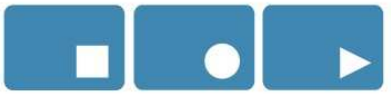
ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
ΕΥΡΩΠΑΪΚΟ ΤΑΜΕΙΟ ΠΕΡΙΦΕΡΕΙΑΚΗΣ ΑΝΑΠΤΥΞΗΣ



ΠΑΙΔΕΙΑ ΜΠΡΟΣΤΑ
2ο Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης

Πρόγραμμα Αναμόρφωσης Προπτυχιακών Σπουδών

Τμήματος Μαθηματικών

Πανεπιστημιο Πατρων

Μάθημα: Εισαγωγή στην Επιστήμη των Υπολογιστών

13η Εβδομαδα - 10 Ιανοταριοτ 2006

Θέμα: Σύντομη Εισαγωγή στην Γλώσσα Αντικειμενοστραφούς Προγραμματισμού JAVA

α. ΕΙΣΑΓΩΓΙΚΑ ΓΙΑ ΤΗΝ «JAVA»

Η **JAVA είναι μια γλώσσα προγραμματισμού**, αντικειμενοστραφούς, που την επινόησε η SUN Microsystems (μια εταιρεία γνωστή για τους προηγμένους σταθμούς εργασίας που διαθέτει) **με στόχο να είναι μικρή, απλή και «μεταφερτή»** (portable) **μεταξύ των διαφόρων υπολογιστών και λειτουργικών συστημάτων**, τόσο σε πηγαίο κώδικα (source) όσο και σε δυαδικό (binary) αρχείο. Έτσι, προγράμματα που γράφονται στη JAVA μπορούν να τρέχουν σε οποιοδήποτε υπολογιστή που έχει εγκατεστημένο την virtual machine (εικονικό υπολογιστή) της JAVA.

Το πολύ σημαντικό σχετικά με το διαδίκτυο (World Wide Web) είναι ότι εφαρμογές Browser (Navigator της Netscape και Internet Explorer της Microsoft) **είναι συμβατές με τη JAVA** και κατά συνέπεια μπορούν να μεταφέρουν και να τρέχουν προγράμματα της JAVA, τις μίνι εφαρμογές - applets, στο σύστημα του χρήστη.

Αυτές οι μίνι εφαρμογές, που μεταφέρονται από το WWW, είναι ίσως η πιο δημοφιλής χρήση της JAVA, **η οποία είναι μια πλήρως λειτουργική γλώσσα γενικού σκοπού**.

β. ΣΧΗΜΑΤΙΚΗ ΠΑΡΑΣΤΑΣΗ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΗΣ JAVA

1. **Ανάπτυξη** του αλγορίθμου επίλυσης

2. **Συγγραφή** του πηγαίου κώδικα με κάποιον editor (vi ή notepad), ως file:

$$"ονομα".java$$

3. **Μεταγλώττιση** του πηγαίου κώδικα με javac, με κλήση:

$$javac \quad "ονομα".java$$

και παραγωγή του αρχείου των bytecodes:

$$"ονομα".class$$

4. **Εκτέλεση** του αρχείου των bytecodes από java, με την εντολή:

$$java \quad "ονομα"$$

γ. Παραδείγματα Προγραμμάτων

(1ο: **Χαιρετισμός του κόσμου**, 2ο: **Εκτύπωση των 20 πρώτων όρων της σειράς** Fibonacci και 3ο **Το παιχνίδι** Fizzbuzz)

1.
```
public class greeting{
   public static void main (String[] args){
      System.out.println("Hello World!");
      }
   }
```

2.
```
public class fibonacci{
   public static void main (String[] args){
      int no=1, n1=1, n2;
      System.out.print(no +" " +n1+" ");
      for(int i=0;i<20;i++){
         n2 = n1 + no;
         System.out.print(n2+" ");
         no = n1;
         n1 = n2;
         }
      System.out.println();
      }
   }
```

3.
```
public class fizzbuzz{
   public static void main (String[] args){
      for(int i=0;i<100;i++){
         if(( (i%5) == 0) && ((i%7) == 0))
            System.out.println("fizzbuzz");
         else if( (i%5) == 0)
            System.out.println("fizz");
         if( (i%7) == 0)
            System.out.println("buzz");
         else System.out.print(i);
      System.out.println();
      }
   }
```

Παρατήρηση: Στην JAVA **διαφοροποιούνται τα πεζά** γράμματα από **τα κεφαλαία**. Έτσι, οι εντολές εκτύπωσης:

$$System.out.print$$

πρέπει να έχουν το πρώτο γράμμα με κεφαλαίο S. Όμοια **η δήλωση String**, μετά το main() πρέπει να έχει κεφαλαίο S.

