Προχωρημένος Προγραμματισμός

Κλάσεις και Αντικείμενα

ΕΛΕΥΘΕΡΙΟΣ ΚΟΣΜΑΣ

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

Περίληψη

Σήμερα ...

- θα μελετήσουμε τις κλάσεις και τα αντικείμενά τους
- θα δούμε, πρακτικά σε κώδικα πώς τα αναπαριστούμε, πώς δημιουργούμε αντικείμενα και πώς καλούμε τις μεθόδους τους

- θα συζητήσουμε υπάρχουσες κλάσεις (που έχουμε ήδη χρησιμοποιήσει)

- θα παρουσιάσουμε κάποιες χρήσιμες δομές για την αποθήκευση των δεδομένων του προγράμματός μας

Αντικειμενοστραφής προγραμματισμός

βάζει μαζί τα δεδομένα και τις διαδικασίες (μεθόδους) σχετικές με τα δεδομένα

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

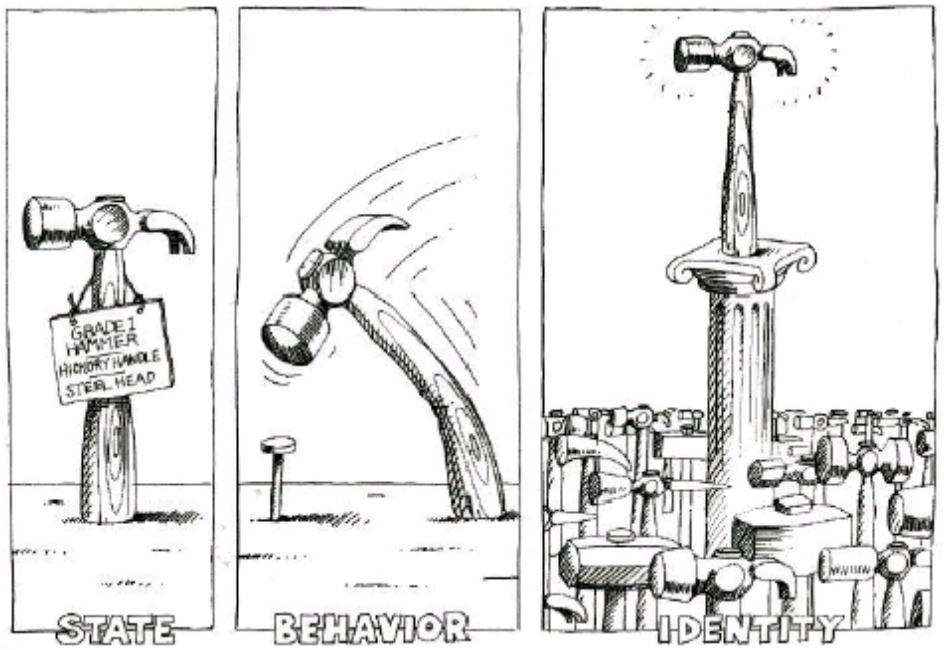
☞ αυτό επιτυγχάνεται με αντικείμενα και κλάσεις

Αντικειμενοστραφής προγραμματισμός

Αντικείμενα

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

Αντικειμενοστραφής προγραμματισμός
Κλάση

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

  Παραδείγματα:

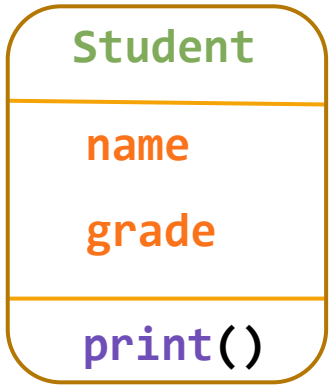
  1. η κλάση `Φοιτητής` έχει τα γενικά χαρακτηριστικά (όνομα, βαθμοί) και τη συμπεριφορά `print`
     - ο φοιτητής `X` είναι ένα αντικείμενο της κλάσης `Φοιτητής` με κατάσταση τα χαρακτηριστικά (Γιώργος, 10,8)
  2. η κλάση `Car` έχει τα χαρακτηριστικά (brand, color) και τη συμπεριφορά (drive, stop)
  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

Αντικειμενοστραφής προγραμματισμός

Κλάσεις και Αντικείμενα - Παράδειγμα

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)

Κλάση & Αντικείμενο
Σύνοψη

- Κλάση: μια αφηρημένη περιγραφή αντικειμένων με κοινά χαρακτηριστικά και κοινή συμπεριφορά
  - ένα καλούπι που παράγει αντικείμενα

- αντικείμενο: ένα στιγμιότυπο μίας κλάσης

- η κλάση ορίζει τον τύπο του αντικειμένου
  - τα χαρακτηριστικά του αντικειμένου
  - τις ενέργειες που μπορεί να επιτελέσει

Κλάση & Αντικείμενο

Πρακτικά στον κώδικα

μία κλάση Κ ορίζεται από:

- κάποιες μεταβλητές τις οποίες ονομάζουμε πεδία
- κάποιες συναρτήσεις που τις ονομάζουμε μεθόδους
  - οι μέθοδοι «βλέπουν» τα πεδία της κλάσης

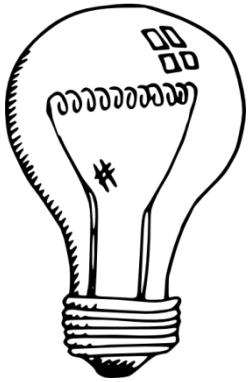
ένα αντικείμενο ορίζεται ως μια μεταβλητή τύπου Κ

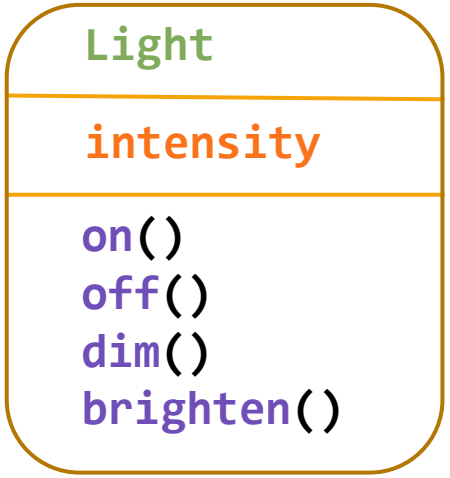
- το αντικείμενο έχει συγκεκριμένες τιμές στα πεδία
- στο πρόγραμμα έχουμε (συνήθως) πρόσβαση μόνο στις μεθόδους
  - μέσω των μεθόδων έχουμε πρόσβαση στα πεδία
- αν υπάρχουν κάποια πεδία στα οποία έχουμε πρόσβαση αυτά τα λέμε ιδιότητες (properties)

Κλάση

Γενική μορφή - Παράδειγμα

- Θέλουμε να κάνουμε ένα πρόγραμμα που να διαχειρίζεται τα φώτα σε διάφορα δωμάτια και θα υλοποιεί και ένα dimmer





Αντικείμενο
Δημιουργία

```
<Όνομα Κλάσης> myObject = new <Όνομα Κλάσης>([Ορίσματα]);
```

- η λέξη κλειδί **new** δημιουργεί ένα καινούριο αντικείμενο
  - δεσμεύει τον απαραίτητο χώρο στη μνήμη
  - παράδειγμα που έχουμε ήδη δει:

```
Scanner input = new Scanner(System.in);
```

