Programmazione
Tre esempi di firma di metodo

Eugenio Omodeo

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Trieste, 05.10.2015

```
class PiGreca{

    public static void main( String[] argg ){

        double piGreca;

        System.out.println( piGreca );

        piGreca = 4 * Math.atan(1);

        System.out.println( '\u03c0' + "=" + piGreca );
    }
}

// Quante invocazioni di metodo vedete?
// Quante firme ('signature') di metodo vedete?
// Che differenza notate fra invocazioni e firme?
```

Un tipo d'istruzione su cui soffermarci

La 'classe'

PiGreca

appena vista ci ha mostrato, fra l'altro, un importante tipo d'istruzione:

- L'*invocazione di metodo*: fra moltissimi metodi della dotazione iniziale, abbiamo scelto

  `System.out.println`

  (con un parametro)

N.B. sull'invocazione di metodo

Ci sono anche invocazioni di metodo che figurano

dentro le espressioni

(a destra di un'assegnazione o in un parametro di un'invocazione)

(Per es. Math.atan(1))

Metafora:

Il passaggio, nello studio di Java, da programmi costituiti da un semplice ‘main’ a programmi la cui realizzazione richiede la messa a punto di piú metodi

Una questione di scala — e non solo

Metafora:

Il passaggio, nello studio di Java, da programmi costituiti da un semplice ‘main’ a programmi la cui realizzazione richiede la messa a punto di piú metodi — non provenienti dalla dotazione di partenza del linguaggio, ma creati dallo stesso programmatore — rassomiglia al passaggio, nello studio dei viventi, dagli organismi monocellulari a quelli pluricellulari.

Una questione di scala — e non solo

Le funzioni di un programma sono realizzate tramite ***metodi***, che rapidamente, con il progressivo aumento di scala delle applicazioni, da pochi diventano centinaia, migliaia, miriadi.

Una questione di scala — e non solo

Le funzioni di un programma sono realizzate tramite ***metodi***, che rapidamente, con il progressivo aumento di scala delle applicazioni, da pochi diventano centinaia, migliaia, miriadi.

Il 'main', da cui ha avvio e si dipana l'esecuzione di un programma, è esso stesso un metodo; ma se tutto dovesse compiersi al suo interno, diventerebbe un monolite — arduo da erigere, arduo da modificare.

Una questione di scala — e non solo

Le funzioni di un programma sono realizzate tramite ***metodi***, che rapidamente, con il progressivo aumento di scala delle applicazioni, da pochi diventano centinaia, migliaia, miriadi.

Il 'main', da cui ha avvio e si dipana l'esecuzione di un programma, è esso stesso un metodo; ma se tutto dovesse compiersi al suo interno, diventerebbe un monolite — arduo da erigere, arduo da modificare.

> La complessità delle applicazioni resta dominabile solo se il programma rimane piccolo e dà avvio all'attività di altri metodi, che invocano altri metodi ancora.

Esercizio

Scrivete un programma che a partire da un ottetto di numeri dati calcoli i numeri tali che $\cdots$ $\cdots$ in base alla formula dei quattro quadrati di Eulero o a un'altra ad essa equipollente.

(Vedi lezione del 22.09.2015)

Esercizio

Scrivete un ~~programma~~ metodo che a partire da un ottetto di numeri dati calcoli i numeri tali che $\cdots$ $\cdots$ in base alla formula dei quattro quadrati di Eulero o a un'altra ad essa equipollente.

(Vedi lezione del 22.09.2015)

Esercizio

Scrivete un ~~programma~~ metodo che a partire da un ottetto di numeri dati calcoli i numeri tali che $\cdots$ $\cdots$ in base alla formula dei quattro quadrati di Eulero o a un'altra ad essa equipollente.

(Vedi lezione del 22.09.2015)

La primissima questione da affrontare è quella di assegnare una '*firma*' al metodo, indicando

il tipo dei suoi operandi e del suo risultato .

Affrontiamo altri tre problemi

- Stabilire, di un numero intero dato, $\geqslant 0$, se è PRIMO o no

Affrontiamo altri tre problemi

- Stabilire, di un numero intero dato, $\geqslant 0$, se è PRIMO o no
- Individuare, dato un numero pari $n \geqslant 4$, due numeri primi p, q tali che

$$p + q \quad = \quad n$$

Affrontiamo altri tre problemi

- Stabilire, di un numero intero dato, $\geqslant 0$, se è PRIMO o no
- Individuare, dato un numero pari $n \geqslant 4$, due numeri primi p, q tali che

$$p + q \quad = \quad n$$

- Individuare, dato un numero intero $n \geqslant 0$, quattro interi a, b, c, d tali che

$$n \quad = \quad a^2 + b^2 + c^2 + d^2$$

Affrontiamo altri tre problemi

- Stabilire, di un numero intero dato, $\geqslant 0$, se è PRIMO o no
- Individuare, dato un numero pari $n \geqslant 4$, due numeri primi p, q tali che

$$p + q \quad = \quad n$$

- Individuare, dato un numero intero $n \geqslant 0$, quattro interi a, b, c, d tali che

$$n \quad = \quad a^2 + b^2 + c^2 + d^2$$

Esercizio: Di che tipo è ciascun risultato ?

Tipi dei risultati

Nei tre rispettivi casi:

test di primalità: Un **boolean**

—potremmo, per strafare, *restituire* 1 oppure un divisore non banale.

Tipi dei risultati

Nei tre rispettivi casi:

test di primalità: Un **boolean**

—potremmo, per strafare, *restituire* 1 oppure un divisore non banale.

scomposizione di Golbach (chi era costui?): Dovremmo fornire una coppia di numeri, ma come aggregarli in un risultato singolo?

—potremmo, giocando al ribasso, contentarci di *restituire* *p*.

Tipi dei risultati

Nei tre rispettivi casi:

test di primalità: Un **boolean**

—potremmo, per strafare, *restituire* 1 oppure un divisore non banale.

scomposizione di Golbach (chi era costui?): Dovremmo fornire una coppia di numeri, ma come aggregarli in un risultato singolo?

—potremmo, giocando al ribasso, contentarci di *restituire* *p*.

scomposizione di Lagrange: qui non si sfugge — occorre *restituire* una quaterna.

Che significa '*restituire*' ?

In Java esiste una parola-chiave, **return**, utilizzata dai metodi invocati per trasmettere il loro risultato

via memoria

al metodo invocante[1]

[1] Niente a che vedere con una stampa!

Che significa '*restituire*' ?

In Java esiste una parola-chiave, **return**, utilizzata dai metodi invocati per trasmettere il loro risultato

via memoria

al metodo invocante[1]

L'istruzione in cui figura tale parola-chiave è la

return ⟨espressione⟩ ;

[1] Niente a che vedere con una stampa!

Che significa '*restituire*' ?

In Java esiste una parola-chiave, **return**, utilizzata dai metodi invocati per trasmettere il loro risultato

via memoria

al metodo invocante[1]

L'istruzione in cui figura tale parola-chiave è la

return ⟨espressione⟩ ;

(Ma l'espressione può anche mancare — Quando?)

[1] Niente a che vedere con una stampa!

Una rilessione riguardante il flusso di controllo

Programmare tramite metodi significa (**metaforicamente**):

impegnare una 3^{a} dimensione :

N.B.: Si tratta solo di una metafora !

Una rilessione riguardante il flusso di controllo

Programmare tramite metodi significa (**metaforicamente**):

impegnare una 3^{a} dimensione :

1. un blocco di sole istruzioni di assegnamento, impegna 1 sola dimensione ;

N.B.: Si tratta solo di una metafora !

Una rilessione riguardante il flusso di controllo

Programmare tramite metodi significa (**metaforicamente**):

impegnare una 3^{a} dimensione :

1. un blocco di sole istruzioni di assegnamento, impegna 1 sola dimensione ;
2. le istruzioni iterative, ne impegnano anche una 2^{a} ;

N.B.: Si tratta solo di una metafora !

Una rilessione riguardante il flusso di controllo

Programmare tramite metodi significa (**metaforicamente**):

impegnare una 3^{a} dimensione :

1. un blocco di sole istruzioni di assegnamento, impegna 1 sola dimensione ;
2. le istruzioni iterative, ne impegnano anche una 2^{a} ;
3. l'invocazione di un metodo lancia il controllo nella 3^{a} dimensione — il 'riatterraggio' avverrà allorché il metodo invocato restituirà il controllo (tramite una return);

N.B.: Si tratta solo di una metafora !

Una rilessione riguardante il flusso di controllo

Programmare tramite metodi significa (**metaforicamente**):

impegnare una 3^{a} dimensione :

1. un blocco di sole istruzioni di assegnamento, impegna 1 sola dimensione ;
2. le istruzioni iterative, ne impegnano anche una 2^{a} ;
3. l'invocazione di un metodo lancia il controllo nella 3^{a} dimensione — il 'riatterraggio' avverrà allorché il metodo invocato restituirà il controllo (tramite una return);
4. allorché entreranno in gioco, le eccezioni impegneranno anche una 4^{a} dimensione.

N.B.: Si tratta solo di una metafora !

Come si formano degli aggregati ?

Anticipazione:

Anche le forme piú tradizionali di aggregazione

- 'stringhe' di caratteri ,
- liste di entità omogenee per tipo , (ad es. gli array)
- record eterogenei ,

richiederanno il supporto di appropriate

classi di oggetti

Come regolarsi quando manca il risultato atteso ?

Siamo poi sicuri che esista sempre

- la scomposizione di Goldbach
- la scomposizione di Lagrange

?

(Java ci fornirà valori 'fittizi' di vario tipo, fra cui:

Integer.MIN_VALUE

NaN

null

)

In conclusione, tre firme utilizzabili:

```
public static boolean primo( int  n )
```

```
public static int goldbach( int  n )
```

```
public static int[] lagrange( int  n )
```

In conclusione, tre (+3) firme utilizzabili:

```
public static boolean primo( int  n )
( o anche... ) public static int primo( int  n )
public static int goldbach( int  n )
( o anche... ) public static int[] goldbach( int  n )
public static int[] lagrange( int  n )
( o anche... ) public static String lagrange( int  n )
```

Nota terminologica

Nella *firma* di un metodo, compaiono identificatori degli operandi. Questi vengono chiamati

- *parametri* **formali** ;

Nell'*invocazione* di un metodo, compaiono espressioni che vengono trasmesse come valori degli operandi. Questi vengono chiamati

- *parametri* **attuali** .