δ. ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ ΤΗΣ JAVA

1. Μεταβλητές και τύποι δεδομένων

Οι μεταβλητές της JAVA είναι ονόματα που αρχίζουν με (i) γράμμα, (ii) με τον χαρακτήρα _ (κάτω παύλα) και (iii) το σύμβολο του δολαρίου $, και μπορούν στη συνέχεια να περιλαμβάνουν οποιοδήποτε αλφαριθμητικό χαρακτήρα του **Unicode , με διάκριση** των πεζών και κεφαλαίων χαρακτήρων· π.χ. η μεταβλητή World είναι διαφορετική από την world, λόγω του πρώτου χαρακτήρα. Εξάλλου στη JAVA, **προτού χρησιμοποιηθεί μια μεταβλητή πρέπει να έχει δηλωθεί**, όπου καθορίζεται ο τύπος της μεταβλητής που μπορεί να είναι:

i) Ένας από τους οκτώ (8) **πρωτογενείς τύπους** (primitive) δεδομένων (byte, short, int και long **για ακέραιους**, float και double **για πραγματικούς**, char **για χαρακτήρες** Unicode και **boolean** - true και false). Γενικά, **μια μεταβλητή ορίζει τη θέση μνήμης**, όπου απομνημονεύονται οι τιμές.

ii) Το όνομα μιας κλάσης ή ενός συστήματος επικοινωνίας (Interface), και

iii) Μια διάταξη (ένα array).

2. **Σταθερές** (Literal - κυριολεκτικά)

   Οι ακέραιες σταθερές στη γλώσσα μπορούν να εκφρασθούν σε οκταδικό (π.χ. O367) με το πρόθεμα **O** ή σε δεκαεξαδική μορφή με το πρόθεμα **Ox** ή **ox** (π.χ. Ox1A5)

Οι **πραγματικές σταθερές** είναι πάντοτε της μορφής double (64 bits), ενώ η χρήση του F ή f στο τέλος τους κάνει να είναι του τύπου float (32 bits), όπως στο παράδειγμα: 3.14f.

Τα λογικά literal είναι οι τιμές true και false.

Τέλος, τα literal του τύπου char είναι μεμονωμένοι χαρακτήρες που περικλείονται σε αποστρόφους, όπως τα 'a', '#' και '3', ενώ τα αλφαριθμητικά (string) είναι ένας συνδυασμός χαρακτήρων μέσα σε αποστρόφους.

3. Αριθμητικές πράξεις

Η JAVA διαθέτει πέντε (5) αριθμητικούς τελεστές, τους:

$+$ (**πρόσθεση**), $-$ (**αφαίρεση**), $*$ (**πολ/σμό**), $/$ (**διαίρεση**),

και % (**ακέραιο υπόλοιπο διαίρεσης ακεραίων** - **modulus**)

Σημείωση: Η γλώσσα χρησιμοποιεί το τελεστή + για τη συνένωση Αλφαριθμητικών (βλέπε τις εντολές εκτύπωσης: System.out.println(. . .))

4. Συγκρίσεις

Η JAVA διαθέτει τους ακόλουθους έξι (6) τελεστές σύγκρισης:

$==$ (**ισότητα**), $!=$ (**διάφορο**), $<$ (**μικρότερο**),

$<=$ (**μικρότερο ή ίσο**), $>$ (**μεγαλύτερο**),

$>=$ (**μεγαλύτερο ή ίσο**)

5. Τελεστές εκχώρησης τιμών

Ο βασικός τελεστής εκχώρησης τιμών είναι το « = », υπό την έννοια ότι εκτελούνται όλες οι πράξεις στο δεξιό μέλος του τελεστή και η τιμή που ευρίσκεται εκχωρείται στην μεταβλητή του αριστερού μέλους. Π.χ. στην εντολή:

$$y = y + 5/3,$$

εάν ο y έχει τιμή 5, τότε θα εκτελείται η διαίρεση ακεραίων στο δεξιό μέλος, που θα δώσει 1, το πηλίκο θα προστεθεί στο y, δηλαδή $5 + 1 = 6$, και το αποτέλεσμα θα εκχωρηθεί στην μεταβλητή y, που θα αντικαταστήσει την προηγούμενη τιμή 5.

