιμο από οποιοδήποτε κώδικα.
- Private.
  - Προσβάσιμο μόνο από μέλη της ίδιας κλάσης.
- Default (χωρίς τροποποιητή πρόσβασης).
  - Προσβάσιμο μόνο από το πακέτο της κλάσης του.

Ανάγκη για Ομάδες Αντικειμένων

- Οι περισσότερες εφαρμογές απαιτούν την οργάνωση της πληροφορίας σε ομάδες.
  - Γραμματεία πανεπιστημιακού τμήματος, προσωπικά σημειωματάρια, βιβλιοθήκη.
- Ο αριθμός των στοιχείων που αποθηκεύονται συνήθως μεταβάλλεται με το χρόνο.
  - Προσθήκη στοιχείων.
  - Διαγραφή στοιχείων.

Συλλογές της Java

- Συλλογή (collection).
  - Δομή δεδομένων που ομαδοποιεί αντικείμενα της ίδιας κλάσης (π.χ. επαφές, βιβλία, φοιτητές, κτλ.).
  - Χρήση για αποθήκευση, ανάκτηση, διαχείριση και ανταλλαγή ενός συνόλου αντικειμένων.
- Πλαίσιο συλλογών (collection framework).
  - Έτοιμες συλλογές και αλγόριθμοι για αναζήτηση, ταξινόμηση, κτλ ανεξάρτητα από τη συλλογή.
  - Το πλαίσιο βρίσκεται στο πακέτο *java.util*

Η Κλάση ArrayList

- Κλάση ArrayList.
  - Διατεταγμένη συλλογή (λίστα) στοιχείων: τα στοιχεία έχουν αριθμημένη θέση στη συλλογή.
- Δήλωση.
  - ArrayList<Person> students;
  - ArrayList<TicketMachine> aMetroStation;
- Αρχικοποίηση.
  - students = new ArrayList<Person>();
  - students = new ArrayList<>();

Γενικές κλάσεις (generics)

Java 7 diamond notation

Ορισμένες Μέθοδοι της ArrayList

- int size();
  - Πλήθος αντικειμένων στη συλλογή.
- boolean add(E element);
  - Προσθέτει το *element* στο τέλος της λίστας.
- E get(int index);
  - Επιστρέφει το αντικείμενο στη θέση *index* [0..size).
- E remove(int index);
  - Αφαιρεί το αντικείμενο στη θέση *index* [0..size).

Παράδειγμα

- Οργάνωση μουσικών αρχείων.
  - Δυναμική προσθήκη/αφαίρεση μουσικών αρχείων από τη συλλογή.
  - Να μας λέει το πλήθος των αρχείων που υπάρχουν στη συλλογή.
  - Να εμφανίζει μια λίστα με τα ονόματα αρχείων που υπάρχουν στη συλλογή.