- η λίστα των ορισμάτων μπορεί να είναι και κενή

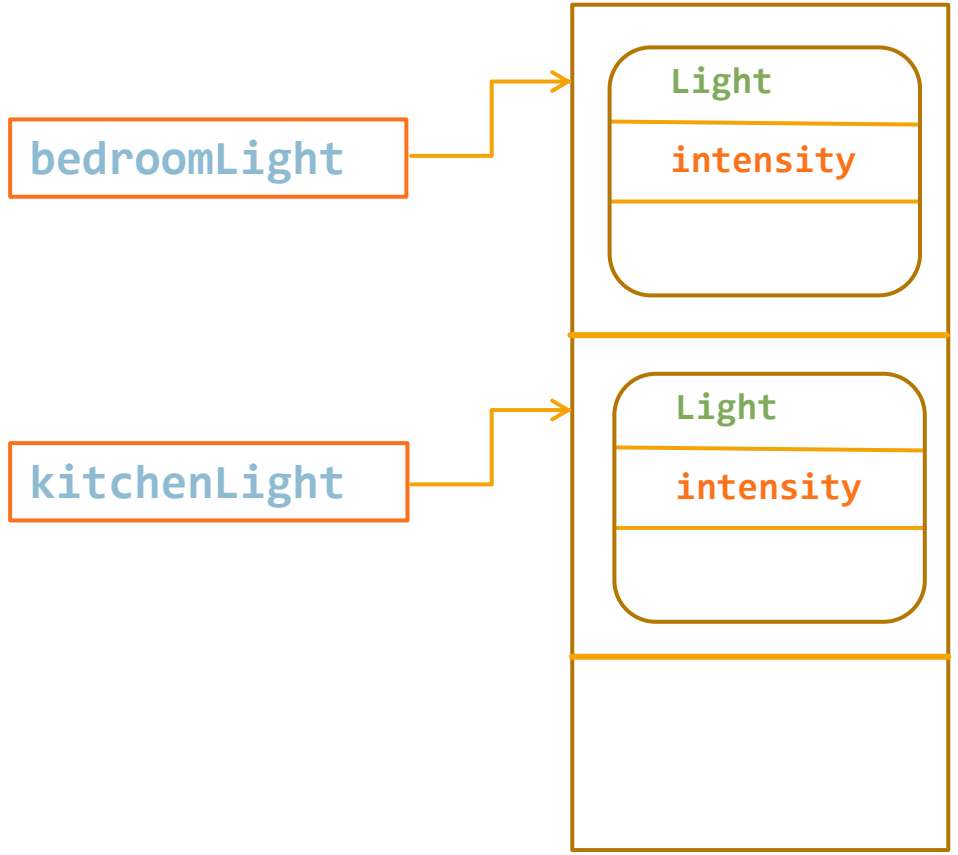
Αντικείμενο
Δημιουργία ΙΙ

```
Light bedroomLight = new Light();
Light kitchenLight = new Light();
```

η εντολή **new**

- δημιουργεί ένα καινούριο αντικείμενο της κλάσης και του δίνει ένα όνομα
- δεσμεύει χώρο μνήμης για το αντικείμενο
- επιστρέφει τη διεύθυνση του χώρου που δεσμεύτηκε

✍ η μεταβλητή που ορίζουμε "δείχνει" σε αυτό τον χώρο μνήμης



Αντικείμενο
Κλήση μεθόδων

- η πρόσβαση που έχουμε στα αντικείμενα είναι (κατά κύριο λόγο) μέσα από τις μεθόδους τους
- κλήση μιας μεθόδου:

<όνομα αντικειμένου>.<όνομα μεθόδου>

  - παράδειγμα:

```
Light bedroomLight = new Light();
bedroomLight.on();
bedroomLight.brighten();
bedroomLight.dim();
bedroomLight.off();
```

Αντικείμενο
Κλήση μεθόδων ΙΙ

- γενικός κανόνας: για να καλέσουμε μια μέθοδο μιας κλάσης θα πρέπει να δημιουργήσουμε ένα αντικείμενο της κλάσης

- εξαίρεση: οι στατικές μέθοδοι
  - μπορούν να κληθούν χρησιμοποιώντας το όνομα της κλάσης
  - παράδειγμα:
    1. η μέθοδος **`main`** καλείται χωρίς να έχουμε δημιουργήσει αντικείμενο
       - για το λόγο αυτό την έχουμε ορίσει ως **`static`**

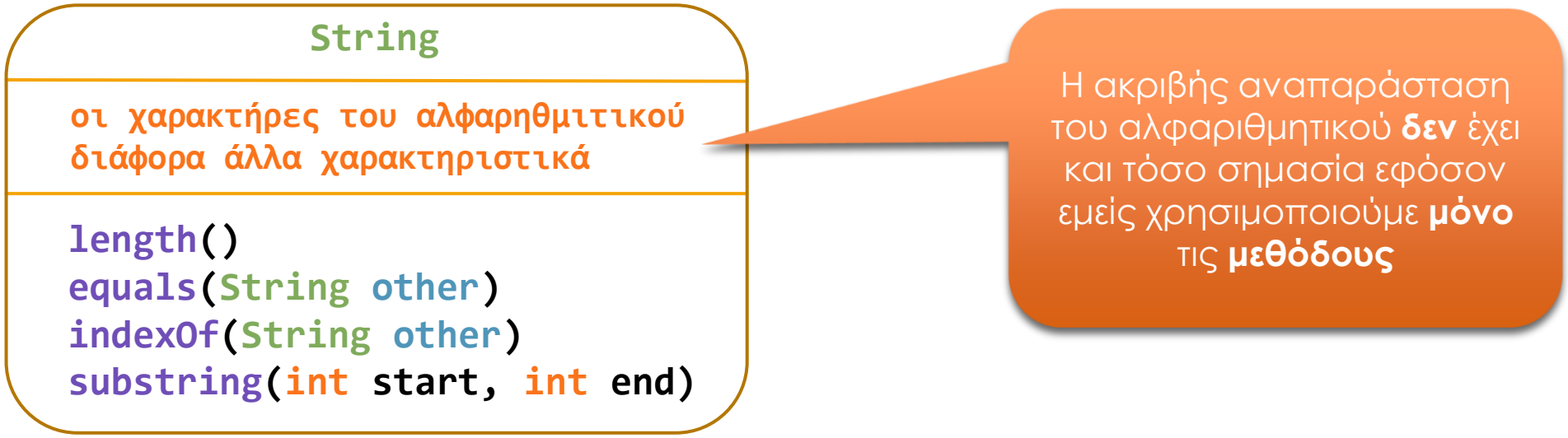
❓ `System.out.println("hello");`

- καλεί τη μέθοδο `println` του στατικού αντικειμένου `System.out`
  - `public static final PrintStream out;` ← στατικό αντικείμενο (`out`) κλάσης (`System`)

Υπάρχουσες Κλάσεις

Υπάρχουσες κλάσεις
String

- έχουμε ήδη χρησιμοποιήσει κλάσεις και αντικείμενα όταν χρησιμοποιούμε `String`



Υπάρχουσες κλάσεις
String - Αντικείμενα

- ένα String αντικείμενο είναι μια μεταβλητή τύπου String
  - τρεις διαφορετικοί τρόποι να δώσουμε τιμή σε ένα String αντικείμενο

```
1.  // τρόποι ανάθεσης τιμής σε String αντικείμενο
2.
3.  import java.util.Scanner;
4.
5.  class InitializeString
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.       String x = input.next();
11.
12.       String y = new String("java");
13.       String z = "java";
14.    }
15. }
```

Υπάρχουσες κλάσεις
String - Μέθοδοι

υπάρχουν πολλές χρήσιμες μέθοδοι της κλάσης `String`

- `length()`: επιστρέφει το μήκος του `String`
- `equals(String x)`: ελέγχει για ισότητα του `String` που καλεί τη μέθοδο με το τη συμβολοσειρά `x`
- `trim()`: αφαιρεί κενά στην αρχή και στο τέλος του `String`
- `split(String delim)`: σπάει το `String` σε πίνακα από `String`s βάσει του `String delim`
- `indexOf(String s)`: επιστρέφει τη θέση της πρώτης εμφάνισης του `s` μέσα στο `String` που καλεί τη μέθοδο
- `substring(int start, int end)`: επιστρέφει την υπακολουθία του `String` που καλεί τη μέθοδο, μεταξύ των θέσεων `start` και `end`
- ...