Εξάλλου, η γλώσσα υιοθετεί και τις ακόλουθες συντομογραφικές εκφράσεις καταχωρήσεων, που πολύ συχνά εμφανίζονται στα προγράμματα:

$$x{+}{+} \quad \text{και} \quad x{-}{-}$$

που είναι συντομογραφίες των εντολών (αύξηση ή ελάττωση της τιμής κατά 1):

$$x = x + 1 \quad \text{και} \quad x = x - 1.$$

Σημείωση: Στις εντολές εκχώρησης:

$$(i)\ y = x++ \quad \text{και} \quad (ii)\ y = ++x$$

στην μεν (i) η y λαμβάνει την τιμή της x πριν την αύξηση, ενώ στην (ii) η y λαμβάνει την τιμή του x μετά την αύξηση.

Επίσης, η γλώσσα υιοθετεί και τις ακόλουθες συντομογραφίες:

$y+$	$=$	x	για την έκφραση	y	$=$	$y + x$
$y-$	$=$	x	για την έκφραση	y	$=$	$y - x$
$y*$	$=$	x	για την έκφραση	y	$=$	$y * x$
$y/$	$=$	x	για την έκφραση	y	$=$	y/x

6. Λογικές πράξεις

Η γλώσσα διαθέτει τους λογικούς τελεστές:

&&	**για τη λογική σύζευξη** - AND
||	**για τη λογική διάζευξη** - OR
!	**για τη λογική άρνηση** - NOT

7. Πράξεις με bits

Τέλος, η γλώσσα υποστηρίζει και τις ακόλουθες πράξεις με bits:

&	**Bitwise AND**,
|	**Bitwise OR**,
^	**Bitwise XOR**,
<<	**Αριστερή μετατόπιση**,
>>	**Δεξιά μετατόπιση**,
>>>	**Μετατόπιση με συμπλήρωση μηδενικών**.

8΄ Ιεραρχία τελεστών

Οι παραπάνω τελεστές ακολουθούν την ακόλουθη ιεραρχία:

i. $*, /, \%$

ii. $+, -$

iii. $<<, >>, >>>$

iv. $<, >, <=, >=$

v. $=, != $

vi. $\&$

vii. $\hat{}$

viii. **OR**

ix. $\&\&$

x. $||$

xi. $=, + =, - =, * =, / =, \% =, \hat{} =, \& =, | =, <<=, >>=, >>>=$

9. Εφαρμογές

(i) Το πρόγραμμα που ακολουθεί παρουσιάζει τα αποτελέσματα πράξεων με ακέραιους και πραγματικούς αριθμούς:

```
class Test1 {
public static void main (String args[]){
   short a=8;
   int b=2;
   float c=12.5f;
   float d=7f;
   System.out.println("a is"+a+",b is"+b);
   System.out.println("a+b="+(a+b));
   System.out.println("a-b="+(a-b));
   System.out.println("a*b="+(a*b));
   System.out.println("a/b="+(a/b));
   System.out.println("a%b="+(a%b));
   System.out.println("c is"+c+",d is"+d);
   System.out.println("c/d="+(c/d));
   }
}
```

Το printout του παραπάνω προγράμματος test1 θα είναι:

```
a is 8, b is 2
a + b = 10
a - b = 6
a / b = 4
a % b = 0
c is 12.5, d is 7
c/d = 1.78571
```

(ii) Το πρόγραμμα που ακολουθεί παρουσιάζει τη λειτουργία των συντομογραφειών **x++** και **++x**:

```
class Test2 {
public static void main (String args[]){
   int a=0;
   int b=0;
   System.out.println("a and b are"+a+"and"+b);
   a++;
   System.out.println("a++ results in"+a);
   ++a;
   System.out.println("++a results in"+a);
   System.out.println("Resetting a back to 0.");
   a=0;
   System.out.println("------");
   b = a++;
   System.out.println("b = a++ (postf(a) results in:");
   System.out.println("a is"+a);
   System.out.println("b is"+b);
   System.out.println("------");
   b = ++a;
   System.out.println("b = ++a (pref(a) results in:");
   System.out.println("a is"+a);
   System.out.println("b is"+b);
   System.out.println("------");
   }
}
```

