I metodi - II

Stefano Mizzaro

Dipartimento di matematica e informatica
Università di Udine
http://www.dimi.uniud.it/mizzaro
mizzaro@dimi.uniud.it

Programmazione, lezione 10
27 ottobre 2004

Dove siamo

- Mattoni, Programmazione strutturata, Sviluppo incrementale, Array
- Intro metodi/sottoprogrammi
  - Definizione (intestazione e corpo) e uso
  - Parametri formali e attuali, associazione
  - Passaggio parametri per valore
  - Funzioni e procedure
  - Utilità dei metodi

Oggi

- Visibilità
- Durata
- Gestione dei metodi nella JVM
- Metodi sovraccarichi
- Funzioni matematiche predefinite
- Esempi

Visibilità (*scope*)

```
for (int i = 1; i <= 10; i++) {
  ...
}
```

- `i` visibile solo nel corpo del for

```
class Prova {
  public static void main (String[] args) {
    int j = 6;
    for (int i = 1; i <= 5; i++) {
      System.out.println(i);
      System.out.println(j);
    }
    System.out.println(i);
    System.out.println(j);
  }
}
```

```
>javac Prova.java
Prova.java:8: Undefined variable: i
    System.out.println(i);
                       ^
1 error
```


Regola di visibilità

- Una variabile è visibile unicamente all'interno del blocco in cui è dichiarata
  - E nei blocchi all'interno del blocco, ecc.
  - Parametri formali come le variabili (solo nel corpo del metodo)

```
class Visibilita {
  static void f (int i) {
    // Qui i e' visibile, j no
  }
  static void g (int j) {
    // Qui j e' visibile, i no
  }
  public static void main (String[] args) {
    // Qui non sono visibili ne' i ne' j
  }
}
```


Variabili con nomi =

```
class Visibilita {
  static void f (int i) {
    int j;
    ...
  }
  static void g (int j) {
    int i;
    ...
  }
  public static void main (String[] args) {
    int j;
    ...
  }
}
```

- Variabili con nomi = purché in blocchi diversi

Variabili all'esterno dei metodi

```
class Visibilita {
  static int x,y;

  static void f (int i) {
    ...
  }
  static int g (int j) {
    ...
  }
  public static void main (String[] args) {
    double j;
    ...
  }
}
```

- `static` (per ora...)
- Variabili locali a un metodo (eventualmente main)
- Variabili di classe (""globali"")

Modello a contorni

- Quindi:
  - Variabili
    - locali a un metodo (parametri formali)
    - di classe
    - definite in un blocco (ad es., for)
  - Variabili con nomi =
- Modello a contorni
  - Utile per capire dove una variabile è visibile
  - Un "contorno" (rettangolo) intorno a ogni blocco

Esempio

```
class Visibilita {
  static int i1;

  static void f (int i2) {
    // int i2; sarebbe lo stesso
    ...
    for (int i3 ...) {
      ...
      if (...) {
        int i4;
        ...
      }
    }
  }

  public static void main (String[] args) {
    int j;
    ...
  }
}
```


Blocchi annidati

- Variabili usate in un blocco ma definite in un blocco esterno: no problem
- Variabili con stesso nome in blocchi annidati
- 2 casi:
  1. Locale al metodo e di classe
     - Quella locale nasconde quella di classe
  2. All'interno di un blocco e locale al metodo
     - Errore, conflitto di nomi

Esempio caso 1

```
class Nascoste {
  static int x,y; //Def. var. globali

  static void f() {
    int x;
    x = 1;  // Locale
    y = 1;  // Globale
    System.out.println(x);
    System.out.println(y);
  }

  public static void main (String[] args) {
    x = 0; // Globale
    y = 0; // Globale
    f();
    System.out.println(x);
    System.out.println(y);
  }
}
```


Esempio caso 2

```
class VariabiliBlocchi {
  public static void main (String[] args) {
    int x = 0;

    if (x > 0) {
      int x = 1;
      x++;
    }
  }
}
```

```
>javac VariabiliBlocchi.java
VariabiliBlocchi.java:6: Variable 'x' is already defined
in this method.
      int x = 1;
          ^
1 error
```

Variabili "globali" vs. passaggio parametri

- Usate parametri, non variabili globali
- È più semplice capire cosa fa un metodo se basta leggere il metodo e non bisogna far riferimento a tutto il resto del programma

Esempio

```
class Parametri {
  static int x;

  static void inc() {
    x++;
  }

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

- Brutto!

Esercizio: traccia di esecuzione

```
class EsercizioScope1 {
  static int x, y;
  static int metodo1(int x) {
    x = 1;
    y = 1;
    System.out.println("metodo1 " + x + " " +y);
    return x + y;
  }
  static void metodo2() {
    int y;
    x = 2;
    y = metodo1(x);
    System.out.println("metodo2 " + x + " " +y);
  }
  public static void main (String[] args) {
    x = 0;
    y = 0;
    metodo2();
    System.out.println("main " + x + " " + y);
  }
}
```

LA TRACCIA DI ESECUZIONE È CONDIZIONE NECESSARIA PER PASSARE L'ESAME!

ergo

SE NON SAPETE FARE LA TRACCIA DI ESECUZIONE NON PASSATE L'ESAME!

La durata (*lifetime*)

- Da quando la memoria viene allocata a quando viene "cancellata"
- Diverso da visibilità:
  - Ad un dato momento dell'esecuzione, una variabile può esistere e non essere visibile
  - (se una variabile non esiste, non può essere visibile…)

Esempio

```
class Automatiche {
  static void f() {
    int i;
    g();
    ...;
  }
  static void g() {
    ...
  }
  public static void main (String[] args) {
    f();
  }
}
```

- Durante l'esecuzione di **`g`**, **`i`** esiste ma non è visibile

Gestione metodi nella JVM

- Come funziona la JVM
- Per ogni metodo c'è un record di attivazione
  - Zona di memoria che contiene le variabili e i parametri formali del metodo
- Pila (stack) di record di attivazione
  - Per ogni invocazione di metodo viene allocato un record di attivazione
  - I record di attivazione sono gestiti a Pila (uno sopra l'altro), con politica Last In First Out (LIFO)
  - Il record di attivazione in cima alla pila corrisponde al metodo in esecuzione in quel momento

Esempio

```
class EsempioPila {
  static void f (int x) {
    int y = 3;
  }

  static void g (int z) {
    int i = 2;
    f(z);
  }

  public static void main (String[] args) {
    int w = 1;
    g(w);
  }
}
```

- Simuliamo l'esecuzione costruendo la pila dei record di attivazione

Visibilità e durata

- La visibilità è un concetto statico
  - Si guarda il codice, ci si aiuta con il modello a contorni
- La durata è un concetto dinamico
  - Bisogna eseguire il programma
- P.S. Variabili di classe:
  - In un record di attivazione in fondo alla pila
  - Allocato a inizio esecuzione
  - Durata = tutta l'esecuzione

Esercizio

```
class ... {
  static void f() {
    int x;
    g();
    ...;
  }

  static void g() {
    ...;
  }

  public static void main (String[] args) {
    ...;
    f();
    g();
  }
}
```

- La variabile `x` esiste durante l'esecuzione di `g`?

Metodi sovraccarichi

- Firma di un metodo =
  - nome + elenco dei (tipi dei) parametri
  - Attenzione: non il tipo restituito!
- "Sovraccarico" (*overload*):
  - Metodi con nome uguale e firma diversa
  - Il nome del metodo è carico di più significati
  - Il Java distingue i metodi sulla base della loro firma

Esempio

```
class Overloading {

  static int somma (int x, int y) {
    return x + y;
  }

  static int somma (int x, int y, int z) {
    return x + y + z;
  }

  public static void main (String[] args) {
    System.out.println(somma(1,1));
    System.out.println(somma(1,2,3));
  }
}
```

```
  static double somma (int x, int y) {
    return (double) x + y;
  }
```

Funzioni matematiche predefinite

- `sqrt(double)`
- `exp(double)`
- `log(double)`
- `cos(double)`
- `sin(double)`
- `tan(double)`
- `pow(double, double)`
- `abs(double)`
- `floor(double)`
- `ceil(double)`
- `min(double, double)`
- `max(double, double)`
- `PI = 3.141592...`
- `E = 2.71828459...`

- Premettere **`Math.`**

Esercizi

- Metodo che calcola il MCD fra 2 numeri
- Programma che visualizza i numeri primi fra 2 e **`n`**, con **`n`** letto in input durante l'esecuzione
- Metodo che visualizza i numeri primi fra 2 e **`n`**, con **`n`** parametro del metodo
- Metodo che visualizza i primi n numeri primi

Riassunto

- Metodi
  - Visibilità (contorni, regola di visibilità, variabili locali a un metodo, di classe, locali a un blocco)
  - Durata, gestione dei metodi nella JVM, pila dei record di attivazione
  - Metodi sovraccarichi
  - Funzioni matematiche predefinite
  - Esempi
- Libro: quasi tutto il cap. 5.
- Eserciziario: quasi tutto il § 3.1
- Prossima lezione: ancora metodi e ricorsione