Διάλεξη 3: Προγραμματισμός σε JAVA I

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:
Εισαγωγή στις έννοιες:

- Στοιχειώδης Προγραμματισμός
- Προγραμματισμός με Συνθήκες
- Προγραμματισμός με Βρόγχους

Διδάσκων: Παναγιώτης Ανδρέου

Σχόλια

- Σχόλιο σε μία γραμμή με //

```
int x;              //Δήλωση μίας μεταβλητής (ακέραιος)

//Δήλωση μίας μεταβλητής (double)
double radius;
```

- Σχόλια σε πολλαπλές γραμμές

```
/*
 * Το πιο κάτω πρόγραμμα
 * βρίσκει το εμβαδό του κύκλου.
 */
double radius;
...
```

Προσδιοριστές (Identifiers)

- Ένας προσδιοριστής (identifier) είναι μία ακολουθία από χαρακτήρες οι οποίοι περιλαμβάνουν: γράμματα, αριθμούς, underscores (_), και σύμβολα δολαρίου ($).
- Ένας identifier πρέπει να ξεκινάει με ένα γράμμα, το underscore (_), ή το σύμβολο δολαρίου ($). Δεν μπορεί να ξεκινάει με ένα αριθμό.
- Ένας identifier δεν μπορεί να είναι μία δεσμευμένη λέξη (π.χ., if, for, while, class, κ.τ.λ.)
- Ένας identifier δεν μπορεί να είναι `true`, `false`, ή `null`.
- Το μέγεθος ενός identifier μπορεί να είναι όσο χρειαζόμαστε.

Δήλωση και Ανάθεση Μεταβλητών

Δήλωση Μεταβλητών

- `int x;` `//Δήλωση μίας μεταβλητής (ακέραιος)`
- `double radius;` `//Δήλωση μίας μεταβλητής (double)`
- `char a;` `//Δήλωση μίας μεταβλητής (χαρακτήρ.)`

Ανάθεση Μεταβλητών

- `x=1;` `// Ανάθεσε το 1 στο x`
- `radius= 1.0;` `// Ανάθεσε το 1.0 στο radius`
- `a='A';` `// Ανάθεσε το 'A' στο a`

Δήλωση και Ανάθεση σε ένα Βήμα

- `int x=1;`
- `double radius= 1.0;`

Σταθερές μεταβλητές

- Σταθερές (constant) Μεταβλητές: Μεταβλητές που δεν αλλάζουν κατά τη διάρκεια εκτέλεσης του προγράμματος
- Ορίζονται όπως τις υπόλοιπες μεταβλητές με τη λέξη final σαν πρόθεμα:

```
final datatype CONSTANTNAME = VALUE;
```

- Παραδείγματα

```
final double PI = 3.14159;
final int SIZE = 3;
```

Αριθμητικοί Τύποι Δεδομένων

Όνομα	Εύρος	Μέγεθος
byte	-2^7 (-128) to 2^7-1 (127)	8-bit signed
short	-2^{15} (-32768) to $2^{15}-1$ (32767)	16-bit signed
int	-2^{31} (-2147483648) to $2^{31}-1$ (2147483647)	32-bit signed
long	-2^{63} to $2^{63}-1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
float	Negative range: -3.4028235E+38 to -1.4E-45 Positive range: 1.4E-45 to 3.4028235E+38	32-bit IEEE 754
double	Negative range: -1.7976931348623157E+308 to -4.9E-324 Positive range: 4.9E-324 to 1.7976931348623157E+308	64-bit IEEE 754

ΠΡΟΣΟΧΗ: Τύποι floating-point δεν είναι αποθηκευμένοι με πλήρη ακρίβεια.

Παράδειγμα: System.out.println(1.0 - 0.9); ➔ 0.09999999999999998 και όχι 0.9

System.out.println(1.0 - 0.1- 0.1- 0.1- 0.1- 0.1); ➔ 0.5000000000000001 και όχι 0.5

Οι τελεστές τις 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.