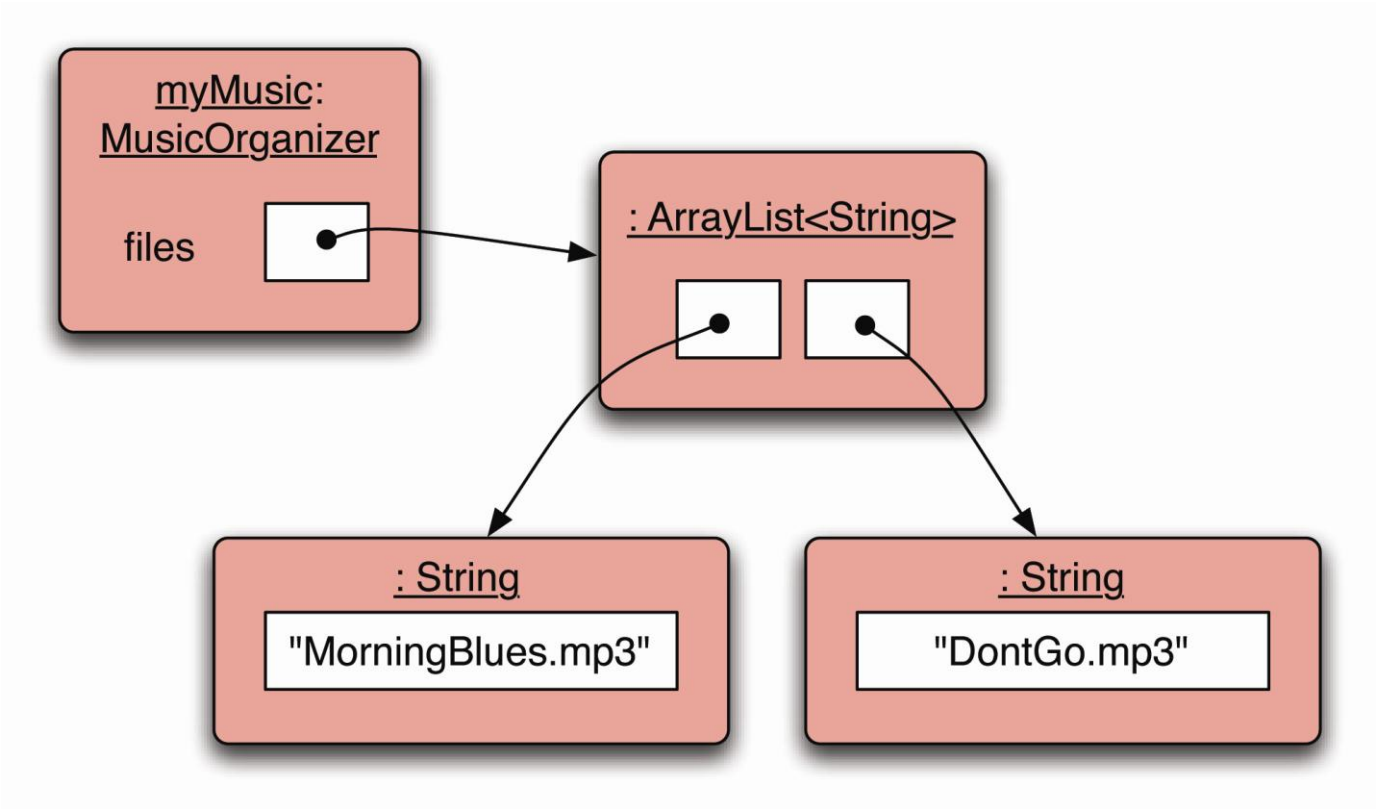
Η Κλάση MusicOrganizer

```
public class MusicOrganizer {
    private ArrayList<String> files;

    public MusicOrganizer() {
        files = new ArrayList<>();
    }

    …
}
```

