

## I sottoprogrammi (o metodi) - I

Stefano Mizzaro

Dipartimento di matematica e informatica  
Università di Udine  
<http://www.dimi.uniud.it/mizzaro/>  
mizzaro@dimi.uniud.it  
Programmazione, lezione 9  
23 ottobre 2007

## Dove siamo

- Mattoni
- Programmazione strutturata
- Sviluppo incrementale
- Array

Stefano Mizzaro - Metodi I

2

## Oggi

- I metodi
  - (sottoprogrammi, routine, subroutine, ...)
- Intro
- Esempi
- Definizioni
- (Prova/Traccia di esecuzione)

Stefano Mizzaro - Metodi I

3

## Metodo

- "Parte di programma con un nome"
- Metodi
  - Già definiti
    - `Math.sqrt`, `Math.sin`, `System.out.println`, ...
    - `double x = Math.sin(Math.PI/2)`
    - Altre istruzioni oltre a dichiarazioni, assegnamento e strutture di controllo programmazione strutturata
  - Definiti dal programmatore
    - `stampaDatiStudiante`, `moltiplicaMatrici`, ...
    - "Estendere il Java con nuove istruzioni"

Stefano Mizzaro - Metodi I

4

## Esempio

```
static double tangente (double alfa) {
    return (Math.sin(alfa) / Math.cos(alfa));
}
...
double x, y;
...
x = tangente(y);
```

- Definizione e uso
  - (Java ha `Math.tan`, ma facciamo finta di no...)
- Parametri
  - formali (`alfa`) e attuali (`y`)

Stefano Mizzaro - Metodi I

5

## Esempio completo (1/3)

```
class EsempioDiMetodo {
    static double cotangente (double x) {
        return (Math.cos(x) / Math.sin(x));
    }

    public static void main (String[] args) {
        double angolo, res;

        angolo = Leggi.unDouble();
        res = cotangente(angolo);
        System.out.println("La cotangente di "+
            angolo+" e': "+res);
    }
}
```

Stefano Mizzaro - Metodi I

6

## Esempio completo (2/3)

```
class EsempioDiMetodo {
    static double cotangente (double x) {
        return (Math.cos(x) / Math.sin(x));
    }

    public static void main (String[] args) {
        double angolo;

        angolo = Leggi.unDouble();
        System.out.println("La cotangente di "+
            angolo+" e': "+
            cotangente(angolo));
    }
}
```

Stefano Mizzaro - Metodi I

7

## Esempio completo (3/3)

```
class EsempioDiMetodo {
    static double tangente (double x) {
        return (Math.sin(x) / Math.cos(x));
    }

    static double cotangente (double x) {
        return (1 / tangente(x));
    }

    public static void main (String[] args) {
        double angolo;

        angolo = Leggi.unDouble();
        System.out.println("La cotangente di "+
            angolo+" e': "+
            cotangente(angolo));
    }
}
```

## Due tipi di metodi

- Funzioni
  - Restituiscono un risultato
  - Es. `tangente`, `Leggi.unInt`, ...
- Procedure
  - "Fanno qualcosa", hanno un effetto
  - Es. `System.out.println`, `stampaMatrice`, `ordinaArray`, `azzerArray`, ...
- O misti...
  - `ordinaArray`, `azzerArray`, ...

Stefano Mizzaro - Metodi I

9

## Definizione metodo

- Al di fuori del `main` (e dentro il `class`)
- Intestazione (prima riga)
  - `static` (per ora, ne riparleremo)
  - Tipo del risultato restituito (`void` se procedura)
  - Nome del metodo
  - Parametri formali e loro tipi (separati da ",")
- Corpo (fra graffe - obbligatorie)
  - Cosa fa e come lo fa
  - `return` termina esecuzione (e restituisce valore)

Stefano Mizzaro - Metodi I

10

## Uso del metodo

- Decidere i parametri attuali
  - Gli argomenti
- Invocarlo con il nome
- (`Leggi.java`
  - Definisce metodi che poi noi usiamo)

Stefano Mizzaro - Metodi I

11

## Struttura programma Java

```
class ... {
    static <tipo> <id> (<parametri>) {
        ...
    }

    static <tipo> <id> (<parametri>) {
        ...
    }

    public static void main (String[] args) {
        ...
    }

    static <tipo> <id> (<parametri>) {
        ...
    }

    static <tipo> <id> (<parametri>) {
        ...
    }
}
```