Τελεστές Ανάθεσης (Συντομεύσεις)

Συντομεύσεως τελεστών ανάθεσης

- **+=** : π.χ., i += 8 ⇔ i = i + 8
- **-=** : π.χ., f -= 8.0 ⇔ f = f - 8.0
- ***=** : π.χ., i *= 8 ⇔ i = i * 8
- **/=** : π.χ., i /= 8 ⇔ i = i / 8
- **%=** : π.χ., i %= 8 ⇔ i = i % 8

Αυξητικοί και Μειωτικοί Τελεστές

- **++var:** αύξησε την var κατά 1 και μετά υπολόγισε
- **--var:** μείωσε την var κατά 1 και μετά υπολόγισε
- **var++:** υπολόγισε και μετά αύξησε την var κατά 1
- **var--:** υπολόγισε και μετά μείωσε την var κατά 1

```
public class test{
  public static void main(
      String[] args){
    int x=5;
    System.out.println(x++); //5
    System.out.println(x);   //6

    x=5;
    System.out.println(++x); //6
    System.out.println(x);   //6
  }
}
```

Μετατροπές Τύπων/Μεταβλητών

- Πάρτε για παράδειγμα τις ακόλουθες δηλώσεις:
  - `byte i = 100;` `int → byte`
  - `long k = i * 3 + 4;` `byte*int*int → long`
  - `double d = i * 3.1 + k / 2;` `byte*float+long/int → double`
- Όταν πραγματοποιείται μία πράξη μεταξύ μεταβλητών διαφορετικού τύπου, η JAVA μετατρέπει αυτόματα σύμφωνα με του ακόλουθους κανόνες:
  1. Αν μία από τις μεταβλητές είναι τύπου double, τότε και η άλλη μετατρέπεται σε double.
  2. Αλλιώς, Αν μία από τις μεταβλητές είναι τύπου float, τότε και η άλλη μετατρέπεται σε float.
  3. Αλλιώς, Αν μία από τις μεταβλητές είναι τύπου long, τότε και η άλλη μετατρέπεται σε long.
  4. Αλλιώς, Αν και οι δύο μεταβλητές μετατρέπονται σε int.

Μετατροπές Τύπων/Μεταβλητών (συν.)

- **Έμμεση Μετατροπή (implicit casting)**
  - Παράδειγμα 1
    - double d = 3;
    - Διεύρυνση/Μεγέθυνση τύπου int → double
    - Κανένα πρόβλημα στη μεταγλώττιση
  - Παράδειγμα 2
    - int i = 3.0;
    - Πρόβλημα στην μεταγλώττιση double → int
    - Συμβαίνει όταν γίνεται ανάθεση ενός πιο μικρού τύπου σε μεγαλύτερο (συρρίκνωση)
- **Άμεση Μετατροπή (explicit casting)**
  - int i = (int) 3.0;
  - Κανένα πρόβλημα στη μεταγλώττιση
  - Συρρίκνωση δεδομένων
  - Μπορεί να γίνει αφαίρεση κλάσματος/δεκαδικών, π.χ., int i = (int) 3.9; → i=3.
- Τι θα γίνει με το ακόλουθο; int x = 5 / 2.0; →→int x = (int) (5 / 2.0);

Strings & Characters

Χαρακτήρες (char)

- `char c = 'A';  //Δήλωση μίας μεταβλητής (char)`
- ΠΡΟΣΟΧΗ: Οι τελεστές αύξησης/μείωσης (++/--) μπορούν να χρησιμοποιηθούν και σε χαρακτήρες. Π.χ., ++c = 'B'.
- Μετατροπές τύπων μεταξύ ακέραιου (int) και χαρακτήρα (char)
  - int i = 'a'; Ισοδύναμο με int i = (int) 'a';
  - char c = 97; Ισοδύναμο με char c = (char) 97;