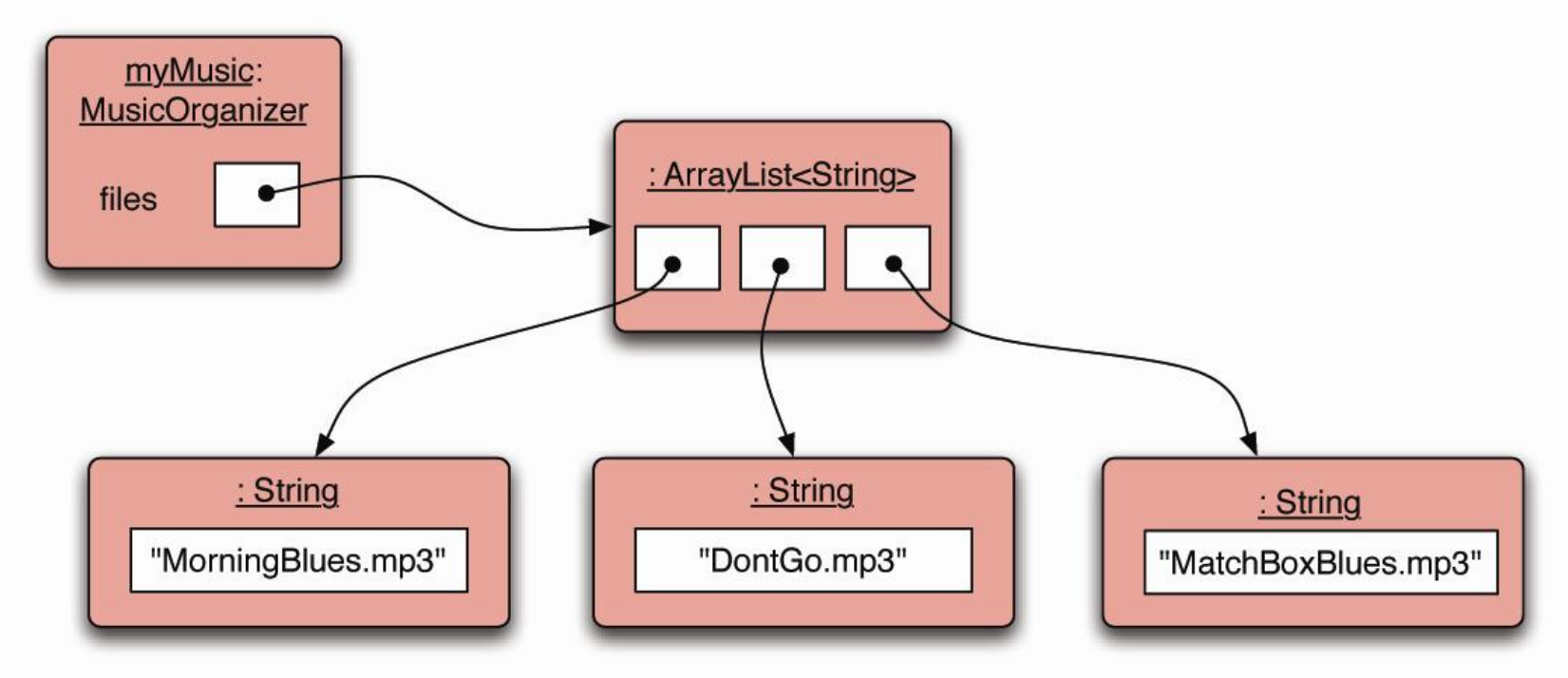
Προσθήκη και Πλήθος Στοιχείων

```
public void addFile(String filename) {
    files.add(filename);
}

public int getNumberOfFiles() {
    return files.size();
}
```

Διάγραμμα Αντικειμένων (1/2)