## Esecuzione di programma con metodi

- L'esecuzione comincia dal main
  - Anche il `main` è un metodo! (particolare)
- Invocazione → l'esecuzione passa al metodo
  - Associazione parametri
  - Esecuzione istruzioni del metodo
  - ... fino al `return` o alla fine del metodo
- Poi l'esecuzione ritorna al chiamante, all'istruzione successiva

Stefano Mizzaro - Metodi I

13

## Un altro esempio

```
class EsempioDiMetodo2 {
    static char convertiInMaiuscolo(char c) {
        if (c < 'a' || c > 'z')
            return(c);
        else
            return ((char) (c + 'A' - 'a'));
    }

    public static void main (String[] args) {
        char carattere;
        do {
            carattere = Leggi.unChar();
            System.out.print(
                convertiInMaiuscolo(carattere));
        } while (carattere != '\n');
    }
}
```

Stefano Mizzaro - Metodi I

14

## Parametri

- Formali
  - Usati alla definizione per definire cosa fa un metodo
  - $f(x) = 2x^2 + 4x + 9$
- Attuali
  - Usati all'invocazione per dire al metodo su quali dati lavorare
  - $f(5), f(5+3), f(5y), \dots$

Stefano Mizzaro - Metodi I

15

## Associazione fra parametri attuali e parametri formali

- All'invocazione il valore del parametro attuale viene assegnato al parametro formale
  - Come con un assegnamento
  - I tipi devono essere compatibili, come per un assegnamento
  - Formale: Left value, variabili
  - Attuale: Right value, espressioni
- Se ci sono più parametri, associazione posizionale (basata sulla posizione)
  - $f(x,y) = 2x^2 + 4y + 9xy$
  - $f(4,7)$

Stefano Mizzaro - Metodi I

16

## Cosa succede?

```
class Parametri {
    static void inc(int x) {
        x++;
    }

    public static void main (String[] args) {
        int y = 0;
        inc(y);
        System.out.println(y);
    }
}
```

```
>javac Parametri.java
>java Parametri
0
>
```

Stefano Mizzaro - Metodi I

17

## Perché

- Passaggio parametri **per valore**
- Il valore del parametro attuale viene copiato nel parametro formale
- Ma sono 2 variabili diverse
- Come per un assegnamento

```
public static void main (String[] args) {
    int x, y = 0;
    x = y;
    y++;
    System.out.println(x);
}
```

Stefano Mizzaro - Metodi I

18

## Metodo (...) per incremento

```
class Parametri {
    static int inc(int x) {
        return (x + 1);
    }

    public static void main (String[] args) {
        int y = 0;
        y = inc(y);
        System.out.println(y);
    }
}
```

- (non è un programma serio!!)

Stefano Mizzaro - Metodi I

19

## Perché i metodi sono utili

- Sintesi
  - Stesse operazioni da ripetere in più punti del programma
  - Evito ripetizione codice → Errori
- Modularità
  - Scompongo problema (programma) in sottoproblemi (sottoprogrammi)
  - "Estendo il Java con nuove istruzioni"
  - [anche se non è il modo migliore, lo vedremo]

Stefano Mizzaro - Metodi I

20

## Esempio: numeri complessi

- Radice quadrata di  $-1$ ?  $i$ 
  - $i * i = -1$
- $i$  è un numero immaginario
  - E anche  $2i$ ,  $5.7i$ , ecc. Ci si può lavorare "normalmente"
- Numeri complessi  $C$ 
  - Composti da parte reale e parte immaginaria
  - Es.:  $2i$ ,  $2\pi + 2i$ ,  $e + \pi i$ , ...
- Operazioni su numeri complessi  $C$ 
  - Somma, prodotto, ...
- Programma per calcolare il prodotto
  - $(a + bi) * (c + di) = (ac - bd) + (ad + bc)i$

Stefano Mizzaro - Metodi I

21

## Esempio (1/2)

```
/* Programma per moltiplicare numeri complessi.
   Versione 1. Un numero complesso e'
   rappresentato da due numeri. */

class Complessi {
    static double prodottoRe(
        double aRe, double aIm,
        double bRe, double bIm) {
        return (aRe * bRe - aIm * bIm);
    }

    static double prodottoIm(double aRe, double aIm,
        double bRe, double bIm) {
        return (aRe * bIm + aIm * bRe);
    }
}
```