Συμβολοσειρές (String)

- `String msg = "Hello World";  //Δήλωση μίας μεταβλητής (string)`
- ΠΡΟΣΟΧΗ: δεν είναι αρχέγονος τύπος (String = java.lang.String)

Προγραμματισμός με Συνθήκες

- Πολλές φορές χρειάζεται να ελέγξουμε τα δεδομένα μίας μεταβλητής και ανάλογα να προβούμε σε διαφορετικές ενέργειες
- Το αποτέλεσμα μίας σύγκρισης, π.χ., x<5, είναι μία μεταβλητή τύπου boolean με τιμές true ή false.
- **Τελεστές Σύγκρισης**
  - < μικρότερο
  - <= μικρότερο ή ίσο
  - > μεγαλύτερο
  - >= μεγαλύτερο ή ίσο
  - == ίσο
  - != όχι ίσο
- Δύο είδη δηλώσεων με συνθήκες
  - if … else , if … else if … else
  - switch … case

Λογικοί Τελεστές

- **!** Όχι (not)
- **&&** Και (and)
- **||** Ή (or)
- **^** exclusive or

p	!p	Example (assume age = 24, gender = 'M')
true	false	!(age > 18) is false, because (age > 18) is true.
false	true	!(gender != 'F') is true, because (grade != 'F') is false.

p1	p2	p1 && p2	Example (assume age = 24, gender = 'F')
false	false	false	(age > 18) && (gender == 'F') is true, because (age > 18) and (gender == 'F') are both true.
false	true	false	
true	false	false	(age > 18) && (gender != 'F') is false, because (gender != 'F') is false.
true	true	true	

p1	p2	p1 || p2	Example (assume age = 24, gender = 'F')
false	false	false	(age > 34) || (gender == 'F') is true, because (gender == 'F') is true.
false	true	true	
true	false	true	(age > 34) || (gender == 'M') is false, because (age > 34) and (gender == 'M') are both false.
true	true	true	

p1	p2	p1 ^ p2	Example (assume age = 24, gender = 'F')
false	false	false	(age > 34) ^ (gender == 'F') is true, because (age > 34) is false but (gender == 'F') is true.
false	true	true	
true	false	true	(age > 34) || (gender == 'M') is false, because (age > 34) and (gender == 'M') are both false.
true	true	false	

Τελεστές Δυαδικών Ψηφίων (Bitwise Operators)

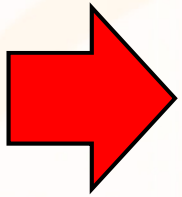
- Επιτρέπουν την επεξεργασία των δυαδικών ψηφίων μεταβλητών αρχέγονου **ακέραιου** τύπου (int, long, short, byte, char).
- Εκτελούν πράξεις άλγεβρας bool στα αντίστοιχα bits των κατηγορημάτων τους και δίνουν το αποτέλεσμα.
- Τελεστές: **AND(&), OR (|), XOR(^), NOT(~).**
- Επίσης: &=, |=, ^= (δεν επιτρέπεται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

Απλό if … else

- Θυμηθείτε το πρόγραμμα για τη δημιουργία κύκλων με ακτίνα
- **Πρόβλημα:** Τι γίνεται αν δώσουμε αρνητική τιμή για την ακτίνα του κύκλου; ➔ λάθος αποτέλεσμα
- **Λύση:** Μόνο αν ο χρήστης δώσει θετική τιμή τότε αλλάζουμε την τιμή της radius
- Σύνταξη απλού if: **if (** <boolean expression> **) { … } else { … }**

```
class Circle {
  //Η ακτίνα αυτού του κύκλου
  double radius = 1.0;

  //Δημιούργησε ένα αντικείμενο τύπου κύκλος
  Circle () {
  };

  //Δημιούργησε ένα αντικείμενο τύπου κύκλος
  //με συγκεκριμένη ακτίνα
  Circle (double newRadius) {
    radius = newRadius;
  };

  //Επέστρεψε το εμβαδό αυτού του κύκλου
  double getArea () {
    return radius * radius * π;
  }

}
```

```
class Circle {
  double radius = 1.0;
  …
  Circle (double newRadius) {
    radius = newRadius;
  };
  …
}
```



```
class Circle {
  double radius = 1.0;
  …
  Circle (double newRadius) {
    if ( newRadius >= 0 ){
      radius = newRadius;
    }
    else {
      radius = 0;
    }
…
```