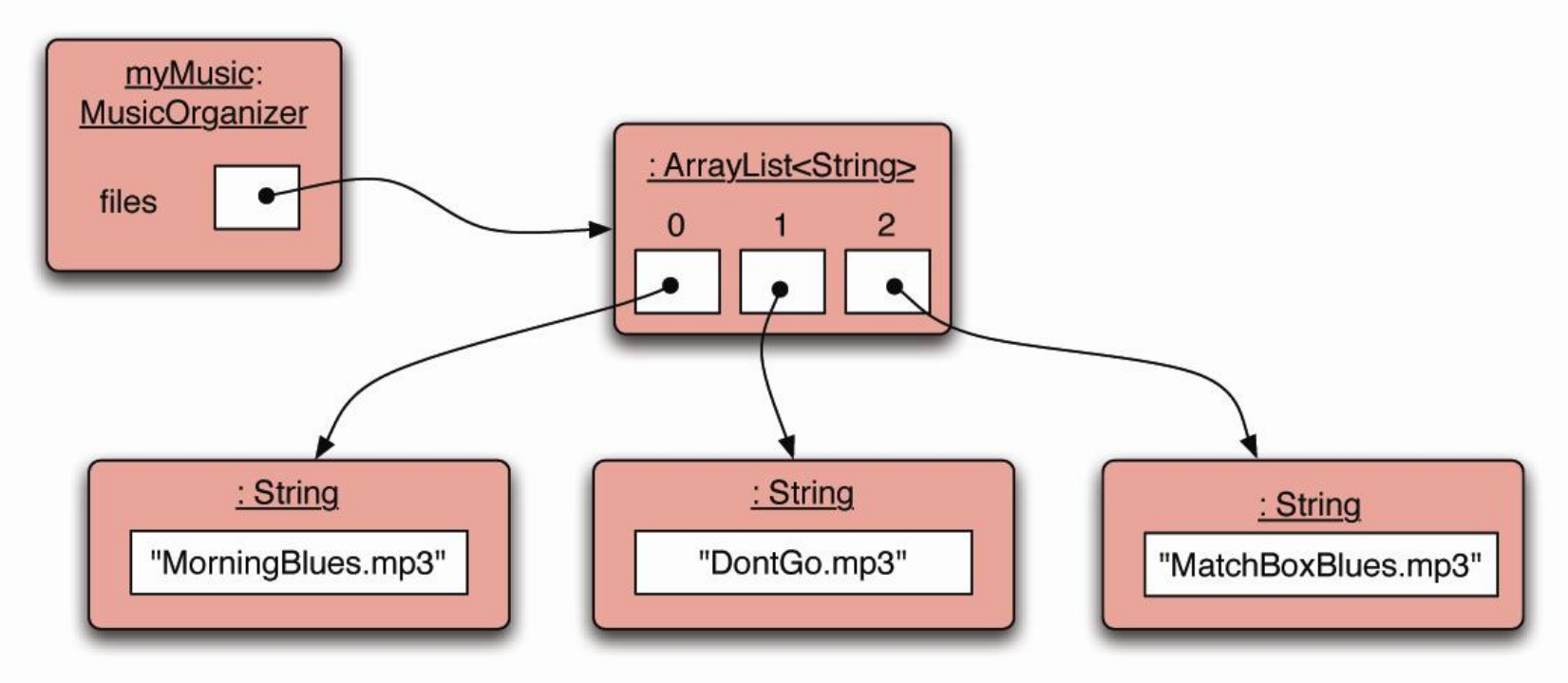
Διάγραμμα Αντικειμένων (2/2)