Το printout του παραπάνω προγράμματος test2 θα είναι:

```
a and b are 0 and 0
a++ results in 1
++a results in 2
Resetting a back to 0
------
b = a++ (postf(a) results in:
a is 1
b is 0
------
b = ++a (pref(a) results in:
a is 2
b is 2
------
```

10. Διατάξεις - Συνθήκες - Βρόγχοι

Οι διατάξεις (arrays) στην java είναι το εργαλείο με το οποίο αποθηκεύουμε σύνολο στοιχείων με μια οντότητα. Μια διάταξη έχει **ένα αριθμό υποδοχών** ή **θέσεων (slots)** σε κάθε μια από τις οποίες αποθηκεύεται ένα στοιχείο (π.χ. οι συντεταγμένες ενός διανύσματος). Οι διατάξεις στην java είναι **πραγματικά αντικείμενα** και έτσι μπορούν να χρησιμοποιηθούν. Οι διατάξεις μπορούν να περιέχουν οποιοδήποτε τύπο τιμής στα στοιχεία τους, αλλά και αντικείμενα, όλα όμως τα στοιχεία τους πρέπει να είναι του ίδιου τύπου (π.χ. ακέραιος, ή αλφαριθμητικά, κλπ.).

Η δήλωση μιας διάταξης αποτελείται από ένα όνομα μεταβλητής που ακολουθείται από δύο ορθογώνιες παρενθέσεις που μπορούν να περιέχουν τον αριθμό των υποδοχών, **που πάντοτε αρχίζουν από το 0 έως τον αριθμό των υποδοχών -1**.

Σε κάθε αντικείμενο διάταξης που δημιουργείται με τον τελεστή **new**, όλες οι υποδοχές της διάταξης αρχικοποιούνται αυτόματα, **μηδέν (0)** για αριθμητικές διατάξεις, **false** για λογικές διατάξεις, η τιμή «**0**» για διατάξεις χαρακτήρων και η null για διάταξη αντικειμένων.

Τέλος, μπορεί κανείς να δημιουργήσει μιά διάταξη και να αρχικοποιήσει τις υποδοχές της ταυτόχρονα, όπως στο παράδειγμα:

$$String[\ ]\ names = \{"Kostas","Giannis","Mary","Anna"\},$$

ενώ το συγκεκριμένο στοιχείο μιας υποδοχής λαμβάνεται με την ένδειξη του δείκτη στο όνομα· π.χ. για τη διάταξη names, θα έχουμε:

$$names[0] = "Kostas",\ names[1] = "Giannis",\ \text{κλπ.}$$

Παράδειγμα:

Στο ακόλουθο πρόγραμμα δημιουργούνται διατάξεις, αρχικοποιούνται υποδοχές, τροποποιούνται στοιχεία που τελικά εκτυπώνονται.

```
class Test3 {
   String[] firstNames={"Dennis", "Grace", "Bjarne", "James"};
   String[] lastNames=new String[firstNames.length];
   void printNames (){
      int i=0;
      System.out.println(firstNames[i]+" "+lastNames[i]);
      i++;
      System.out.println(firstNames[i]+" "+lastNames[i]);
      i++;
      System.out.println(firstNames[i]+" "+lastNames[i]);
      i++;
      System.out.println(firstNames[i]+" "+lastNames[i]);
      i++;
   }
   public static void main(String args[]){
      Test3 a=new Test3();
      a.printNames();
      System.out.println("-------");
      a.lastNames[0]="Ritchie";
      a.lastNames[1]="Happer";
      a.lastNames[2]="Stroustrup";
      a.lastNames[3]="Gosling";
      a.printNames();
   }
}
```

Τα αποτελέσματα του προγράμματος θα είναι:

$Dennis$	$null$		
$Grace$	$null$		
$Bjarne$	$null$		
$James$	$null$		
$Dennis$	$Richie$	$\rightarrow$	(Επινοητής της γλώσσας C)
$Grace$	$Hopper$	$\rightarrow$	(Επινοητής της γλώσσας Cobol)
$Bjarne$	$Stroustrup$	$\rightarrow$	(Επινοητής της γλώσσας C++)
$James$	$Gosling$	$\rightarrow$	(Επινοητής της γλώσσας JAVA)