Υπάρχουσες κλάσεις

String - Μέθοδοι - Παράδειγμα

```
1.  // χρήση μεθόδων String αντικειμένων
2.
3.  class StringExample
4.  {
5.     public static void main(String args[])
6.     {
7.        String x = new String("Introduction to Java Programming");
8.        String y = "Java";
9.
10.       int offset = x.indexOf(y);          // offset = 16
11.       int end = x.length();               // end = 32
12.       x = x.substring(offset,end);
13.       System.out.println(x);              // Java Programming
14.    }
15. }
```

- ένα αντικείμενο `String` είναι αμετάβλητο (immutable)
- η τελευταία ανάθεση (γραμή 12) δημιουργεί ένα καινούργιο αντικείμενο και το αναθέτει στη μεταβλητή **x**

Αμετάβλητα (immutable) αντικείμενα

- η εσωτερική κατάστασή τους δε μπορεί να μεταβληθεί
  - εσωτερική κατάσταση: τα πεδία τους

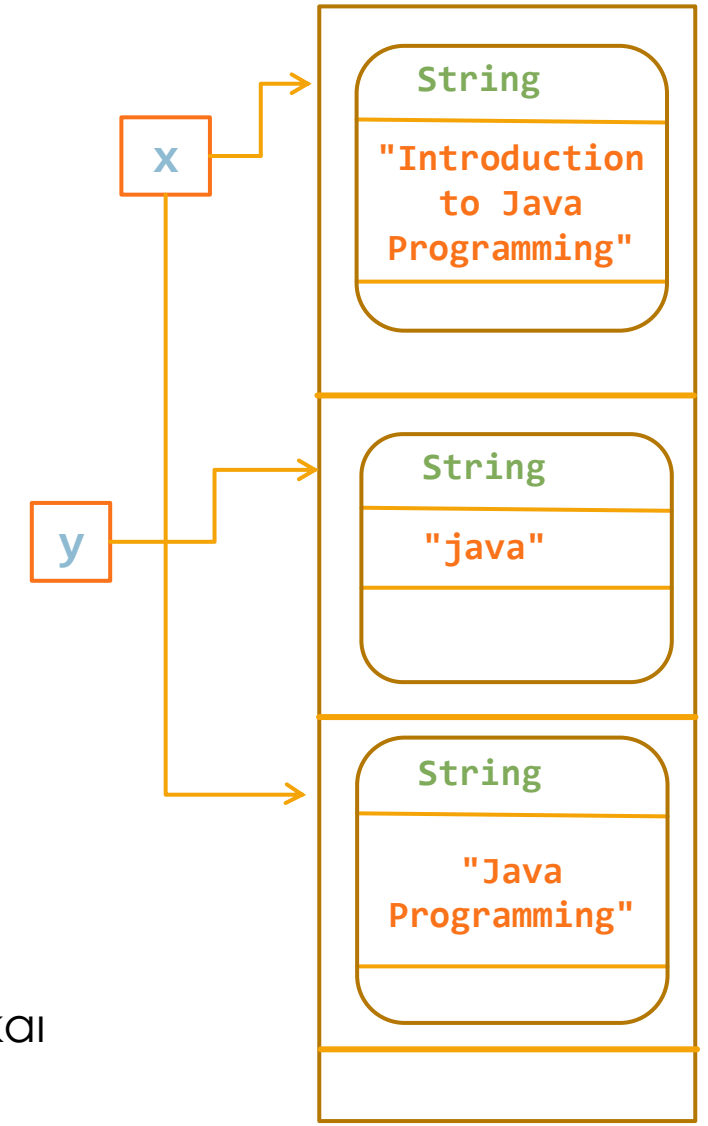
παράδειγμα:

- τα `String`s είναι αμετάβλητα αντικείμενα
  - αυτό σημαίνει ότι δε μπορούμε να αλλάξουμε τα περιεχόμενα ενός αντικειμένου `String`
    - π.χ., δε μπορούμε να αλλάξουμε ένα χαρακτήρα ενός String
  - οποιαδήποτε αλλαγή κάνουμε, χρησιμοποιώντας κάποια μέθοδο της κλάσης `String`, έχει ως αποτέλεσμα:
    1. τη δημιουργία ενός καινούργιου `String`,
    2. την επιστροφή του από τη μέθοδο και
    3. την εκχώρησή του στη μεταβλητή μας

Αμετάβλητα (immutable) αντικείμενα

Παράδειγμα

```
1.  // χρήση μεθόδων String αντικειμένων
2.
3.  class StringExample
4.  {
5.     public static void main(String args[])
6.     {
7.        String x = new String("Introduction to Java Programming");
8.        String y = "Java";
9.
10.       int offset = x.indexOf(y);          // offset = 16
11.       int end = x.length();               // end = 32
12.       x = x.substring(offset,end);
13.       System.out.println(x);              // Java Programming
14.    }
15. }
```



- τα `String` αντικείμενα είναι αμετάβλητα
- η τελευταία ανάθεση δημιουργεί ένα καινούργιο αντικείμενο και το αναθέτει στη μεταβλητή **x**

Υπάρχουσες κλάσεις

String - Ισότητα

```
1.  // έλεγχος ισότητας String αντικειμένων
2.
3.  class StringEquality
4.  {
5.     public static void main(String args[])
6.     {
7.        String x = new String("java");
8.        String y = new String("java");
9.        String z = y;
10.
11.       System.out.println("1. " + (x == y));        // 1. false
12.       System.out.println("2. " + (y == z));        // 2. true
13.       System.out.println("3. " + (z == x));        // 3. false
14.       System.out.println("4. " + x.equals(y));     // 4. true
15.       System.out.println("5. " + y.equals(z));     // 5. true
16.       System.out.println("6. " + z.equals(x));     // 6. true
17.    }
18. }
```

Τι θα εκτυπωθεί?

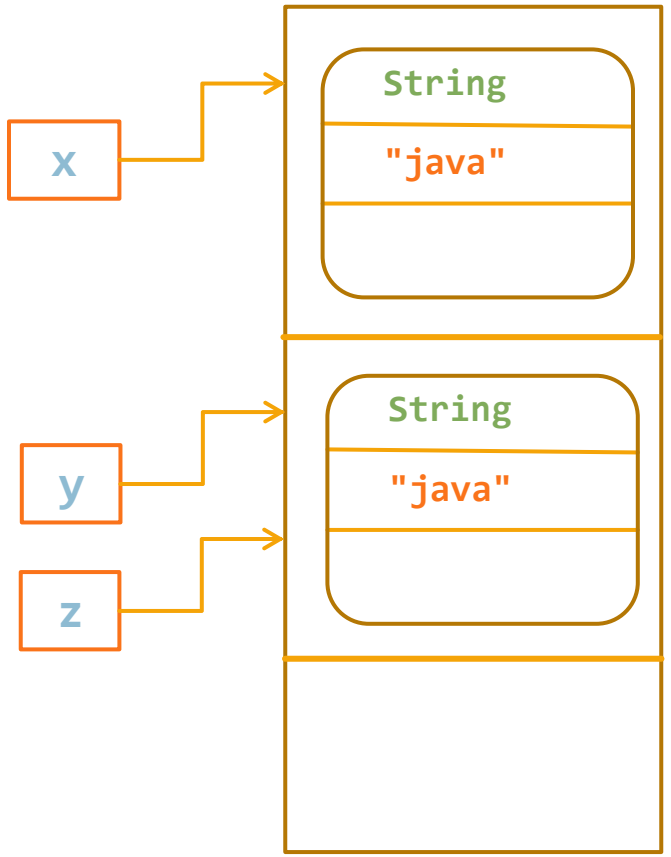
✍ (σημείωση: μια λογική συνθήκη αποτιμάται σε `true` ή `false` ανάλογα με το εάν είναι αληθής/ψευδής)