if … else if … else

- Τι συμβαίνει αν υπάρχουν πολλές συνθήκες για την ίδια μεταβλητή;
- **Παράδειγμα:** Υπάρχουν οι ακόλουθοι φόροι σύμφωνα με τον μισθό κάποιου ατόμου:

  μισθός <=20,000 → 0% φόρος
  μισθός >20,000 και <=30,000 → 20% φόρος
  μισθός >30,000 και <=40,000 → 30% φόρος
  μισθός >40,000 → 40% φόρος

- **Πρόβλημα:** Χρειάζεται να γράψουμε 3x if statements; → **ΟΧΙ**
- **Λύση:** Χρήση του if … else if … else

```
computeTax (double salary) {
  double tax = 0.0;
  if ( salary <= 20000 ) { tax = 0.0; }
  else if ( salary > 20000 && salary <= 30000) { tax = 0.2; }
  else if ( salary > 30000 && salary <= 40000) { tax = 0.3; }
  else { tax = 0.4; }
  return tax * salary;
}
```

Λογικός Τελεστής ΚΑΙ (&&)
(στις επόμενες διαφάνειες)

Περισσότερα για if … else

```
boolean even;
if (number  % 2 == 0 ) {
  even = true;
else
  even = false;
```

Ισοδύναμο

⇔

```
boolean even =
 (number  % 2 == 0);
```

```
if ( even == true )
  …
```

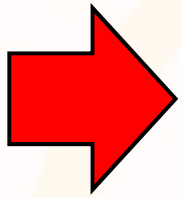
Ισοδύναμο

⇔

```
if ( even )
  …
```

- Το πρόγραμμα ελέγχου του δίσεκτου χρόνου σε μία γραμμή!