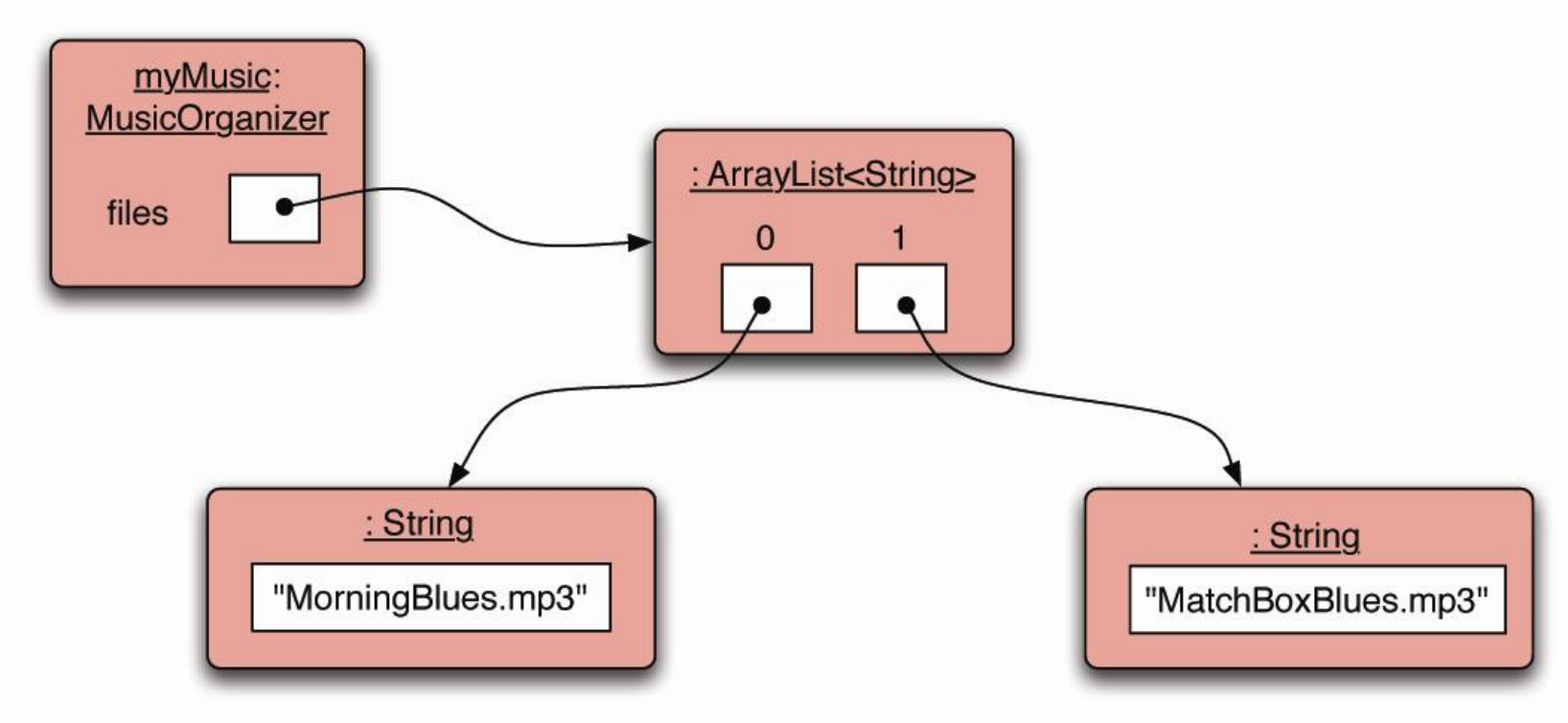
Εμφάνιση και Αφαίρεση Στοιχείου

```
public void listFile(int index) {
    if (index >= 0 && index < files.size()) {
        String filename = files.get(index);
        System.out.println(filename);
    }
}
public void removeFile(int index) {
    if (index >= 0 && index < files.size())
        files.remove(index);
}
```

Αρίθμηση Στοιχείων



Αποτέλεσμα Αφαίρεσης



Η Κλάση MusicOrganizerApp

```
public class MusicOrganizerApp {

  public static void main(String[] args) {
    MusicOrganizer m = new MusicOrganizer();

    m.addFile("EarlyMorningBlues.mp3");

    m.addFile("BabyPleaseDontGo1.mp3");

    m.addFile("MatchBoxBlues.mp3");

    m.removeFile(1);

    m.listFile(1);

  }

}
```

Ας Ακούσουμε Λίγη Μουσική

- Χρήση εξωτερικής βιβλιοθήκης javazoom.
  - www.javazoom.net
- Βοηθητική έτοιμη δική μας κλάση MusicPlayer.
  - Προκαθορισμένος κατασκευαστής.
  - Μέθοδος startPlaying(String filename).
  - Μέθοδος stop().
- Προσθήκη μεθόδων στο MusicOrganizer.
  - startPlayingFile(int index).
  - stopPlaying().