✍ για τη σύγκριση `String` αντικειμένων πάντα χρησιμοποιούμε τη μέθοδο `equals`

Υπάρχουσες κλάσεις

String - Ισότητα - Εξήγηση

```
1.  // έλεγχος ισότητας String αντικειμένων
2.
3.  class StringEquality
4.  {
5.     public static void main(String args[])
6.     {
7.        String x = new String("java");
8.        String y = new String("java");
9.        String z = y;
10.    }
11. }
```

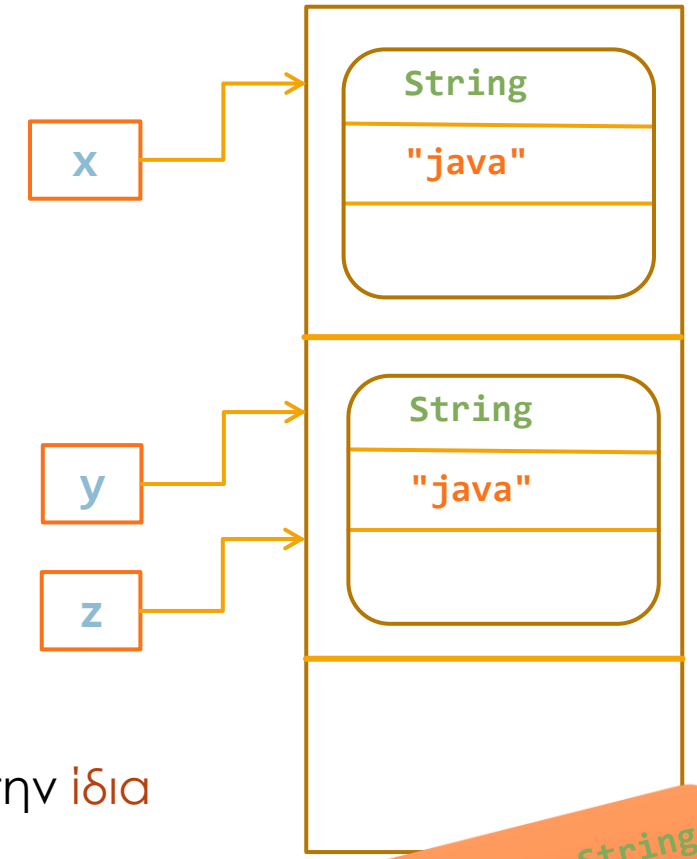


- όταν δημιουργούμε ένα `String` αντικείμενο δεσμεύουμε χώρο στη μνήμη για το αντικείμενο
- η μεταβλητή που ορίζουμε «δείχνει» σε αυτό το χώρο μνήμης
- η εκχώρηση μεταξύ αντικειμένων τα κάνει να δείχνουν στην ίδια θέση μνήμης

Υπάρχουσες κλάσεις

String - Ισότητα - Εξήγηση II

```
1.  // έλεγχος ισότητας String αντικειμένων
2.
3.  class StringEquality
4.  {
5.     public static void main(String args[])
6.     {
7.        String x = new String("java");
8.        String y = new String("java");
9.        String z = y;
10.
11.       System.out.println("2. " + (y == z));
12.       ...
13.    }
14. }
```



- ✍ ο τελεστής == μεταξύ δύο αντικειμένων εξετάζει αν δείχνουν στην ίδια θέση μνήμης
  - για το λόγο αυτό η συνθήκη `(y == z)` αποτιμάται ως `true`
- ☞ όλα αυτά θα είναι πιο ξεκάθαρα όταν θα μιλήσουμε για αναφορές

για τη σύγκριση String αντικειμένων χρησιμοποιούμε πάντα τη μέθοδο equals

Υπάρχουσες κλάσεις
String - Σταθερές

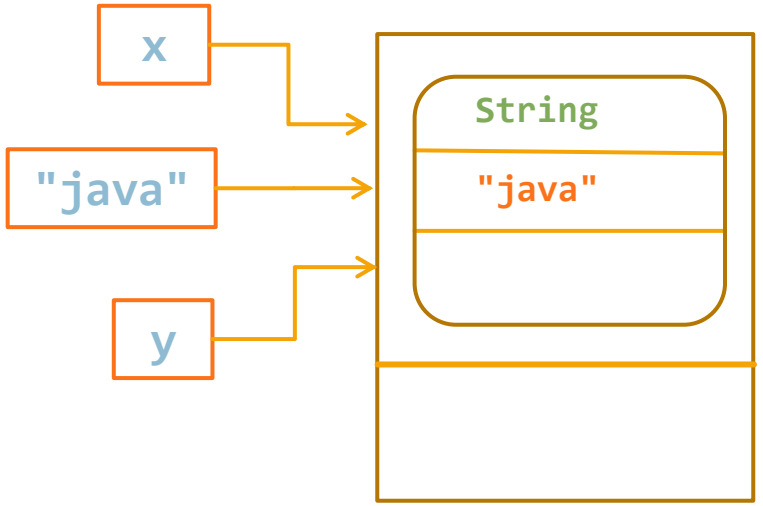
- οι String τιμές είναι κι αυτές αντικείμενα και μπορούμε να καλέσουμε τις μεθόδους τους

```
1.  // χρήση μεθόδων String τιμών
2.
3.  class StringConstants
4.  {
5.     public static void main(String args[])
6.     {
7.        int offset = "Java Programming".indexOf("Pro");        // offset = 5
8.        int end = "Java Programming".length();                 // end = 16
9.        String z = "Java Programming".substring(offset,end);
10.       System.out.println(z);                                 // Programming
11.    }
12. }
```

Υπάρχουσες κλάσεις

String - Σταθερές II

```
1.  // έλεγχος ισότητας String αντικειμένων
2.
3.  class StringEquality
4.  {
5.     public static void main(String args[])
6.     {
7.        String x = "java";
8.        String y = "java";
9.        System.out.println((x == y));       // true
10.    }
11. }
```



- ο ορισμός της σταθεράς `"java"` δημιουργεί ένα αντικείμενο με αυτό το όνομα
  - intern `String`
- οι εκχωρήσεις `x = "java"` και `y = "java"` κάνει τις μεταβλητές `x` και `y` να δείχνουν σε αυτό το αντικείμενο
  - για το λόγο αυτό, η συνθήκη `(x == y)` αποτιμάται ως `true`
- ☞ θα το συζητήσουμε ξανά όταν θα μιλήσουμε για αναφορές

Υπάρχουσες κλάσεις
Scanner

- δημιουργία αντικειμένου Scanner

```
Scanner input = new Scanner(System.in);
```

- μέθοδοι της `Scanner`
  - `next()`: επιστρέφει το επόμενο `String` από την είσοδο
    - όλοι οι χαρακτήρες από το σημείο που σταμάτησε την προηγούμενη φορά μέχρι να βρει λευκό καρακτήρα
    - ✍ λευκός χαρακτήρας: κενο, tab, αλλαγή γραμμής
  - `nextInt()`: διαβάζει το επόμενο `String`, το μετατρέπει σε αριθμό `int` και τον επιστρέφει
  - `nextDouble()`: διαβάζει το επόμενο `String`, το μετατρέπει σε αριθμό `double` και τον επιστρέφει
  - `nextLine()`: διαβάζει ότι υπάρχει μέχρι να βρει αλλαγή γραμμής και το επιστρέφει ως `String`

Υπάρχουσες κλάσεις
Wrapper classes

- για κάθε βασικό τύπο η Java έχει και μία wrapper class
  - `Integer` class
  - `Double` class
  - `Boolean` class
- οι κλάσεις αυτές έχουν κάποιες μεθόδους και πεδία που μπορεί να μας είναι χρήσιμα
  - κατά κύριο λόγο μετατροπή από και προς `String`
  - τη μέγιστη και την ελάχιστη τιμή κάθε τύπου
- κάποιες από τις μεθόδους είναι στατικές
  - μπορούμε να τις καλέσουμε χωρίς να έχουμε αντικείμενο
- έχουμε επίσης πρόσβαση σε κάποια στατικά πεδία

