Variables in Java

Variable scope

- Οι μεταβλητές στη Java χαρακτηρίζονται και από την «εμβέλειά» τους (scope)
- Βάσει του scope έχουμε τις εξής 4 κατηγορίες μεταβλητών:
  - Local variables (τοπικές μεταβλητές εντός μεθόδων)
  - Method parameters (οι μεταβλητές που χρησιμοποιούνται ως παράμετροι στις μεθόδους)
  - Instance Variables (Global μεταβλητές που ανήκουν σε κάθε αντικείμενο ξεχωριστά. Οι συγκεκριμένες αναφέρονται και ως τα «πεδία» ή τα «χαρακτηριστικά» του αντικειμένου)
  - Class Variables (Global μεταβλητές που ανήκουν στην τάξη. Συγκεκριμένα οι static variables)

Local variables – Method parameters

- Έχουν το μικρότερο «χρόνο ζωής» αφού ορίζονται εντός μεθόδων και πολλές φορές σε τμήματα μεθόδων (π.χ μέσα σε ένα loop)
- Πάντα κοιτάζουμε το block ({ }) μέσα στο οποίο ορίζονται
- Φυσικά δεν υπάρχει πρόσβαση σε αυτές έξω από τη μέθοδο, ούτε και έξω από το block

```
class Student {
    private double marks1, marks2, marks3;
    private double maxMarks = 100;
    public double getAverage() {
        double avg = 0;
        avg = ((marks1 + marks2 + marks3) / (maxMarks*3)) * 100;
        return avg;
    }
    public void setAverage(double val) {
        avg = val;
    }
}
```

Instance variables

Local variable avg

This code won't compile because avg is inaccessible outside the method getAverage.

Instance variables

- Εντός της τάξης, έξω από μεθόδους
- Χωρίς τη χρήση του keyword: static
- Σε αυτές έχουν πρόσβαση όλες οι μη στατικές μέθοδοι
- Κάθε αντικείμενο έχει τις δικές του

```
class Phone {
    private boolean tested;
    public void setTested(boolean val) {
        tested = val;
    }
    public boolean isTested() {
        return tested;
    }
}
```

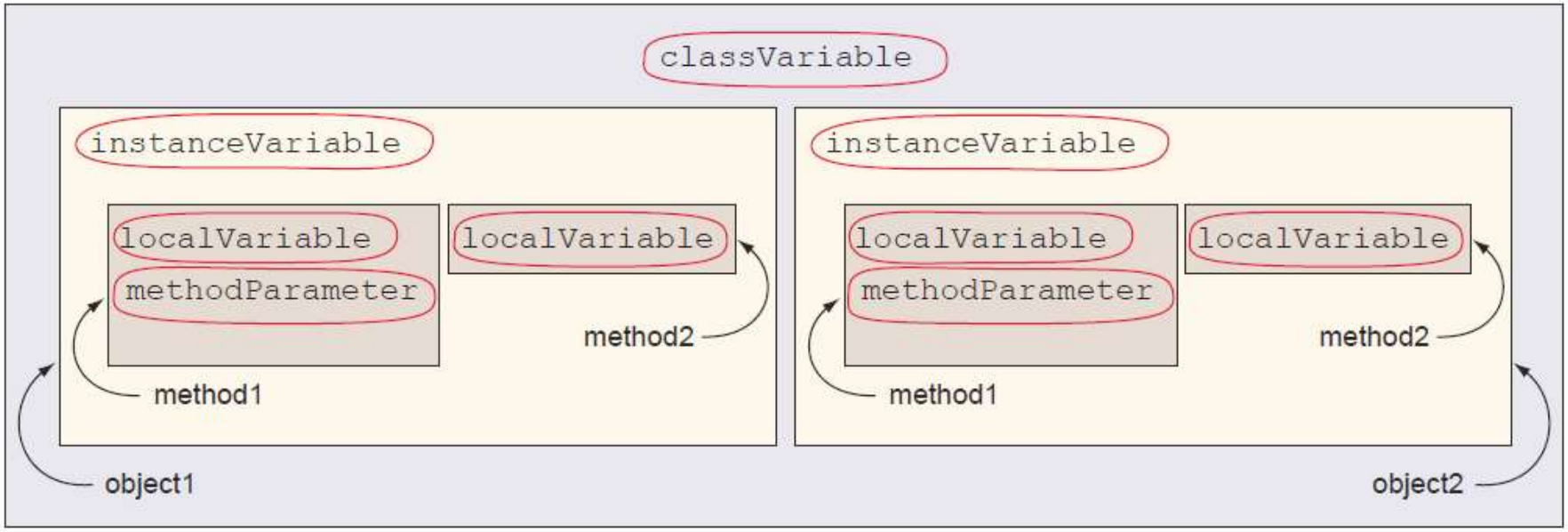
Instance variable tested

Variable tested is accessible in method setTested

Variable tested is also accessible in method isTested

Class variables

- Εντός της τάξης, έξω από μεθόδους
- Με τη χρήση του keyword: static
- Σε αυτές έχουν πρόσβαση όλες οι μη μέθοδοι, στατικές και μη (σκεφτείτε γιατί!)
- Ανήκουν στην τάξη, οπότε όλα τα αντικείμενα έχουν πρόσβαση στις ίδιες static variables. Δεν μπορεί ένα αντικείμενο να έχει τις δικές του


classVariable
instanceVariable
localVariable
methodParameter
localVariable
method2
method1
object1
instanceVariable
localVariable
methodParameter
localVariable
method2
method1
object2

Ίδια ονόματα μεταβλητών

- Δε μπορούμε να έχουμε instance και static variable με το ίδιο όνομα
- Δε μπορούμε να έχουμε παράμετρο μεθόδου και local variable με το ίδιο όνομα
- Μπορούμε να έχουμε είτε instance και local variable με το ίδιο όνομα, είτε static και local variable με το ίδιο όνομα. Όμως θέλει πολύ μεγάλη προσοχή!

Συμπληρωματικές γνώσεις 1

Method return statement

- Είναι υποχρεωτικό σε μεθόδους που «επιστρέφουν» τιμές
- Δεν είναι υποχρεωτικό σε μεθόδους που επιστρέφουν void
- Πολλές φορές χρησιμοποιείται μέσα σε συνθήκη ελέγχου για τον τερματισμό της μεθόδου (δείτε το οπωσδήποτε με παράδειγμα, καθώς και μέσα σε loop)

- For a method that returns a value, the `return` statement must be followed immediately by a value.
- For a method that doesn't return a value (return type is `void`), the `return` statement must *not* be followed by a return value.
- If the compiler determines that a `return` statement isn't the last statement to *execute* in a method, the method will fail to compile.

Variable Arguments (varargs)

- Java 5+
- Είναι ουσιαστικά πίνακες
- Μας επιτρέπουν να «στείλουμε» σε μια μέθοδο ένα οποιονδήποτε αριθμό από ορίσματα, του ιδίου τύπου
- Μια μέθοδος μπορεί να έχει έως μια παράμετρο varargs
- Αν υπάρχει παράμετρος varargs τότε πρέπει να είναι η τελευταία παράμετρος

```
public static void main(String[] args) {
    varargTest( ...args: "one","two","three");
}
public static void varargTest(String... args){
    for (int i=0;i<args.length;i++)
        System.out.println(args[i]);
}
```