Η Κλάση MusicOrganizer (1/2)

```
public class MusicOrganizer {
    private ArrayList<String> files;
    private MusicPlayer player; // <--
    public MusicOrganizer() {
        files = new ArrayList<>();
        player = new MusicPlayer(); // <--
    }
    …
}
```

Η Κλάση MusicOrganizer (2/2)

```
public void startPlayingFile(int index) {
    if (index >= 0 && index < files.size()) {
        String filename = files.get(index);
        player.startPlaying(filename);
    }
}

public void stopPlaying() {
    player.stop();
}
```

Η Κλάση MusicOrganizerApp

```
public class MusicOrganizerApp {


   public static void main(String[] args) {
      MusicOrganizer m = new MusicOrganizer();
      m.addFile("EarlyMorningBlues.mp3");
      m.addFile("BabyPleaseDontGo1.mp3");
      m.addFile("MatchBoxBlues.mp3");
      m.startPlaying(1);
   }
}
```

Η Εντολή Επανάληψης for (1/2)

```
int i;
for (i=1; i<=10; i++) {
  System.out.println(i);
}
for (int j=1; j<=10; j++) {
  System.out.println(j);
}
for (;;) {
  System.out.println("αέναη επανάληψη");
}
```

Η Εντολή Επανάληψης for (2/2)

```
int i;
for (i=0; i<10; ) {
  System.out.println(i);
  i++;
}
int i, j;
for (i=0, j=10; i<j; i++, j--) {
  System.out.println("i,j: " + i + "," + j);
}
```

For.java

Η Εντολή Επανάληψης while

```
int counter = 1;
while (counter < 11) {
    System.out.println(counter);
    counter++;
}

while (true) {
    System.out.println("αέναη επανάληψη");
}
```

Η Εντολή Επανάληψης do-while

```
int counter = 11;
do {
    System.out.println(counter);
    counter++;
} while (counter < 20);
```

While.java

Τόσες Επαναλήψεις…

- Γνωρίζετε τον αριθμό των επαναλήψεων;
  - **`for`**
- Δεν γνωρίζετε τον αριθμό των επαναλήψεων;
  - Θα γίνει τουλάχιστον μία επανάληψη;
    - **`do-while`**
  - Μπορεί και να μην γίνει καμία επανάληψη;
    - **`while`**

Η Εντολή break

```
/*
 * Μικρότερος ακέραιος μεταξύ 100 και 999
 * που διαιρείται ακριβώς με το 7
 */
for (int i=100; i<1000; i++) {
    if (i % 7 == 0) {
        System.out.println(i);
        break;
    }
}
```

Η Εντολή continue

```
/*
 * Όλοι οι ακέραιοι μεταξύ 100 και 999
 * που διαιρούνται ακριβώς με το 7
 */
for (int i=100; i<1000; i++) {
    if (i % 7 != 0)
        continue;
    System.out.println(i);
}
```

Break.java

Διάσχιση με Επανάληψη (1/2)

```
/*
 * Χρησιμοποιώντας τις εντολές επανάληψης
 * της Java, σε συνδυασμό με τη μέθοδο get
 */
public void listAllFilesA1() {
    int i=0;
    while (i < files.size()) {
        String filename = files.get(i);
        System.out.println(filename);
        i++;
    }
}
```

Διάσχιση με Επανάληψη (2/2)

```
/*
 * Χρησιμοποιώντας τις εντολές επανάληψης
 * της Java, σε συνδυασμό με τη μέθοδο get
 */
public void listAllFilesA2() {
    for (int i=0; i<notes.size(); i++) {
        String filename = files.get(i);
        System.out.println(filename);
    }
}
```

Διάσχιση με For-Each

```
/*
 * Χρησιμοποιώντας τη δομή for-each, η οποία
 * δουλεύει με οποιαδήποτε συλλογή της Java.
 * Κάποιες δεν έχουν (αποδοτική) μέθοδο get.
 * for (Τύπος μεταβλητή : συλλογή) { σώμα }
 */
public void listAllFilesB() {
    for (String filename : files) {
        System.out.println(filename);
    }
}
```

Η μεταβλητή *filename* είναι τοπικό αντίγραφο

Δεν γνωρίζουμε τη θέση στη συλλογή

Διάσχιση με Iterator (1/2)

```
/*
 * Ο επαναλήπτης (iterator) δουλεύει με
 * οποιαδήποτε συλλογή της Java.
 * Κάποιες δεν έχουν (αποδοτική) μέθοδο get.
 */
public void listAllFilesC1() {
    Iterator<String> it = files.iterator();
    while (it.hasNext()) {
        String filename = it.next();
        System.out.println(filename);
    }
}
```