Υπάρχουσες κλάσεις

Wrapper classes - Παράδειγμα

```
1.  // παράδειγμα χρήσης wrapper classes
2.
3.  class WrapperExample
4.  {
5.     public static void main(String args[])
6.     {
7.        int i = Integer.valueOf("2");
8.        double d = Double.parseDouble("2.5");
9.        System.out.println("i*d = " + i*d);          // i*d = 5
10.
11.       Integer x = 5;
12.       Double y = 2.5;
13.       String s = x.toString() + y.toString();
14.       System.out.println("s = " + s);             // s = 52.5
15.
16.       System.out.println(Integer.MAX_VALUE);      // στατικό πεδίο
17.    }
18. }
```

Δομές

Δομές
Πίνακες

- πολλές φορές έχουμε πολλές μεταβλητές του ίδιου τύπου που συσχετίζονται και θέλουμε να τις βάλουμε μαζί
  - τα ονόματα των φοιτητών σε μία τάξη
  - οι βαθμοί ενός φοιτητή για όλα τα εργαστήρια
- ☞ για το σκοπό αυτό μπορούμε να χρησιμοποιήσουμε πίνακες

- συντακτικό:

```
<τύπος>[] myArray;
```

  - ο τύπος είναι οποιαδήποτε κλάση
  - ο τύπος των δεδομένων που αποθηκεύει ο πίνακας μας
- παραδείγματα:

```
int[] integerArray;      // πίνακας από ακεραίους
String[] stringArray;    // πίνακας από String
```

Δομές
Πίνακες - Ορισμός

```
1.  ...
2.  int[] myArray1 = {10,20};
3.  int myArray2[] = new int[2];
4.  int[] myArray3;
5.    ...
```

ορίζει έναν πίνακα ακεραίων δύο θέσεων με αρχικές τιμές 10, 20

Ορίζει ένα πίνακα ακεραίων δύο θέσεων χωρίς αρχικές τιμές (αρχικοποιούνται αυτόματα στο μηδέν)

ορίζει μια μεταβλητή που είναι πίνακας από ακεραίους, χωρίς να δεσμεύει χώρο για τον πίνακα

- οι πίνακες ορίζονται με ένα μέγεθος (length), το οποίο δεν αλλάζει, είναι σταθερό
  - εκτός και έαν ορίσουμε ξανά τον πίνακα
- στη Java ένας πίνακας είναι ένα αντικείμενο και έχει ιδιότητες (properties)
  - `System.out.println(myArray2.length);  // τυπώνει το μέγεθος του πίνακα`

Δομές
Πίνακες - Πρόσβαση στοιχείων του πίνακα

Προσοχή!

- τα στοιχεία του πίνακα αριθμούνται 0…length-1
  - ΟΧΙ 1…length
  - `int myArray[] = {10,20,30,40,50};`

10	20	30	40	50
0	1	2	3	4

- για να προσπελάσουμε το 2ο στοιχείο του πίνακα
  - `myArray[1] += 5;`
  - `System.out.println(myArray[1]);`

Δομές
Πίνακες - Διατρέχοντας έναν πίνακα

- στη Java έχουμε δύο τρόπους να διατρέξουμε ένα πίνακα

1ος τρόπος: διατρέχουμε τα στοιχεία

```
for (<array type> element: array)
{
    // … do something with element…
}
```

```
int array[] = {1,3,5,7};
for (int element: array)
{
    System.out.println(element);
}
```

2ος τρόπος: διατρέχουμε τις θέσεις του πίνακα

```
for (int i = 0; i < array.length; i++ )
{
    // … do something with array[i]…
}
```

```
int array[] = {1,3,5,7};
for (int i = 0; i < array.length; i++ )
{
    System.out.println(array[i]);
}
```

Δομές

Πίνακες - Διατρέχοντας έναν πίνακα - Παράδειγμα

```
1.  // παράδειγμα διάσχισης πινάκων
2.
3.  class TestArrays1
4.  {
5.     public static void main(String args[])
6.     {
7.        int[] arr0;                                  // int arr0[];
8.
9.        int[] arr1 = {1, 2, 3, 4};
10.       for (int i = 0; i < arr1.length; i ++) {
11.          System.out.print(arr1[i]);                // 1234
12.       }
13.
14.       int[] arr2 = new int [10];
15.       arr0 = arr2;
16.       for (int i = 0; i < arr2.length; i ++) {
17.          arr2[i] = i+1;
18.       }
19.       for (int x: arr0) {
20.          System.out.print(x);                      // 12345678910
21.       }
22.    }
23. }
```

ισοδύναμη σύνταξη

ορισμός πίνακα, χωρίς δέσμευση χώρου

ο πίνακας arr0 δείχνει στον ίδιο χώρο μνήμης όπως και ο arr2 → οπότε, όποια αλλαγή γίνει στον ένα θα εμφανιστεί και στον άλλο

Δομές
Πίνακες - Άσκηση

- τυπώστε όλα τα στοιχεία του πίνακα και όλα τα ζεύγη από στοιχεία στον πίνακα

```
1.  // εκτύπωση στοιχείων και ζευγών στοιχείων πίνακα
2.
3.  class ScanArray
4.  {
5.     public static void main(String args[])
6.     {
7.        double[] array = {5.3, 3.4, 2.3, 1.2, 0.1};
8.
9.        for (double element: array) {                    // εκτύπωση όλων των στοιχείων
10.          System.out.println(element);
11.       }
12.
13.       for (int i = 0; i < array.length; i++) {         // εκτύπωση ζευγών στοιχείων
14.          for (int j = i+1; j < array.length; j++) {
15.             System.out.println(array[i] + " " + array[j]);
16.          }
17.       }
18.    }
19. }
```

σε αυτή την περίπτωση δε μπορούμε να διατρέξουμε τα στοιχεία, πρέπει να διατρέξουμε τις θέσεις

Δομές
Πίνακες - Πολυδιάστατοι

- μπορούμε να ορίσουμε και πολυδιάστατους πίνακες

```
int[][] myArray1 = {{10,20,30},{3,4,5}};
```

0	10	20	30
1	3	4	5
	0	1	2

```
int[][] myArray2 = new int[2][3];
```

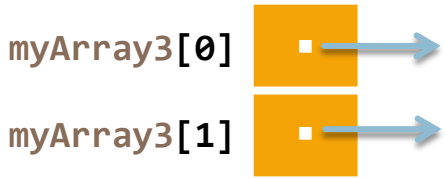
0	0	0	0
1	0	0	0
	0	1	2

Δομές
Πίνακες - Πολυδιάστατοι II

- ένας δισδιάστατος πίνακας είναι ένας πίνακας από αντικείμενα-πίνακες

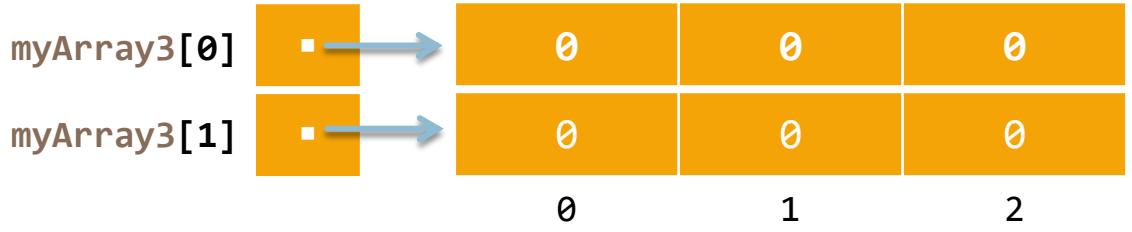
```
int[][] myArray3 = new int[2][];
```