Stefano Mizzaro - Metodi I

22

## Esempio (2/2)

```
public static void main (String[] args) {
    double xRe, xIm, yRe, yIm, zRe, zIm;
    System.out.print("Parte reale di x: ");
    xRe = Leggi.unDouble();
    System.out.print("Parte immaginaria di x: ");
    xIm = Leggi.unDouble();
    System.out.print("Parte reale di y: ");
    yRe = Leggi.unDouble();
    System.out.print("Parte immaginaria di y: ");
    yIm = Leggi.unDouble();
    zRe = prodottoRe(xRe, xIm, yRe, yIm);
    zIm = prodottoIm(xRe, xIm, yRe, yIm);
    System.out.print("(" + xRe + " + i * " + xIm + ")");
    System.out.print(" * ");
    System.out.print("(" + yRe + " + i * " + yIm + ")");
    System.out.print(" = ");
    System.out.println(
        "(" + zRe + " + i * " + zIm + ")");
}
```

Stefano Mizzaro - Metodi I

23

## Esempio

- Scrivere un metodo per stampare una matrice bidimensionale

```
class StampaMatrice {
    static void stampaMatrice(int[][] m) {
        for(int i = 0; i < m.length; i++){
            for (int j = 0; j < m[i].length; j++){
                System.out.print(m[i][j] + "\t");
                System.out.println();
            }
        }
    }

    public static void main (String[] args) {
        int[][] matrice =
            {{0,1,2},{3,4,5},{6,7,8},{9,10,100}};
        stampaMatrice(matrice);
    }
}
```

Stefano Mizzaro - Metodi I

24

## Esercizio

- Scrivere un metodo che dice se un numero  $n$  è primo o no
- Ossia, che riceve come parametro un numero intero positivo  $n$  e restituisce un valore booleano `true` se il parametro  $n$  è un numero primo, `false` altrimenti
- (Scrivete anche un main che invoca il metodo opportunamente)

Stefano Mizzaro - Metodi I

25

## Idea

- Guardo tutti i numeri minori di  $n$  (ciclo su  $i$ )
  - Escluso l'1
  - Basta arrivare a  $n/2$
- Controllo per ogni  $i$  se è un divisore di  $n$
- Quando trovo un divisore,  $n$  non è primo
  - Ne basta uno; mi fermo subito
- Se non ne trovo,  $n$  è primo
  - Devo guardarli tutti; esco solo alla fine

Stefano Mizzaro - Metodi I

26

## Primo.java

```
class Primo{
    public static void main (String[] args){
        System.out.print(
            "Inserisci un numero intero --> ");
        System.out.println(
            (primo(Leggi.unInt())?"E' ":"Non e' ")
            + " primo");
    }
    static boolean primo(int n) {
        for (int i = 2; i <= n/2; i++)
            if (n % i == 0)
                return false;
        return true;
    }
}
```

Stefano Mizzaro - Metodi I

27

## Prova (Traccia) di esecuzione

- Simulare il comportamento dell'interprete
- Molto utile quando un programma non funziona
- **Se non lo sapete fare non passate l'esame!!!**
- Tecnica:
  - Ci si scrivono tutte le variabili
  - Si esegue un'istruzione alla volta segnandosi
    - il valore che le variabili assumono man mano
    - Il punto a cui è giunta l'esecuzione
  - **SI FA MOLTA ATTENZIONE PERCHÉ È FACILISSIMO SBAGLIARE!!!**

Stefano Mizzaro - Metodi I

28

## Esempio

- Quale valore viene restituito se si invoca `abc(4)`?

```
static int abc(int x) {
    int y = 4;
    int k = 2;
    while (x >= y) {
        k = k+1;
        if (k > y)
            for (int i = 0; i < 2; i = i+1)
                x = x-1;
        else
            y = y-1;
    }
    return y + x + k;
}
```

Stefano Mizzaro - Metodi I

29


## Esercizio

- Quale valore viene restituito se si invoca `abc(8)`?

```
static int abc(int x) {
    int y = 6;
    int k = 2;
    while (x >= y) {
        k = k+1;
        if (k > y)
            for (int i = 0; i < 4; i = i+1)
                x = x-1;
        else
            y = y-3;
    }
    return y + x + k;
}
```

Stefano Mizzaro - Metodi I

30



## ■ Riassunto

- Metodi
  - Pezzi di programma
  - Con un nome
  - Richiamabili, invocabili
  - Funzioni e procedure
  - Parametri attuali e formali
- Prova/traccia di esecuzione
- Libro, fino a p. 111; Eserciziario: fino a inizio cap. 3
- Prossima lezione: ancora metodi

Stefano Mizzaro - Metodi I

31