Διάσχιση με Iterator (2/2)

```
/*
 * Ο επαναλήπτης (iterator) δουλεύει με
 * οποιαδήποτε συλλογή της Java.
 * Κάποιες δεν έχουν (αποδοτική) μέθοδο get.
 */
public void listAllFilesC2() {
  for (Iterator<String> it = files.iterator();
       it.hasNext(); ) {
         String filename = it.next();
         System.out.println(filename);
    }
}
```

Η Κλάση MusicOrganizerApp

```
public class MusicOrganizerApp {
   public static void main(String[] args) {
      MusicOrganizer m = new MusicOrganizer();
      m.addFile("EarlyMorningBlues.mp3");
      m.addFile("BabyPleaseDontGo1.mp3");
      m.addFile("MatchBoxBlues.mp3");
      m.listAllFilesC2();
   }
}
```

Διάσχιση και Αφαίρεση (1/3)

- Εύρεση και διαγραφή ενός κομματιού.
  - Δεν μπορεί να χρησιμοποιηθεί for-each.
  - Χρησιμοποιείται η μέθοδος remove() του Iterator.

```
public void delete(String fileToDelete) {
  for (Iterator<String> it = files.iterator();
       it.hasNext(); ) {
    String filename = it.next();
    if (filename.equals(fileToDelete))
      it.remove();
  }
}
```

Διάσχιση και Αφαίρεση (2/3)

- Μόνο για συλλογές που έχουν μέθοδο *get()*.

```
public void delete2(String fileToDelete) {
    for (int i=0; i<files.size(); i++) {
        String filename = files.get(i);
        if (filename.equals(fileToDelete)) {
            files.remove(i);
        }
    }
}
```

Τι πρόβλημα υπάρχει σε αυτόν τον κώδικα και πως θα το διορθώσουμε;

Διάσχιση και Αφαίρεση (3/3)

- Μόνο για συλλογές που έχουν μέθοδο *get()*.

```
public void delete2(String fileToDelete) {
    for (int i=0; i<files.size(); i++) {
        String filename = files.get(i);
        if (filename.equals(fileToDelete)) {
            files.remove(i);
            i--;
        }
    }
}
```

Τόσες Διασχίσεις…

- Διάσχιση και αφαίρεση στοιχείων.
  - Iterator (for/while).
- Απλή διάσχιση.
  - Όλων των στοιχείων.
    - For-each.
  - Άγνωστου αριθμού στοιχείων.
    - Iterator (while).

Πίνακες

- Πίνακες.
  - Σταθερού μεγέθους ομάδες στοιχείων ίδιου τύπου.
  - Τα στοιχεία μπορεί να είναι τόσο τύπου αναφοράς (κλάσεις) όσο και θεμελιώδους τύπου.
- Ο ίδιος ο πίνακας είναι τύπος αναφοράς.
  - Μια μεταβλητή τύπου πίνακα περιέχει μια αναφορά σε ένα αντικείμενο πίνακα στη μνήμη.

Δήλωση Πινάκων

```
int c[];
int c[], x;
int c[], x, a[], b[];

int[] c;
int[] a, b, c;
```

Προτιμητέο στυλ δήλωσης

Δημιουργία Πινάκων

- Δημιουργία πίνακα.

```
int[] a1;

a1 = new int[100];

double[] a2 = new double[5]; // με δήλωση

String[] a3 = new String[3]; // με δήλωση
```

- Αρχικές τιμές.
  - 0 για *int*, *float*, *double*, *byte* και *short*, *false* για *boolean*, '\u0000' για *char* και *null* για αναφοράς.
- Δήλωση, δημιουργία και απόδοση τιμών.

```
int[] a = {4, 9, 5};
```