  - ο τύπος του πίνακα είναι `int[]`, δηλαδή πίνακας από `int`



- η κάθε γραμμή είναι ένας πίνακας και πρέπει να αρχικοποιηθεί

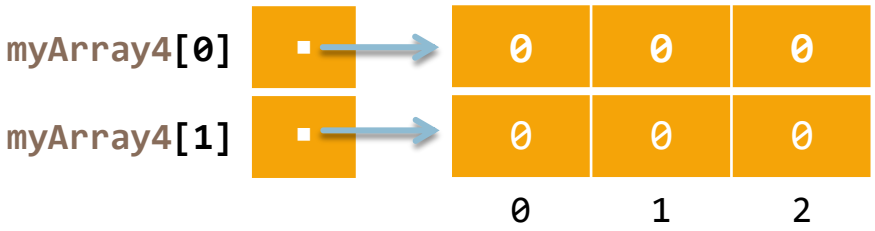
```
myArray3[0] = new int[3];
myArray3[1] = new int[3];
```



Δομές
Πίνακες - Πολυδιάστατοι III

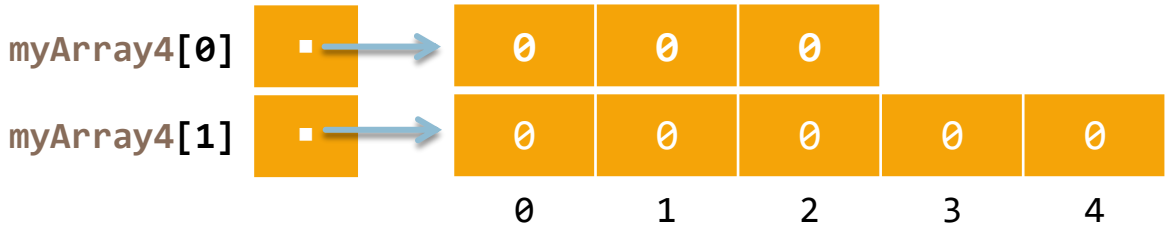
- ένας δισδιάστατος πίνακας είναι ένας πίνακας από αντικείμενα-πίνακες

```
int[][] myArray4 = new int[2][];
myArray4[0] = new int[3];
myArray4[1] = new int[3];
```



- ο πίνακας μπορεί να είναι ασύμμετρος

```
myArray4[1] = new int[5];
```



? τι εκτυπώνουν τα παρακάτω;

```
System.out.println(myArray4.length);        // 2
System.out.println(myArray4[1].length);     // 5
System.out.println(myArray4[0].length);     // 3
```

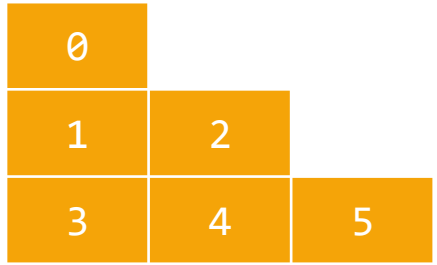
Δομές

Πίνακες - Πολυδιάστατοι - Παράδειγμα

```
1.  // εκτύπωση στοιχείων και ζευγών στοιχείων πίνακα
2.
3.  class TestArrays2
4.  {
5.     public static void main(String args[])
6.     {
7.        int[][] array1 = {{1, 2, 3}, {3, 4, 5}};     // πίνακας με αρχικές τιμές
8.        int[][] array2 = new int[10][20];            // πίνακας 10x20
9.
10.       array2 = array1;                   /* ο πίνακας array2 γίνεται ίδιος με τον array1, δηλαδή
11.                                           δείχνει στον ίδιο χώρο μνήμης. Ο προηγούμενος χώρος
12.                                           μνήμης (του array2) χάνεται */
13.
14.       System.out.println (array2.length + " " + array2[0].length);   // τυπώνει: 2 3
15.
16.       int[][] array3 = new int[2][];             // array3: ασύμμετρος πίνακας
17.       array3[0] = new int[3];
18.       array3[1] = new int[5];
19.       System.out.println(array3.length + " " +
20.                          array3[0].length + " " +
21.                          array3[1].length);      // τυπώνει: 2 3 5
22.    }
23. }
```

Δομές
Πίνακες - Αρχικοποίηση

- δημιουργήστε τους παρακάτω πίνακες:
  1. ένα μονοδιάστατο πίνακα με n θέσεις με τις τιμές `0..n-1`
  2. ένα δισδιάστατο πίνακα με nxn θέσεις με τις τιμές `0..n²-1`
  3. ένα κάτω διαγώνιο πίνακα nxn
     - π.χ. παρακάτω 3x3



- το μέγεθος του πίνακα θα το δίνει ο χρήστης

Δομές

Πίνακες - Αρχικοποίηση - Μονοδιάστατος πίνακας

```
1.  // αρχικοποίηση μονοδιάστατου πίνακα
2.
3.  import java.util.Scanner;
4.
5.  class SingleDimensionArrayInitialization
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.       int n = input.nextInt();
11.
12.       int[] array1d = new int[n];
13.       for (int i = 0; i < n; i++) {
14.          array1d[i] = i;
15.       }
16.
17.       for (int element : array1d) {
18.          System.out.print(element + " ");
19.       }
20.       System.out.println();
21.    }
22. }
```

Δομές

Πίνακες - Αρχικοποίηση - Δισδιάστατος πίνακας

```
1.  // αρχικοποίηση δισδιάστατου πίνακα
2.
3.  import java.util.Scanner;
4.
5.  class SingleDimensionArrayInitialization {
6.     public static void main(String args[]) {
7.        Scanner input = new Scanner(System.in);
8.        int n = input.nextInt();
9.
10.       int[][] array2d = new int[n][n];
11.       for (int i = 0; i < n; i++) {
12.          for (int j = 0; j < n; j++) {
13.             array2d[i][j] = i*n + j;
14.          }
15.       }
16.
17.       for (int i = 0; i < n; i++) {
18.          for (int j = 0; j < n; j++) {
19.             System.out.print(array2d[i][j] + " ");
20.          }
21.          System.out.println();
22.       }
23.    }
24. }
```

εκτυπώνουμε τα στοιχεία του πίνακα με τον 2ο τρόπο διάσχισης

Άσκηση: εκτυπώστε τα στοιχεία του πίνακα εφαρμόζοντας τον 1ο τρόπο διάσχισης

Δομές

Πίνακες - Αρχικοποίηση - Κάτω τριγωνικός πίνακας

```
1.  // αρχικοποίηση κάτω τριγωνικού πίνακα
2.
3.  import java.util.Scanner;
4.
5.  class SingleDimensionArrayInitialization {
6.     public static void main(String args[]) {
7.        Scanner input = new Scanner(System.in);
8.        int n = input.nextInt();
9.
10.       int[][] lowerDiagonal = new int[n][];
11.       for (int i = 0; i < n; i++) {
12.          lowerDiagonal[i] = new int[i+1];
13.          for (int j = 0; j < i+1;  j++) {
14.             lowerDiagonal[i][j] = i*(i+1)/2 + j;
15.          }
16.       }
17.
18.       for (int i = 0; i < n; i++) {
19.          for (int j = 0; j < i+1; j++) {
20.             System.out.print(lowerDiagonal[i][j] + " ");
21.          }
22.          System.out.println();
23.       }
24.    }
25. }
```

Άσκηση: εκτυπώστε τα στοιχεία του πίνακα εφαρμόζοντας τον 1° τρόπο διάσχισης

Δομές
Πίνακες από String - 1ο Παράδειγμα - Περιγραφή

- αναπτύξτε ένα πρόγραμμα που διαβάζει από την είσοδο το όνομα και το επώνυμο n ατόμων και τα αποθηκεύει
- το n δίνεται ως είσοδος από το χρήστη

Δομές

Πίνακες από String - 1ο Παράδειγμα - Υλοποίηση

```
1.  // αποθήκευση ονοματεπώνυμου n ατόμων
2.
3.  import java.util.Scanner;
4.
5.  class Name
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.       System.out.println("Give number of persons");
11.       int n = input.nextInt();
12.
13.       String[][] nameArray = new String[n][2];
14.       for (int i = 0; i < n; i++) {
15.          nameArray[i][0] = input.next();
16.          nameArray[i][1] = input.next();
17.       }
18.
19.       for (int i = 0; i < n; i++) {
20.          System.out.println(nameArray[i][0] + " " + nameArray[i][1]);
21.       }
22.    }
23. }
```

