Βασικά στοιχεία προγραμματισμού στη JAVA

Τελεστές, Επαναλήψεις, Πίνακες και Παραδείγματα

Οι τελεστές της JAVA

- Η Java χρησιμοποιεί τους ίδιους τελεστές με την C και την C++, και κατά τον ίδιο γενικά τρόπο.
- *Σχεδόν όλοι* οι τελεστές εφαρμόζονται **μόνο** σε αρχέγονους τύπους.
  - Εξαίρεση αποτελούν οι τελεστές: **'='**, **'=='** και **'!='**, οι οποίοι εφαρμόζονται *και σε αντικείμενα*.
  - Επίσης, οι τελεστές **'+'** και **'+='** εφαρμόζονται και σε αντικείμενα της κλάσης **String.**
- **Κανόνες προτεραιότητας** (precedence): ο πολλαπλασιασμός και η διαίρεση έχουν προτεραιότητα έναντι της πρόσθεσης και της αφαίρεσης.

Ανάθεση

- Η ανάθεση σε μεταβλητές αρχέγονων τύπων είναι απλή και ακολουθεί γνωστούς κανόνες.
- Η ανάθεση ενός αντικειμένου **A** σε ένα άλλο **B**, προϋποθέτει την ανάθεση του χειριστηρίου **a,** του **A,** στο χειριστήριο **b** του **B:**
  - **b = a;**
  - Αποτέλεσμα της ανάθεσης αυτής είναι το b και το a να δείχνουν στο ίδιο αντικείμενο **(A),** ενώ η πρόσβαση στο αντικείμενο **B** έχει χαθεί.
- Το φαινόμενο αυτό λέγεται **aliasing (**ψευδωνυμία) και είναι αποτέλεσμα του πως η Java χειρίζεται τα αντικείμενα.
- Η ψευδωνυμία προκύπτει και όταν περνάμε αντικείμενα σαν παραμέτρους σε μια μέθοδο.

Πέρασμα Ορισμάτων κ. Ψευδωνυμία

```
class Letter { char c; }
public class PassObject {
    static void f(Letter y) { y.c = 'z'; }
    public static void main(String[] args) {
            Letter x = new Letter();
            x.c = 'a';
            System.out.println("1: x.c: " + x.c);
            f(x);
            System.out.println("2: x.c: " + x.c);
}}
```

- Σε ορισμένες γλώσσες η μέθοδος **f()** θα έκανε ένα αντίγραφο του ορίσματος **Letter y,** μέσα στο πεδίο ισχύος (scope) της (**pass-by-value**).
- Στην Java όμως περνάμε το χειριστήριο σαν παράμετρο (**pass-by-reference**), κι έτσι μέσα στην **f()** αλλάζουμε το ίδιο το αντικείμενο.

Τελεστές

- Μαθηματικοί τελεστές: όπως στην C
- **Συσχετιστικοί τελεστές-relational operators**
  - Δημιουργούν αποτελέσματα τύπου **boolean**.
  - Δέχονται κατηγορήματα αρχέγονου τύπου.
  - Οι τελεστές σύγκρισης (>,<,<=, >=) δεν δέχονται κατηγορήματα **boolean.**

Λογικοί Τελεστές

- Όπως στην C: AND(&&), OR(||), NOT(!)
- Επιστρέφουν **boolean** τιμές.
- Δέχονται σαν κατηγορήματα μόνο **boolean** μεταβλητές.
- **Δεν επιτρέπεται** η χρήση ακέραιων κατηγορημάτων με λογικούς τελεστές (όπως π.χ. στην C): i && j
- **Βραχυκύκλωμα (short circuiting):** κατά την αποτίμηση μιας λογικής έκφρασης (δηλ. έκφρασης με λογικούς τελεστές), η αποτίμηση συνεχίζεται μέχρι να γίνει σαφής η αλήθεια ή το σφάλμα της έκφρασης. Π.χ.:

  if (test1(0) && test2(2) && test3(2)) ...

Τελεστές Δυαδικών Ψηφίων

- Επιτρέπουν την επεξεργασία των δυαδικών ψηφίων μεταβλητών αρχέγονου **ακέραιου** τύπου (int, long, short, byte, char).
- Εκτελούν πράξεις άλγεβρας bool στα αντίστοιχα bits των κατηγορημάτων τους και δίνουν το αποτέλεσμα.
- Τελεστές: **AND(&), OR (|), XOR(^), NOT(~).**
- Επίσης: &=, |=, ^= (δεν επιτρέπεται όμως το ~=)

  Οι τύποι **boolean** σε συνδυασμό με τους δυαδικούς τελεστές, αντιμετωπίζονται σαν τιμές ενός bit.
- Η Java προσφέρει επίσης τελεστές διολίσθησης (shift operators), που δέχονται μεταβλητές αρχέγονου **ακέραιου** τύπου (int, long, short, byte, char)
- **Τριαδικός Τελεστής:**

  boolean-exp?value0:value1

Μετατροπή Τύπων - casting

- Η Java μετατρέπει δεδομένα ενός τύπου σε δεδομένα άλλου τύπου *αυτομάτως*, όποτε αυτό χρειαστεί και όταν επιτρέπεται.
- Η **μετατροπή τύπων** (type casting) μας επιτρέπει να κάνουμε αυτόν τον μετασχηματισμό ρητά ή να τον εξαναγκάσουμε να γίνει, σε περιπτώσεις που δεν γίνεται αυτόματα.

```
void casts() {
    int i = 200;
    long l = (long) i;  // γίνεται και αυτόματα
    long l2 = (long) 200; // γίνεται και αυτόματα
}
```

- **Μετασχηματισμός συρρίκνωσης** (narrowing conversion): δεν γίνεται αυτόματα, και όταν γίνει υπάρχει ο κίνδυνος να χαθεί πληροφορία. Η Java επιτρέπει την μετατροπή συρρίκνωσης από και προς οποιονδήποτε αρχέγονο τύπο εκτός **boolean.**
- Μετατροπή τύπων από μιά κλάση σε κάποια άλλη **δεν** επιτρέπεται.

Παράδειγμα μετατροπών

```
char c = (char) (Math.random() * 26 + 'a');
```

- Math.random() επιστρέφει **double.**
- Το 26 μετατρέπεται σε double για να γίνει ο πολλαπλασιασμός.
- Το 'a' μετατρέπεται σε double για να γίνει η πρόσθεση.
- Το αποτέλεσμα της πρόσθεσης μετασχηματίζεται σε char με αποκοπή.

sizeof

- Στην C και στην C++, η συνάρτηση sizeof() μας επιστρέφει τον αριθμό των χαρακτήρων που έχουν δεσμευθεί για κάποιο δεδομένο.
- Ο κυριότερος λόγος χρήσης της sizeof() είναι η φορητότητα (portability). Διάφοροι τύποι δεδομένων μπορεί να έχουν διαφορετικό μέγεθος σε διαφορετικές μηχανές. Συνεπώς, ο προγραμματιστής πρέπει να μπορεί να βρει το ακριβές μέγεθος ενός τύπου.
- Στην Java **δεν χρειάζεται sizeof()** διότι σε όλα τα μηχανήματα οι τύποι δεδομένων έχουν το ίδιο μέγεθος.

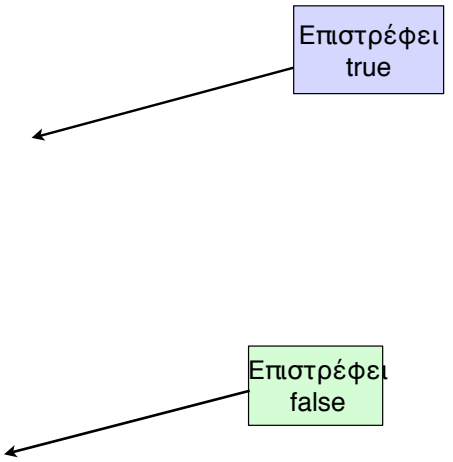
Έλεγχος ισοδυναμίας αντικειμένων

- Ισότητα/Ισοδυναμία αντικειμένων.
  - Πώς ορίζεται;
  - Πώς ελέγχεται;
- Για τον έλεγχο των περιεχομένων ενός αντικειμένου για ισοδυναμία με άλλο αντικείμενο, χρησιμοποιείται η μέθοδος **equals**, η οποία υπάρχει σε όλα τα αντικείμενα.

Έλεγχος ισοδυναμίας αντικειμένων

```
public class EqualsMethod {
    public static void main(String[] args) {
        Integer n1 = new Integer(47);
        Integer n2 = new Integer(47);
        System.out.println(n1.equals(n2));
    }}

class Value { int i; }
public class EqualsMethod2 {
    public static void main(String[] args) {
        Value v1 = new Value();
        Value v2 = new Value();
        System.out.println(v1.equals(v2));
    }}
```



```
class Letter { char c; }
public class PassObject {
    static void f(Letter y) { y.c = 'z'; }
    public static void main(String[] args) {
        Letter x = new Letter();
        x.c = 'a';
        System.out.print("1: " + x.c);
        f(x);
        System.out.println(", 2: " + x.c); }}
```

Τι τυπώνει το πρόγραμμα;

- **1: a, 2: z**
- **1: a, 2: a**
- **1: z, 2: z**
- **Τίποτε από τα πιο πάνω**

Έλεγχος εκτέλεσης

- Οι εντολές της Java είναι παρόμοιες με αυτές της C και της C ++.
- if-else:

```
if (boolean-expression)
        statement
else
        statement
```

- return:
  - χρησιμοποιείται για να καθορίσει την τιμή που επιστρέφει μια μέθοδος
  - επιβάλλει την επιστροφή της τιμής αυτής αμέσως

Επαναλήψεις - Iterations

- **while** (boolean-expression)
  statement
- **do**
  statement
  while (Boolean-expression);
- **for** (initialization; Boolean-expression; step)
  statement

```
public class ListCharacters {
    public static void main(String[] args) {
        for (char c =0; c<128; c++)
          if (c != 26) //ANSI clear screen
                System.out.println("value: " + (int) c + "character: " + c);
        }
}
```


Ο τελεστής «,»

```
for (int i = 0, j = 1; i < 10 && j != 11; i++, j++ )
    /* body of the loop */
```

- Ο τελεστής «,» μπορεί να χρησιμοποιηθεί μόνο σε εντολές αρχικοποίησης και βήματος τής εντολής **for**.
- Στην αρχικοποίηση τού **for** μπορούμε να ορίσουμε περισσότερες της μιας μεταβλητές, του ίδιου όμως τύπου.

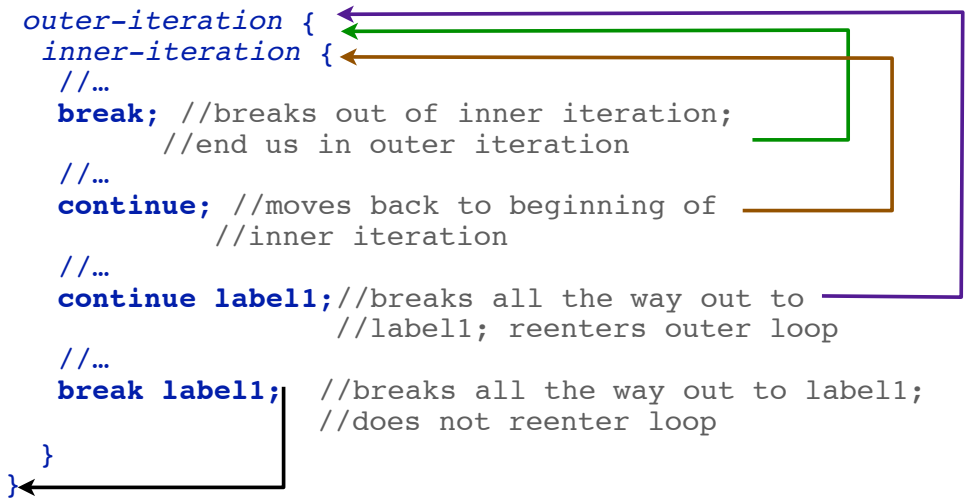
break και continue

```
public class BreakAndContinue {
  public static void main(String[] args) {
     for (int i=0; i<100; i++) {
       if (i==74) break; // out of loop
       if (i % 9 != 0) continue; // next iteration
              System.out.println(i);
    }
    int i = 0;
    while (true) {
      i++;
      int j = i * 27;
      if (j == 1296) break; // out of loop
      if (I % 10 != 0) continue; // top of loop
      System.out.println(i);
  }}}
```


goto και labels

- Το goto στην Java είναι δεσμευμένη λέξη αλλά δεν χρησιμοποιείται ως εντολή.
- Ωστόσο η Java χρησιμοποιεί **labels**, σε συνδυασμό με εντολές **continue** ή **break** για να επιτρέπει την έξοδο από φωλιασμένους βρόχους.

labels



switch

```
switch (integral-expression) {
  case integral-value1: statement; break;
  case integral-value2: statement; break;
  case integral-value3: statement; break;
  case integral-value4: statement; break;
     // …
  default: statement;

}
```

- Η χρήση του **break**.
- **integral-expression**: integer or char (or *enum* constants).

*Για ποιό λόγο δεν υπάρχει **sizeof** στη JAVA;*

- **Έχει αποκλειστεί από τους σχεδιαστές της για την αποφυγή σφαλμάτων στρογγυλοποίησης**
- **Δεν χρειάζεται - τα προγράμματα JAVA τρέχουν μέσα στη JVM**
- **Την προσθέτει αυτόματα ο μεταγλωττιστής όπου χρειαστεί**
- **Οι τύποι έχουν πλέον προκαθορισμένο μέγεθος σε όλες τις αρχιτεκτονικές**

Πίνακες στη JAVA

Πίνακες στην JAVA

- Οι πίνακες στην JAVA έχουν σχεδιαστεί κατά τρόπον ώστε να ξεπερνιούνται οι δυσκολίες του προγραμματισμού πινάκων της C/C ++.
- Στη JAVA είναι εξασφαλισμένο ότι για να χρησιμοποιηθεί **ένας πίνακας θα πρέπει πρώτα να αρχικοποιηθεί** και ότι **δεν** θα επισυμβεί **πρόσβαση εκτός των ορίων του**.
- Τα χαρακτηριστικά αυτά υλοποιούνται με κάποιο σχετικό κόστος μνήμης και χρόνου εκτέλεσης (κατά την εκτέλεση γίνεται έλεγχος κατά πόσο δεν γίνεται υπέρβαση των ορίων του πίνακα).
- Κατά τη δημιουργία ενός πίνακα, κατ' ουσίαν δημιουργείται ένας πίνακας χειριστηρίων (Handles), τα οποία αρχικοποιούνται σε null.
- Είναι ευθύνη του προγραμματιστή να αρχικοποιήσει σωστά τα χειριστήρια, ώστε να παραπέμπουν σε αντικείμενα.

Πίνακες στη JAVA

- Ακολουθίες από αντικείμενα ή μεταβλητές αρχέγονου τύπου, οι οποίες ομαδοποιούνται κάτω από το ίδιο όνομα.
- Δήλωση πινάκων:

  int[] a ;

  int a[];
- Παρατηρείστε ότι δεν επιτρέπεται να δηλώσουμε το μέγεθος του πίνακα στην εντολή της δήλωσής του.
  - Κατ ' ουσίαν, στο σημείο της δήλωσης δημιουργείται μόνο το χειριστήριο του πίνακα.
  - Για την κράτηση μνήμης για τον πίνακα, χρειάζεται να γράψουμε μια έκφραση *αρχικοποίησης*, η οποία μπορεί να βρίσκεται οπουδήποτε στον κώδικα (πριν το σημείο στο οποίο γίνεται χρήση του πίνακα).

Αρχικοποίηση Πινάκων (συνέχεια)

- Αρχικοποίηση ενός πίνακα μπορεί να γίνει και στο σημείο της δήλωσής του, ως εξής:

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

- Το πεδίο **length** υπάρχει σε όλους τους πίνακες και δίνει τον αριθμό των στοιχείων τους (δεν επιτρέπεται ανάθεση σε αυτό).
- Σε περίπτωση που προσπαθήσουμε να ξεπεράσουμε το μήκος ενός πίνακα, λαμβάνουμε εξαίρεση την στιγμή της εκτέλεσης (**exception**).

Αρχικοποίηση Πινάκων (συνέχεια)

- Εαν δεν γνωρίζουμε το μήκος ενός πίνακα πριν την εκτέλεση του προγράμματός μας, θα πρέπει να χρησιμοποιήσουμε τη **new** για την κράτηση τής μνήμης του πίνακα, κατά την εκτέλεση του προγράμματος

Ποιό είναι το μήκος του πίνακα a;

```
import java.util.*;
public class ArrayNew {
   static Random rand=new Random();
   static int pRand(int mod) {
        return Math.abs(rand.nextInt())%mod+1;
   }
   public static void main(String[] args) {
        int[] a = new int[pRand(20)];
        for(int i=0;i<a.length; i++)
                System.out.println(a[i]);
   } }
```

- **20**
- **Από 0 ως 20**
- **Από 1 ως 20**
- **Κανένα από τα πιο πάνω. Η αρχικοποίηση είναι λάθος.**

00:40 00:40

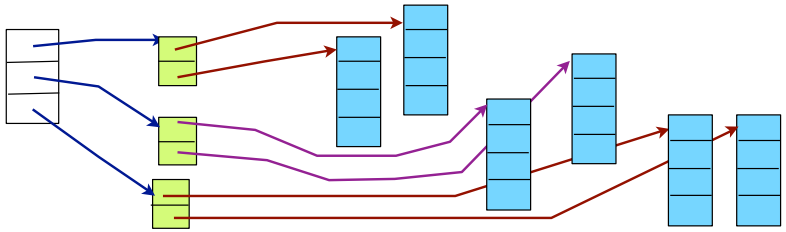
Αρχικοποίηση Πινάκων (συνέχεια)

- Σε περίπτωση πινάκων με στοιχεία *μή-αρχέγονου τύπου*, πρέπει να χρησιμοποιείται η **new** για την αρχικοποίηση των στοιχείων του πίνακα.

```
import java.util.*;
public class ArrayClassObj {
        static Random rand = new Random();
        static int pRand(int mod) {
           return Math.abs(rand.nextInt()) % mod + 1;
        }
public static void main(...)  {
        int q, p = pRand(20);
        Integer[] a=new Integer[p];
        for(int i=0;i<a.length;i++){
                        q = pRand(500);
                        a[i] = new Integer(q);
        }
}
```

Πολυδιάστατοι Πίνακες

```
int[ ][ ]  a1 = {{ 1, 2, 3, },{ 4, 5, 6, },};
// 3-D array with fixed length:
int[ ][ ][ ] a2 = new int[3][2][4];
for(int i = 0; i<a2.length; i++)
  for(int j=0;j<a2[i].length;j++)
       for(int k=0; k <a2[i][j].length;k++){  }
```



Πολυδιάστατοι Πίνακες

```
// 3-D array with varied-length vectors:
   int[ ][ ][ ]   a3 = new int[q][ ][ ];
         for(int i = 0; i< a3.length; i++) {
                  a3[i] = new int[p][];
                  for(int j=0; j < a3[i].length; j++)
                           a3[i][j] = new int[r];
         }
```


Μερικά απλά υπολογιστικά προβλήματα

Εισαγωγή στην χαρτοπαιξία!

- Μια τράπουλα αποτελείται από:
  - 52 χαρτιά
  - Τέσσερα “χρώματα” χαρτιών:
    - σπαθί (clubs) ♣
    - καρρώ (diamonds) ♦
    - καρδιά (hearts) ♥
    - πίκα (spades)♠
  - Σε κάθε χρώμα υπάρχουν 13 χαρτιά, με την ακόλουθη διαβάθμιση: 2, 3, 4, 5, 6, 7, 8, 9, 10, Αξιωματικός (Jack), Βασίλισσα (Queen), Βασιλιάς (King), Άσσος (Ace)

Η τράπουλα σε πίνακες..

```
String[] suit = { “Clubs”, “Diamonds”, “Hearts”, “Spades”};

String[] rank = { “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”,
                  “Jack”, “Queen”, “King”, “Ace” };

String[] deck = new String[52];

for (int i = 0; i<13; i++)
    for (int j = 0; j<4; j++)
        deck[4*i+j] = rank[i] + “ of ” + suit[j];
```


Ανακάτεμα της τράπουλας

```
int N = deck.length;


for (int i=0; i<N; i++) {
    int r = i + (int) (Math.random() * (N-i));
    String t = deck[i];
    deck[i] = deck[r];
    deck[r] = t;

}
```

επιστρέφει τυχαίο αριθμό μεταξύ i και N