Μέγεθος Πίνακα

- Πεδίο *length.*
  - Δημόσιας πρόσβασης πεδίο, όπου αποθηκεύεται το πλήθος των στοιχείων ενός πίνακα.
  - Προσοχή: δεν έχουμε παρενθέσεις όπως στη δημόσια μέθοδο πρόσβασης *size()* των συλλογών.
- Παράδειγμα.

```
int[] a = {3, 5, 7, 9};
System.out.println("Μέγεθος: " + a.length);
```

Πρόσβαση στα Στοιχεία

- Μέσω της θέσης τους (int, byte ή short).
  - Το πρώτο στοιχείο έχει θέση 0, ενώ το τελευταίο έχει θέση όσο το μέγεθος του πίνακα μείον 1.
  - Γίνεται έλεγχος από την JVM αν η θέση είναι εντός των επιτρεπτών ορίων του πίνακα.
- Παράδειγμα.

```
int[] array = {1, 2, 3};
array[0] = 7;
System.out.println(array[1]);
array[4] = 16; // σφάλμα εκτέλεσης
```

Πως Διατρέχουμε Έναν Πίνακα;

- Χρησιμοποιώντας την εντολή for.

```
String[] array = {"a", "b", "c"};
for (int i=0; i<array.length; i++)
  System.out.println(array[i]);
```

- Με τις εντολές επανάληψης *while, do-while.*
- Με την απλούστερη δομή *for-each.*

```
for (String x : array)
  System.out.println(x);
```

Παραδείγματα (1/3)

```
int[] array = {1, 2, 3};

System.out.println("ο πίνακας περιέχει:");

for (int x : array) {
      System.out.println(x);
}

for (int x : array) {
      x = 5;
}

System.out.println("πλέον περιέχει:");

for (int x : array) {
      System.out.println(x);
}
```

ArraysForEach.java

Παραδείγματα (2/3)

```
public class Person {

    private String name;

    public Person(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public void changeName(String newName) {
        name = newName;
    }

}
```

Person.java

Παραδείγματα (3/3)

```
Person[] a = new Person[2];
a[0] = new Person("Γρηγόρης");
a[1] = new Person("Δήμητρα");
System.out.println("ο πίνακας περιέχει:");
for (String x : a)
     System.out.println(x.getName());
for (String x : a) x.changeName("Γιώργος");
System.out.println("πλέον περιέχει:");
for (String x : a)
     System.out.println(x.getName());
```

ArraysForEach2.java

Φωλιασμένοι Πίνακες (1/2)

- Δήλωση φωλιασμένων πινάκων.

```
int[][] array2d; // διδιάστατος;
int[][][] array3d; // τριδιάστατος;
```

- Έμμεση υποστήριξη πολυδιάστατων πινάκων.
  - Πίνακες των οποίων τα στοιχεία είναι πίνακες, των οποίων τα στοιχεία είναι πίνακες…
  - Αυτό σημαίνει ότι ένας πίνακας-στοιχείο ενός πολυδιάστατου πίνακα μπορεί να έχει διαφορετικό αριθμό στοιχείων από έναν άλλο!

Φωλιασμένοι Πίνακες (2/2)

- Δημιουργία πίνακα με ανάθεση τιμών.

```
int[][] a = {{1, 2, 3}, {4, 5, 6}};
int[][] b = {{2, 3}, {4, 5, 6}};
```

- Δημιουργία πίνακα.

```
int[][] a = new int[2][3];
int[][] b = new int[2][];
b[0] = new int[2];
b[1] = new int[3];
```

- Πώς θα διατρέχατε τον φωλιασμένο πίνακα b.
  - α) με χρήση απλής for, και β) με χρήση for-each;

OutputArray2D.java

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Τέλος Ενότητας

Επεξεργασία: Εμμανουήλ Ρήγας

Θεσσαλονίκη, Εαρινό Εξάμηνο 2013-2014

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης