Ενότητα 1: Μάθημα 8

Διάλεξη

Παράμετροι, Ορίσματα και Τοπικές Μεταβλητές
Κλήση με Τιμή (call by value)
Η παράμετρος "this"
Ενδοσκόπηση κλάσεων

University of Cyprus
Department of Computer Science

Τυπικές Παράμετροι (formal parameters)

- Οι **τυπικές παράμετροι** (**formal parameters**) μιας μεθόδου καθορίζουν τις πληροφορίες που πρέπει να περάσουμε στην μέθοδο όταν την καλέσουμε.

```
public class Charge {
    private double rx, ry;   // position
    private double q;        // charge

    public Charge(double x0, double y0, double q0) {
        rx = x0;
        ry = y0;
        q  = q0;
    }
    public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }

    public String toString() {
        return q + " at " + "(" + rx + ", " + ry + ")";
    } }
```

Τυπικές παράμετροι
(formal parameters)

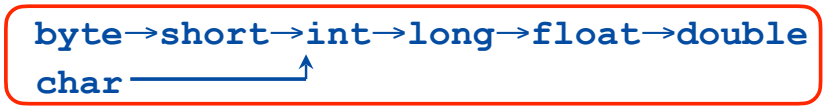
Πραγματικές Παράμετροι ή Ορίσματα

- Κατά την **κλήση μεθόδου**, στη θέση της κάθε τυπικής παραμέτρου εισάγουμε ένα ***όρισμα*** (argument), το οποίο μπορεί να είναι:
  - Χειριστήριο αντικειμένου του ίδιου τύπου με την τυπική παράμετρο
  - Μεταβλητή ή Τιμή αρχέγονου/πρωταρχικού τύπου (primitive value), συμβατού τύπου με την τυπική παράμετρο
- Τα ορίσματα αποκαλούνται επίσης πραγματικές παράμετροι (actual parameters).
- Ο αριθμός και η σειρά των ορισμάτων *πρέπει να ταιριάζει ακριβώς σε αυτά της λίστας παραμέτρων*
- Ο τύπος κάθε ορίσματος πρέπει να είναι **συμβατός** (*compatible*) με τον τύπο της αντίστοιχης παραμέτρου

```
int a=1,b=2,c=3;
double result = myMethod(a,b,c);
```


Συμβατότητα και μετασχηματισμός (αρχέγονων) τύπων

- Εάν οι τύποι ορισμάτων και παραμέτρων δεν ταιριάζουν ακριβώς, η Java θα προσπαθήσει να πραγματοποιήσει αυτόματο μετασχηματισμό τύπου (**automatic type conversion**)
- Ένα αρχέγονο όρισμα μπορεί να υποστεί αυτόματα μετασχηματισμό τύπου (**type casting**) από οποιονδήποτε από τους παρακάτω τύπους, σε οποιονδήποτε από τους τύπους που εμφανίζονται στα δεξιά:



Τυπικές παράμετροι
(formal parameters)

```
public class Charge {
    private double rx, ry;    // position
    private double q;         // charge

    public Charge(double x0, double y0, double q0) {
        rx = x0;
        ry = y0;
        q  = q0;
    }
    public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }

    public String toString() {
        return q + " at " + "(" + rx + ", " + ry + ")";
    } }
```

Ορίσματα ή **πραγματικές παράμετροι**
(arguments ή actual parameters)

Χρήση όρων «Παράμετρος» και «Όρισμα»: Προσοχή!

- Παράμετρος (parameter) είναι η **μεταβλητή**
- Το όρισμα (argument) είναι η **τιμή** που δίνεται στην παράμετρο
- Οι όροι **παράμετρος** και **όρισμα** συχνά χρησιμοποιούνται εναλλακτικά
- Όταν τους συναντήσετε, ίσως χρειαστεί να προσδιορίσετε την ακριβή τους σημασία από τα συμφραζόμενα.

Κλήση με τιμή (call by value)

```
int a=1,b=2,c=3;
double result = myMethod(a,b,c);
```

- Στο πιο πάνω παράδειγμα, η τιμή του κάθε ορίσματος και όχι το όνομα της μεταβλητής, περνιέται στην αντίστοιχη παράμετρο μεθόδου.
- Αυτός ο τρόπος σύνδεσης ορισμάτων με τυπικές παραμέτρους είναι γνωστός ως μηχανισμός *κλήσης με τιμή* (*call-by-value mechanism*)

Τοπικές μεταβλητές (local variables)

```
public class ChargeClient {

   public static void main(String[] args) {
          double x = Double.parseDouble(args[0]);
          double y = Double.parseDouble(args[1]);

          Charge c1 = new Charge(.51, .63, 21.3);
          Charge c2 = new Charge(.13, .94, 81.9);
          System.out.println(c1);
          System.out.println(c2);

          double v1 = c1.potentialAt(x, y);
          double v2 = c2.potentialAt(x, y);
          System.out.println(v1+v2);
      }
   }
```

Μια μεταβλητή που δηλώνεται στο **εσωτερικό μεθόδου** αποκαλείται *τοπική μεταβλητή*

Οι παράμετροι μεθόδων χρησιμοποιούνται σαν να ήταν τοπικές μεταβλητές.

```
public class Charge {
    private double rx, ry;    // position
    private double q;         // charge

    public Charge(double x0, double y0, double q0) {
        rx = x0;
        ry = y0;
        q  = q0;
    }
    public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }

    public String toString() {
        return q + " at " + "(" + rx + ", " + ry + ")";
    } }
```