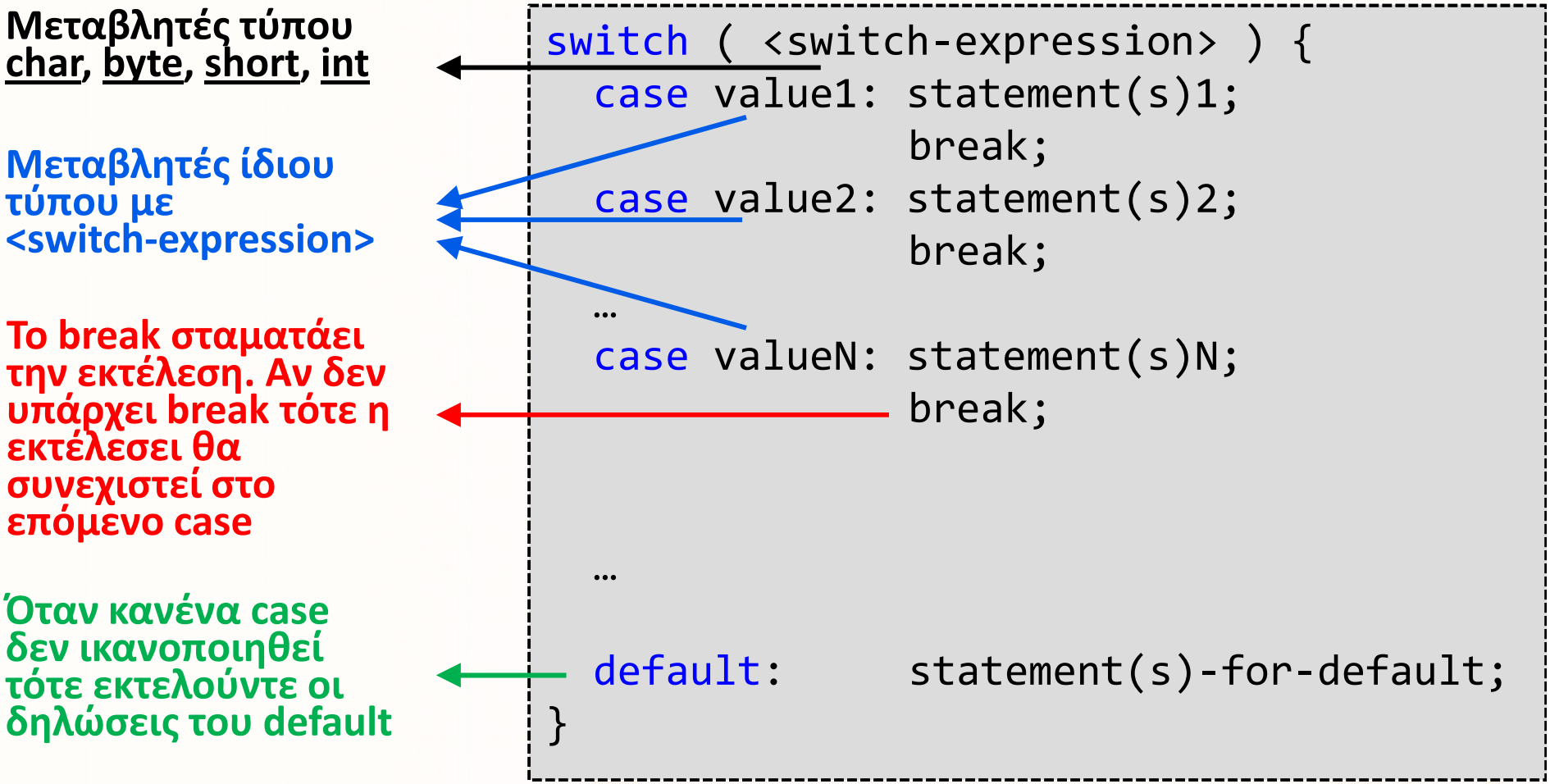
```
    if (year % 400)
      result = true;
    elseif (year % 100)
      result = false;
    elseif(year % 4)
      result = true;
    result = false;
```



```
if ( (year % 4 == 0 &&
      year % 100 != 0)
     ||
     (year % 400 == 0) )
```

switch … case

- Το switch ομαδοποιεί πολλά if
- Σύνταξη:



Ο τριαδικός τελεστής συνθήκης (ternary operator)

- Η ακόλουθη δήλωση

```
if (x > 0)
  y = 1;
else
  y = -1;
```

είναι ισοδύναμη με

```
y = (x > 0) ? 1 : -1;
```

- **Σύνταξη Τριαδικού Τελεστή: (<boolean-exp>) ? exp1 : exp2;**

- Παραδείγματα

```
System.out.println(     ( num % 2 == 0 ) ?  // (boolean-exp)
                        num + "is even"  :  // exp1
                        num + "is odd");    // exp2
```

Προτεραιότητα τελεστών

1. var++, var--
2. +, - (Unary plus and minus), ++var,--var
3. (type) Casting
4. ! (Not)
5. *, /, % (Multiplication, division, and remainder)
6. +, - (Binary addition and subtraction)
7. <, <=, >, >= (Comparison)
8. ==, !=; (Equality)
9. ^ (Exclusive OR)
10. && (Conditional AND) Short-circuit AND
11. || (Conditional OR) Short-circuit OR
12. =, +=, -=, *=, /=, %= (Assignment operator)

Προγραμματισμός με βρόγχους (loops)

3 είδη δηλώσεων προγραμματισμού με βρόγχους

- **while**

```
while ( <boolean expression> ) {
  //δηλώσεις
 }
```

- **do … while**

```
do {
  //δηλώσεις
} while ( <boolean expression> );
```

- **for**

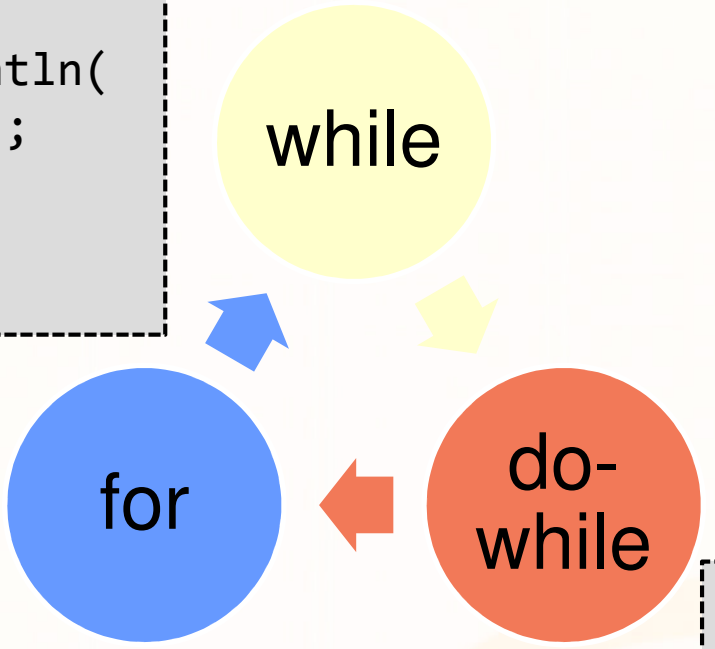
```
for( <initial actions>; <boolean expr.>; actions after step )
  //δηλώσεις
}
```

- **Όλα τα είδη βρόγχων είναι ισοδύναμα**.

Παραδείγματα προγραμματισμού με βρόγχους

Παράδειγμα <while>

```
int i = 0;
while (i < 10) {
  System.out.println(
          "while");
  i ++;
}
```



Παράδειγμα <do-while>

```
int i = 0;
do {
  System.out.println(
          "do-while");
  i++;
} while (i < 10);
```

Παράδειγμα <for>

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

Δηλώσεις Branching: break, continue και return

- **Break**: Σταματάει την εκτέλεση **<u>του πιο κοντινού βρόγχου</u>**
  - **<u>Ερώτηση:</u>** Πως σταματάμε την εκτέλεση φωλιασμένων βρόγχων;
  - **<u>Απάντηση:</u>** Με τη χρήση break και label

- **Continue:** Συνεχίζει με την επόμενη εκτέλεση του πιο κοντινού βρόγχου (οι δηλώσεις κάτω από το continue δεν εκτελούνται
  - **<u>Ερώτηση:</u>** Πως συνεχίζουμε την επόμενη εκτέλεση κάποιου πιο μακρινού βρόγχου;
  - **<u>Απάντηση:</u>** Με τη χρήση continue και label

- **Return:** Σταματάει την εκτέλεση της συγκεκριμένης μεθόδου.

Παραδείγματα: break, continue και return

```
for (i = 0; i < SIZE; i++) {
  …
  break;
  …
}
…
```

```
for (i = 0; i < SIZE; i++) {
  …
  continue;
  …
}
…
```

```
search1:   //search1 is a label
for (i = 0; i < SIZE; i++) {
  for (j = 0; j < SIZE; j++) {
    …
    break search1;
    …
  }
  …
}
…
```

```
search1:   //search1 is a label
for (i = 0; i < SIZE; i++) {
  for (j = 0; j < SIZE; j++) {
    …
    continue search1;
    …
  }
  …
}
…
```

```
someMethod() {
    … return; … }
```

→ **Exit someMethod()**