Παρατήρηση:

Τα πολυδιάστατα arrays μπορούν να οριστούν ως απλά arrays, που έχουν υποδοχές αντικείμενα arrays· π.χ.

int coords[ ] [ ] = newint[12][12];

coords[0][0]=1;

---

cordr[11][11]=144;

11. Συνθήκες

Η βασική έκφραση συνθήκης πραγματοποιείται με την εντολή **if**, που έχει την ακόλουθη δομή:
if (logical condition) {**Τμήμα Κώδικα**};
else {**Τμήμα Κώδικα**};

Παραδείγματα:

(i)

```
if (x < y)
   System.out.println("x is smaller than y");
else
   System.out.println("y is bigger than x");
```

(ii)

```
if (estate==true) System.out.println("Engine is on");
else System.out.println("Now engine is starting...")
    if (gasLevel >=1) state = true;
    else System.out.println ("Can't start engine! Low on gas.")}
```

(iii)

```
if (number%2 == 0) System.out.println("The Number is even!");
```

Μια άλλη ενδιαφέρουσα εντολή συνθήκης (ή συνθηκών) είναι η switch, με δομή:

```
switch (control){
    case value1:
        result1;
        break;
    case value2:
        result2;
        break;
.....................
    default: defaultresult;
}
```

όπου στο control υπάρχει είτε μια μεταβλητή είτε μια έκφραση που αποτιμάται σε τιμή ακέραια ή char και συγκρίνεται με κάθε μια από τις τιμές **case** (value1, value2, . . .) με την σειρά, έτσι ώστε εάν βρεθεί ταύτιση εκτελείται η η αντίστοιχη result, εάν όχι τότε εκτελείται η default, εάν και αυτή δεν υπάρχει τότε η εντολή switch αγνοείται και δεν εκτελείται.

Σημείωση: Στην java , η μεταβλητή του CONTROL μπορεί να είναι μόνο της μορφής των απλών πρωτογενών τύπων (όχι της μορφής long, float και double) και η μόνη σχέση που ελέγχεται είναι της ισότητας.

Παράδειγμα: Στο παρακάτω παράδειγμα η σταθερά oper είναι τύπου char.

```
switch(oper){
   case "+":
      addargs(arg1,arg2);
      break;
   case "-":
      subargs(arg1,arg2);
      break;
   case "*":
      mulargs(arg1,arg2);
      break;
   case "/":
      divargs(arg1,arg2);
      break;
   case "%":
      modargs(arg1,arg2);
      break;
   }
```

Σημείωση 1η: Μετά την case, στην switch και προ της break, μπορούν να παρεμβληθούν οποιεσδήποτε εντολές επιθυμούμε **χωρίς** άγκιστρα.

Σημείωση 2η: Η εντολή break σε κάθε case είναι απαραίτητη, άλλως ο έλεγχος δεν μεταβιβάζεται, αλλά συνεχίζεται η εκτέλεση των εντολών έως ότου βρεθεί τέλος της switch, το δεξιό άγκιστρο.

Στην συνέχεια ερχόμεθα **στις περιπτώσεις των βρόγχων**, της java , που έχουν την ακόλουθη δομή:

```
for (Initialization; Test; Increment){
   εντολές;
}
```

με:

Initialization, μια έκφραση που αρχικοποιεί τον βρόχο, π.χ. **int i=0**.

Test, έναν έλεγχο, που ενεργοποιείται πριν από κάθε πέρασμα του βρόχου, π.χ. **i<50**, και όταν **διαπιστώνεται αληθής** τότε **ο βρόγχος εκτελείται**, ενώ όταν διαπιστωθεί **το false**, τότε σταματάει η εκτέλεση του βρόγχου.

Increment, μία έκφραση τροποποίησης της μεταβλητής του βρόγχου, που προκαλεί την επαναληπτική υφή του όλου μηχανισμού - π.χ. i++.