Δομές
Πίνακες από String - 2ο Παράδειγμα - Περιγραφή

αναπτύξτε ένα πρόγραμμα που

1. διαβάζει μία γραμμή από κείμενο και
2. ψάχνει μία λέξη που δίνουμε σαν όρισμα μέσα σε αυτή τη γραμμή

- τρόπος εκτέλεσης:

  ```
  java LookFor hello
  ```

  1. περιμένει να διαβάσει μια γραμμή από κείμενο και
  2. ψάχνει τη λέξη hello μέσα στο κείμενο

Δομές

Πίνακες από String - 2ο Παράδειγμα - Υλοποίηση

```
1.  // αναζήτηση λέξης μέσα σε γραμμή κειμένου
2.
3.  import java.util.Scanner;
4.
5.  class LookFor {
6.     public static void main(String args[]) {
7.        String name = "default";
8.        if (args.length == 1) {
9.           name = args[0];
10.       }
11.
12.       Scanner input = new Scanner(System.in);
13.       System.out.println("Give line");
14.       String line = input.nextLine();
15.       String[] words = line.split(" ");
16.       for (int i = 0; i < words.length; i++) {
17.          if (name.equals(words[i])) {
18.             System.out.println(name + " found it at " + i + "-th word");
19.          }
20.       }
21.    }
22. }
```

Τα command-line ορίσματα του προγράμματος αποθηκεύονται στον πίνακα από Strings που είναι όρισμα στην main()

η μέθοδος split της κλάσης String με όρισμα ένα delimiter String σπάει το String με βάση το delimiter και επιστρέφει ένα πίνακα από Strings

στη δική μας περίπτωση σπάμε το String line με delimeter String το κενό (" "), ώστε να πάρουμε τις λέξεις

Δομές
Κλάση ArrayList

- ορίζει έναν πίνακα μεταβλητού μεγέθους
  - το μέγεθος του πίνακα αλλάζει δυναμικά ανάλογα με τον αριθμό των στοιχείων που τοποθετούμε
- διατηρεί αντικείμενα συγκεκριμένου τύπου

```
1.  // ArrayList Παραδείγματα
2.
3.  import java.util.ArrayList;
4.  ...
5.     ArrayList<Integer> myList1;  // λίστα από ακεραίους
6.     ArrayList<String> myList2;   // λίστα από συμβολοσειρές
7.  ...
```

- συντακτικό:

```
import java.util.ArrayList;

ArrayList<Τύπος> myList;
```

  - ο τύπος είναι οποιαδήποτε κλάση
    - αυτός είναι ο τύπος των δεδομένων που αποθηκεύει ο πίνακας μας
  - για να αποθηκεύσουμε βασικούς τύπους χρειαζόμαστε την αντίστοιχη wrapper class

Δομές
Κλάση ArrayList II

- δημιουργία αντικειμένου
  - `ArrayList<T> myList = new ArrayList<T>();`
- μέθοδοι
  - `add(T x)`: προσθέτει το στοιχειό `x` στο τέλος του πίνακα.
  - `add(int i, T x)`: προσθέτει το στοιχειό `x` στη θέση `i` και μετατοπίζει τα υπόλοιπα στοιχεία κατά μια θέση
  - `get(int i)`: επιστρέφει το αντικείμενο τύπου `T` στη θέση `i`
  - `remove(int i)`: αφαιρεί το στοιχείο στη θέση `i`
  - `remove(T x)`: αφαιρεί το στοιχείο `x`
  - `set(int i, T x)`: θέτει στη θέση `i` την τιμή `x` αλλάζοντας την προηγούμενη
  - `size()`: το πλήθος των στοιχείων του πίνακα
- διατρέχοντας τον πίνακα:
  - `for(T x: myList){…}`

Δομές
Κλάση ArrayList - Παράδειγμα - Περιγραφή

αναπτύξτε ένα πρόγραμμα όπου

1. επαναληπτικά θα ζητάτε από την είσοδο ακεραίους αριθμούς
   - μέχρι ο χρήστης να δώσει το -1
2. θα αποθηκεύετε τους αριθμούς σε ένα πίνακα
3. θα εκτυπώσετε τους αριθμούς

<u>σημείωση</u>:

- δε γνωρίζουμε εκ των προτέρων το πλήθος των αριθμών που θα χρειαστεί να αποθηκεύσουμε
  - ούτε και ο χρήστης το γνωρίζει → άρα δε μπορεί να μας το δώσει ως είσοδο
  - ☞ θα χρησιμοποιήσουμε `ArrayList` αντί για πίνακα

Δομές

Κλάση ArrayList - Παράδειγμα - Υλοποίηση

```
1.  // διαβάζει (από το χρήστη), αποθηκεύει και εκτυπώνει άγνωστο πλήθος ακεραίων
2.
3.  import java.util.ArrayList;
4.  import java.util.Scanner;
5.
6.  class ArrayListTest
7.  {
8.     public static void main(String args[])
9.     {
10.       ArrayList<Integer> numbers = new ArrayList<Integer>();
11.       Scanner input = new Scanner(System.in);
12.       int x = input.nextInt();
13.       while (x != -1) {
14.          numbers.add(x);
15.          x = input.nextInt();
16.       }
17.
18.       for (Integer y : numbers) {
19.          System.out.print(y + " ");
20.       }
21.       System.out.println();
22.    }
23. }
```

Δομές
Πίνακες από String - 3ο Παράδειγμα - Περιγραφή

- αναπτύξτε ένα πρόγραμμα που να διαβάζει από την είσοδο όνομα και επώνυμο ενός ατόμου, μέχρι να τερματίσουμε την είσοδο
  - π.χ. μέχρι να δώσουμε τη συμβολοσειρά `"exit"` ως όνομα ατόμου

- <u>σημείωση</u>:
  - δε γνωρίζουμε εκ των προτέρων το πλήθος των ατόμων που θα χρειαστεί να αποθηκεύσουμε
    - ούτε και ο χρήστης το γνωρίζει → άρα δε μπορεί να μας το δώσει ως είσοδο

Δομές

Πίνακες από String - 3ο Παράδειγμα - Υλοποίηση

```
1.  // διαβάζει (από το χρήστη), αποθηκεύει και εκτυπώνει άγνωστο πλήθος ονομάτων
2.
3.  import java.util.ArrayList;
4.  import java.util.Scanner;
5.
6.  class ArrayListTest
7.  {
8.     public static void main(String args[])
9.     {
10.       ArrayList<String[]> nameList = new ArrayList<String[]>();
11.       Scanner input = new Scanner(System.in);
12.       while (input.hasNext()) {
13.          String[] nameArray = new String[2];
14.          nameArray[0] = input.next();
15.          if (nameArray[0].equals("exit")) {
16.               break;
17.          }
18.          nameArray[1] = input.next();
19.          nameList.add(nameArray);
20.       }
21.
22.       for (String[] nameArray : nameList) {
23.          System.out.println(nameArray[0] + " " + nameArray[1]);
24.       }
25.    }
26. }
```

ορισμός λίστας από πίνακες String

επιστρέφει true, εάν υπάρχει λέξη να διαβαστεί

Δομές
Σύνολα

- οι πίνακες και η κλάση `ArrayList` μας επιτρέπουν να κρατάμε μια διατεταγμένη σειρά από αντικείμενα
- αν δε μας ενδιαφέρει η διάταξη, αλλά μόνο η συλλογή των αντικειμένων → τότε, μπορούμε να τα αποθηκεύσουμε σε ένα σύνολο
  - τα σύνολα επιτρέπουν γρήγορη αναζήτηση μιας τιμής μέσα στη συλλογή

- `HashSet`: μια κλάση στη Java για να ορίσουμε ένα σύνολο από αντικείμενα
- συντακτικό:

```
import java.util.HashSet;

HashSet<Τύπος> mySet;
```

Δομές
Κλάση HashSet

- δημιουργία αντικειμένου
  - `HashSet<T> mySet = new HashSet<T>();`
- μέθοδοι
  - `add(T x)`: προσθέτει το στοιχειό `x`, αν δεν υπάρχει ήδη
  - `remove(T x)`: αφαιρεί το στοιχείο `x`
  - `contains(T x)`: επιστρέφει `true` αν το σύνολο περιέχει το στοιχείο `x`, αλλιώς επιστρέφει `false`
  - `isEmpty()`: επιστρέφει `true` αν το σύνολο είναι άδειο, αλλιώς επιστρέφει `false`
  - `size()`: το πλήθος των στοιχείων στο σύνολο
- διατρέχοντας τα στοιχεία του συνόλου
  - `for(T x: mySet){…}`

Δομές
Κλάση HashSet - Παράδειγμα - Περιγραφή

- διαβάζουμε ένα String που περιέχει ένα κείμενο
- θέλουμε να βρούμε όλες τις μοναδικές λέξεις στο κείμενο
  - π.χ. να φτιάξουμε το λεξικό ενός βιβλίου

Δομές

Κλάση HashSet - Παράδειγμα - 1η Υλοποίηση

```
1.  // διαβάζει (από το χρήστη), αποθηκεύει και εκτυπώνει μοναδικές λέξεις
2.
3.  import java.util.HashSet;
4.  import java.util.Scanner;
5.
6.  class HashSetExample
7.  {
8.     public static void main(String args[])
9.     {
10.       HashSet<String> mySet = new HashSet<String>();
11.       Scanner input = new Scanner(System.in);
12.
13.       String text = input.nextLine();
14.       String[] words = text.split(" ");
15.       for (String word : words) {
16.          if (!mySet.contains(word))
17.             mySet.add(word);
18.       }
19.
20.       for (String word : mySet) {
21.          System.out.println(word);
22.       }
23.    }
24. }
```

δήλωση μιας μεταβλητής HashSet από Strings

τοποθετούμε στο HashSet μόνο τα Strings τα οποία δεν έχουμε ήδη δει (δεν είναι ήδη στο σύνολο)

διατρέχουμε και τυπώνουμε τα στοιχεία του HashSet

Δομές
Κλάση HashSet - Παράδειγμα - 2η Υλοποίηση

```
1.  // διαβάζει (από το χρήστη), αποθηκεύει και εκτυπώνει μοναδικές λέξεις
2.
3.  import java.util.HashSet;
4.  import java.util.Scanner;
5.
6.  class HashSetExample
7.  {
8.     public static void main(String args[])
9.     {
10.       HashSet<String> mySet = new HashSet<String>();
11.       Scanner input = new Scanner(System.in);
12.
13.       String text = input.nextLine();
14.       String[] words = text.split(" ");
15.       for (String word : words) {
16.          mySet.add(word);
17.       }
18.
19.       for (String word : mySet) {
20.          System.out.println(word);
21.       }
22.    }
23. }
```

επειδή το HashSet κρατάει μοναδικά αντικείμενα, δε χρειάζεται να κάνουμε τον έλεγχο (καθώς αν υπάρχει ήδη το String δε θα το προθέσει ξανά)

Δομές
HashMap

- οι πίνακες και η κλάση `ArrayList` μας επιτρέπουν να κρατάμε μια διατεταγμένη σειρά από αντικείμενα και τα δεικτοδοτούν με τη θέση τους
- αν θέλουμε να δεικτοδοτήσουμε με κάποιο κλειδί → χρειαζόμαστε λεξικό
  - τα λεξικά κρατάνε ζευγάρια της μορφής (κλειδί,τιμή) και επιτρέπουν γρήγορη αναζήτηση με το κλειδί

- `HashMap`: μια κλάση στη Java για να ορίσουμε λεξικό
- συντακτικό:

```
import java.util.HashMap;

HashMap<Τύπος1 κλειδιού,Τύπος2 τιμής> myMap;
```

Δομές
Κλάση HashMap

- δημιουργία αντικειμένου
  - `HashMap<K,V> myMap = new HashMap<K,V>();`
- μέθοδοι
  - `put(K key, V value)`: προσθέτει το ζευγάρι (`key`, `value`) → δημιουργεί μια συσχέτιση
    - εάν το κλειδί (`key`) υπάρχει ήδη, ενημερώνει την τιμή του ζευγαριού (θέτοντας την `value`)
  - `V get(K key)`: επιστρέφει την τιμή για το κλειδί `key`
  - `remove(K key)`: αφαιρεί το ζευγάρι με κλειδί `key`
  - `containsKey(K key)`: επιστρέφει `true` αν το σύνολο περιέχει το κλειδί `key`, αλλιώς επιστρέφει `false`
  - `containsValue(V value)`: επιστρέφει `true` αν το σύνολο περιέχει την τιμή `value`, αλλιώς επιστρέφει `false`
  - `isEmpty()`: επιστρέφει `true` αν το λεξικό είναι άδειο, αλλιώς επιστρέφει `false`
  - `Set<K> keySet()`: επιστρέφει ένα `Set` με τα κλειδιά
  - `size()`: το πλήθος των στοιχείων (κλειδιών) στο λεξικό

Δομές
Κλάση HashMap - Παράδειγμα - Περιγραφή

- διαβάζουμε ένα String που περιέχει ένα κείμενο
- θέλουμε να βρούμε όλες τις μοναδικές λέξεις στο κείμενο και τον αριθμό των εμφανίσεων τους

Δομές

Κλάση HashMap - Παράδειγμα - Υλοποίηση

```
1.  // διαβάζει (από το χρήστη), αποθηκεύει και εκτυπώνει μοναδικές λέξεις και το πλήθος εμφάνισής τους
2.  import java.util.HashMap;
3.  import java.util.Scanner;
4.  class HashMapExample
5.  {
6.     public static void main(String args[])
7.     {
8.        HashMap<String, Integer> myMap = new HashMap<String, Integer>();
9.        Scanner input = new Scanner(System.in);
10.
11.       String text = input.nextLine();
12.       String[] words = text.split(" ");
13.       for (String word : words) {
14.          if (!myMap.containskey(word)) {
15.             myMap.put(word,1);
16.          } else {
17.             myMap.put(word, myMap.get(word)+1);
18.          }
19.       }
20.
21.       for (String word : myMap.keySet()) {
22.          System.out.println(word + " " + myMap.get(word));
23.       }
24.    }
25. }
```

Δήλωση μιας μεταβλητής HashMap που συσχετίζει Strings(κλειδια) και Integers(τιμές). Για κάθε λέξη (String) τον αριθμό ενφανίσεων(Integer)

αν η λέξη δεν είναι ήδη στο HashMap τότε πρόσθεσε την με τιμή 1

Αλλιώς πάρε την τιμή της λέξης, αύξησε την κατά 1, και βάλε τη πίσω σαν νέα τιμή

Διέτρεξε το σύνολο με τα κλειδιά (ονόματα) στο HashMap. Για κάθε κλειδί πάρε και τύπωσε την τιμή του.

Σύνοψη

- Αντικειμενοστραφής προγραμματισμός
- Κλάσεις και Αντικείμενα
  - γενική μορφή και πρακτικά στον κώδικα
  - δημιουργία αντικειμένων και κλήση μεθόδων
- Υπάρχουσες κλάσεις
  - String (ισότητα και σταθερές)
  - Scanner και Wrapper classes
  - Αμετάβλητα αντικείμενα
- Δομές
  - Πίνακες (μονοδιάστατοι, πολυδιάστατοι, ασύμμετροι, αρχικοποίηση, πίνακες από string)
  - ArrayList
  - HashSet, Set
  - HashMap