Παράμετροι μεθόδων κ. Τοπικές Μεταβλητές

- Οι τυπικές παράμετροι συχνά θεωρούνται ως σύμβολα αναπλήρωσης (placeholders) που αντικαθίστανται από τις τιμές των αντίστοιχων ορισμάτων τους.
- Ωστόσο, στην πραγματικότητα οι παράμετροι είναι κάτι περισσότερο: έχουν θέση και χρήση τοπικών μεταβλητών.
- Όταν κληθεί μια μέθοδος, υπολογίζονται οι τιμές των ορισμάτων της και οι τιμές αυτές χρησιμοποιούνται για να αρχικοποιηθούν οι αντίστοιχες παράμετροι.
- Ακόμα κι αν η τιμή μιας τυπικής παραμέτρου αλλάξει μέσα στο σώμα μιας μεθόδου, **η τιμή του ορίσματος δεν μπορεί να αλλάξει.**

Call-by-value semantics

Τι συμβαίνει κατά την εκτέλεση της

```
public class Silly {
 public double dummySum(double x, double y, int z) {
     z = 288;
     return x + y;
 }}
```

```
public class TestSilly {
 public static void main(String args[]){
   int zop = 133;
   Silly sillyObj = new Silly();
   System.out.println(sillyObj.dummySum(1, 2, zop) + " "+ zop);
}}
```

- **Εκτυπώνει: 0 0**
- **Εκτυπώνει: 3.0 zop**
- **Εκτυπώνει: 3.0 133**
- **Εκτυπώνει: 3.0 288**

Ενότητα 1: Μάθημα 8

Η χρήση του «this»

University of Cyprus
Department of Computer Science

```
public class Charge {
    private double rx, ry;    // position
    private double q;         // charge

    public Charge(double x0, double y0, double q0) {
        rx = x0;
        ry = y0;
        q  = q0;
    }
    public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }

    public String toString() {
        return q + " at " + "(" + rx + ", " + ry + ")";
    }
}
```


Η παράμετρος this

```
public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }
```

- Σε κάθε χρήση μεταβλητής στιγμιοτύπου (instance variable) υπονοείται ότι ενυπάρχει παραπομπή στο αντικείμενο που περιέχει τη μεταβλητή:

  `<object handle>.identifier`

- Αν χρειαστούμε να αναφερθούμε ρητά στο περικλείον αντικείμενο, μπορούμε να χρησιμοποιήσουμε την κλειδολέξη (keyword) **this**

Η παράμετρος this

- **myInstanceVariable** always means and is always interchangeable with **this.myInstanceVariable**

```
public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }
```

```
public double potentialAt(double x, double y) {
        double k = 8.99e09;
        double dx = x - this.rx;
        double dy = y - this.ry;
        return k * this.q / Math.sqrt(dx*dx + dy*dy);
    }
```


Η παράμετρος this

- Η **this πρέπει** να χρησιμοποιείται **υποχρεωτικά**, εάν στο σώμα της μεθόδου ορίζεται παράμετρος ή τοπική μεταβλητή με το ίδιο όνομα με μια μεταβλητή στιγμιοτύπου της κλάσης
  - Διαφορετικά, όλες οι εμφανίσεις του ονόματος της μεταβλητής θα ερμηνεύονται ως τοπικές

```
int someVariable = this.someVariable
```

local — instance variable

Η παράμετρος this

```
class Banana {
  double param;

  Banana(int prm) {
    param = prm;
  }

  void f(int i) {
    System.out.println("Calc: " + i * param);
  }
}


Banana a = new Banana(5), b = new Banana(7);
a.f(1);
b.f(2);
```


Η χρήση του this (συνέχεια)

- Πως μπορεί η μέθοδος `f()` να γνωρίζει αν καλείται από το αντικείμενο **a** ή το αντικείμενο **b**;
  - `a.f(1)` ⇔ `Banana.f(a,1)`
  - `b.f(2)` ⇔ `Banana.f(b,2)`
- Η παράμετρος **this** είναι «κρυμμένη»
- Αν και δεν εμφανίζεται στη λίστα τυπικών παραμέτρων μιας μεθόδου, αυτομάτως και υπόρρητα περνιέται σαν όρισμα στο σώμα της μεθόδου:
  - Όταν κληθεί μια μέθοδος σε ένα αντικείμενο, το σώμα της μεθόδου συνδέεται αυτόματα με τη `this`, η οποία παραπέμπει στο αντικείμενο της κλήσης.

Σύνοψη: η παράμετρος this

- Αν θέλουμε, μέσα από το σώμα κάποιου αντικειμένου, να αποκτήσουμε πρόσβαση-χειριστήριο προς το ίδιο το αντικείμενο, μπορούμε να χρησιμοποιήσουμε την ειδική μεταβλητή **this**, η οποία είναι χειριστήριο για το αντικείμενο μας.
- Ανάθεση στην **this** δεν επιτρέπεται.
- Μέσω της **this** μπορούμε να περάσουμε το τρέχον αντικείμενο σαν παράμετρο σε μεθόδους άλλων αντικειμένων.

Παράδειγμα χρήσης this

```
// Simple use of the "this" keyword.
public class Leaf {
  private int i = 0;
  Leaf increment()
   i++;
   return this;
  }
  void print() { System.out.println("i = " + i); }
  public static void main(String[] args) {
   Leaf x = new Leaf();
   x.increment().increment().increment().print();
  }
}
```

- **Πολλαπλή κλήση της ίδιας μεθόδου πάνω στο ίδιο αντικείμενο.**