Παράδειγμα: Στο παρακάτω παράδειγμα, αρχικοποιούνται όλες οι τιμές μιας αλφαριθμητικής διάταξης σε αλφαριθμητικά κενά (null strings):

```
String[ ] strArray[ ]=new String[10];
for (int i =0; i<strArray.length;i++)
   strArray[i] = "";
```

Ένας άλλος τρόπος πραγμάτωσης βρόγχου είναι η εντολή while, με την ακόλουθη δομή:

while (**συνθήκη**){
   εντολές
}

όπου συνθήκη είναι μια έκφραση ελέγχου, που εάν διαπιστωθεί ότι είναι αληθής, τότε εκτελούνται οι εντολές του βρόγχου και ελέγχεται η αρχική συνθήκη, πράγμα που επαναλαμβάνεται συνεχής έως ότου η συνθήκη διαπιστωθεί false, οπότε διακόπτεται η εκτέλεση των εντολών και ο έλεγχος του προγράμματος μεταβιβάζεται στην επόμενη εντολή.

Παράδειγμα: Το παρακάτω πρόγραμμα στοχεύει στην αντιγραφή των στοιχείων μιας διάταξης ακεραίων **array1[]**, σε διάταξη κινητής υποδιαστολής **array2[]**, κάνοντας την αντίστοιχη μετατροπή και εκτυπώνοντας τα στοιχεία.

```
public class CopyArray {
   public static void main (String args[]){
      int[]array1={8,7,6,5,4,3,2,1};
      float[]array2=new float[array1.length];
      System.out.print(" array1:[ ");
      for (int=0;i<array1.length;i++){
         System.out.print(array1[i]+" ");
         }
      System.out.println(" ]");
      System.out.print(" array2:[");
      int count=0;
      while(count<array1.length){
         array2[count]=float array1[count];
         System.out.print(array2[count++]+" ");
         }
      System.out.println("]");
   }
```

Το printout του προγράμματος θα είναι:

$array1$: [8 7 6 5 4 3 2 1]
$array2$: [8.0 7.0 6.0 5.0 4.0 3.0 2.0 1.0]

Εξάλλου ένα άλλο είδος βρόγχων **while** συνδεδεμένου με το **do** έχει την ακόλουθη δομή:

do {
  εντολές
} **while** (συνθήκη);

στην οποία οι εντολές εκτελούνται την πρωτη φορά και στη συνέχεια ελέγχεται η συνθήκη, εάν είναι αληθείς οι εντολές και πάλι εκτελούνται, πράγμα που συνεχίζεται έως ότου η συνθήκη διαπιστωθεί false.

Παράδειγμα: Στο παρακάτω πρόγραμμα εκτυπώνονται οι κύκλοι 1 έως 9 (round 1 $\cdots$ round 9):

```
public class Dowhilesample {
   public static void main (String args[]){
      int x=1;
      do {
         System.out.println(" Round:  "+x);
         x++;
      } while(x<10);
   }
}
```

Η έξοδος του προγράμματος είναι:

$$\begin{aligned} Round :: & \ 1 \\ Round :: & \ 2 \\ & \vdots \\ Round :: & \ 9 \end{aligned}$$

Τέλος, υπάρχει η δυνατότητα να έχουμε και **επώνυμους** (labeled) βρόγχους, που όταν αυτό συνδεθεί με τις εντολές break και continue μπορούμε να μεταφέρουμε τον έλεγχο της ροής του προγράμματος σε επιθυμητό σημείο. Στο παρακάτω πρόγραμμα εκτυπώνονται οι βρόγχοι που εκτελούνται.

Παράδειγμα:

```
public class Label_loop {
   public static void main (String args[]){
      label:
         for (int i=1; i<6; i++)
            for (int j=1; j<4; j++){
               System.out.println(" loop:  i is "+i+", j is "+j);
               if ((i+j)>4)
                  break label;
               }
      System.out.println(" end of loops ");
      }
   }
```

Το printout του προγράμματος είναι:

$$
\begin{aligned}
&loop: \ i \ is \ 1, \ j \ is \ 1\\
&loop: \ i \ is \ 1, \ j \ is \ 2\\
&loop: \ i \ is \ 1, \ j \ is \ 3\\
&loop: \ i \ is \ 2, \ j \ is \ 1\\
&loop: \ i \ is \ 2, \ j \ is \ 2\\
&loop: \ i \ is \ 2, \ j \ is \ 3\\
&end \ of \ loops
\end{aligned}
$$

ε. ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ (Objects and Classes)

Ο προγραμματισμός σε JAVA, αλλά και σε κάθε αντικειμενοστραφή γλώσσα, εδράζεται στη «φιλοσοφία» του lego! Δηλαδή, της δημιουργίας αντικειμένων από άλλα απλούστερα, όπως π.χ. από τα τουβλάκια κατασκευάζουμε τοίχους ή από τα διάφορα κομμάτια-εξαρτήματα ενός μοντέλου αυτοκινήτου το αυτοκίνητο, κτλ.

Ένα από τα βασικά χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού είναι οι **κλάσεις**. Όταν γράφεται ένα πρόγραμμα σε JAVA, **δεν ορίζονται πραγματικά αντικείμενα, ορίζονται κλάσεις αντικειμένων**, όπου **κάθε κλάση είναι ένα πρότυπο για πολλαπλά αντικείμενα με παρόμοια χαρακτηριστικά**. Έτσι, οι κλάσεις περιέχουν όλα τα χαρακτηριστικά μιας δεδομένης ομάδας αντικειμένων. Π.χ. μπορούμε να έχουμε μία κλάση Trees (δένδρα) στην οποία περιέχονται τα χαρακτηριστικά των δένδρων, όπως ύπαρξη ρίζας, φύλλων, παρουσία χλωροφύλλης, κλπ.

Φυσικά, δεδομένης της κλασης των δένδρων, μπορεί κανείς να δημιουργήσει διάφορα υποδείγματα (Instances) της κλάσεως, όπου κάθε υπόδειγμα μπορεί να έχει διαφορετικά χαρακτηριστικά, όπως υψηλό ή βραχύ, αειθαλές ή φυλλοβόλο, κλπ. Πάντως, σε οποιοδήποτε υπόδειγμα είναι δυνατόν να αναγνωρισθεί ως δένδρο. Από τα παραπάνω είναι σαφές ότι **ένα υπόδειγμα είναι η συγκεκριμένη, απολύτως σαφής, αναπαράσταση μιας κλάσης, ενώ τα υποδείγματα και τα αντικείμενα είναι το ίδιο πράγμα**.

Γενικότερα, μπορούμε να πούμε ότι όταν γράφουμε ένα πρόγραμμα σε JAVA, στην πραγματικότητα κατασκευάζουμε ένα σύνολο κλάσεων, ενώ όταν το πρόγραμμα «τρέχει» δημιουργούνται και καταστρέφονται υποδείγματα αυτών των κλάσεων όταν και όπως απαιτείται από τα δεδομένα που εισάγονται.

Το ενδιαφέρον είναι ότι **το περιβάλλον της γλώσσας** είναι εφοδιασμένο με ένα βασικό σύνολο κλάσεων (**την βιβλιοθήκη κλάσεων**), που υλοποιούν, σαν βασικό δομικό υλικό μεγάλο μέρος της βασικής συμπεριφοράς που χρειάζεται ο προγραμματιστής στη δομή του προγράμματος που κατασκευάζει, έτσι που αρκετές φορές το μόνο που χρειάζεται από τον προγραμματιστή να είναι η δημιουργία μιας κλάσης η οποία να χρησιμοποιεί τις βιβλιοθήκες κλάσεων.

Πιο συγκεκριμένα στο περιβάλλον της γλώσσας υπάρχουν:

1. Το πακέτο **java.lang**, που περιέχει τις κλάσεις και τα **interfaces** που συναποτελούν **τον πυρήνα της γλώσσας**.

2. Το πακέτο **java.util**, που περιέχει βοηθητικές κλάσεις και **interfaces** συμπεριλαμβανομένων κλάσεων για παραγωγή τυχαίων αριθμών, κλάσεων για ιδιότητες συστήματος, κλπ.

3. Το πακέτο java.io που διαθέτει κλάσεις εισόδου και εξόδου στοιχείων καθώς και interfaces **για αρχεία** και **streams**.

4. Το πακέτο **java.net**, που περιέχει κλάσεις και interfaces για την εκτέλεση λειτουργιών δικτύου, όπως π.χ. τα sockets και τα URLs.

5. Το πακέτο **java.awt**, που περιέχει κλάσεις και interfaces που συναποτελούν το πακέτο Abstract Windowing Toolkit (AWT).

6. Το πακέτο **java.applet**, που περιέχει την υποστήριξη που απαιτείται για τις γραφικές μίνι-εφαρμογές της JAVA, που τρέχουν μέσα στα όρια μιας Web σελίδας.