Programmazione Orientata agli Oggetti in Linguaggio Java

Sintassi e Semantica
Peculiarità di Java

versione 2.0

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

Introduzione

ATTENZIONE
alle differenze
tra Java e il C/C++

- In sintesi
  - ⇨con riferimento alle caratteristiche comuni alla programmazione procedurale, ci sono molti punti in comune con il C/C++
- Ma ci sono anche alcune differenze
  - ⇨modifiche introdotte in Java per evitare gli aspetti “pericolosi” della programmazione C/C++

Principali Differenze

- Le principali differenze
  - ⇨in Java non esistono puntatori (non è consentita la gestione diretta della memoria)
  - ⇨Java è più fortemente tipato del C++ e meno disinvolto nelle conversioni di tipo
  - ⇨Java richiede l’inizializzazione obbligatoria delle variabili prima dell’uso
  - ⇨utilizzo delle stringhe
  - ⇨utilizzo degli array

Principali Differenze

- I) In Java non esistono i puntatori
  - ⇨Java non prevede i tipi puntatore
  - ⇨è noto, infatti, che la programmazione sui puntatori è complessa e fonte di errori
  - ⇨d'altro canto, Java prevede i riferimenti
  - ⇨tutto quello che è possibile fare con i puntatori si può fare (in modo più controllabile) usando i riferimenti

Principali Differenze

- II) Java è più fortemente tipato del C++
  - ⇨in Java le conversioni di tipo sono fatte con maggiore attenzione dal processore
  - ⇨in Java il tipo a valori booleani NON è considerato un tipo numerico
  - ⇨questo evita alcuni frequenti errori di programmazione

Principali Differenze

- Il tipo boolean
  - ⇨chiamato diversamente per rimarcare la differenza (in C++ il tipo si chiama bool)
  - ⇨ammette come valori esclusivamente le costanti true e false e non valori numerici
- Esempio

```
boolean b; int i = 10;
b = true; // consentito
b = i;  // ERRORE: sintatticamente scorretto
```


Principali Differenze

- In particolare
  - ⇨le condizioni nelle strutture di controllo devono essere rigorosamente espressioni a valori booleani
  - ⇨non sono ammessi i valori interi
- Esempio

```
if (a == b) {...}  // ok consentito
if (a = b) { ...}  // ERRORE: sintatticamente scorretto
int i; while (i) {...}  // ERRORE: sintatticamente scorretto
```

Principali Differenze

- Regole di compatibilità tra tipi
  - ⇨tutti i tipi numerici sono compatibili
  - ⇨enumerabili (byte, short, int, long, char)
  - ⇨virgola mobile (float, double)
  - ⇨boolean NON è un tipo numerico
- Attenzione
  - ⇨in Java non esistono tipi per i numeri privi di segno (unsigned)

Principali Differenze

- Conversioni
  - ⇨troncamenti ed ampliamenti
  - ⇨gli ampliamenti sono effettuati automaticamente dal processore
  - ⇨viceversa, a differenza del C/C++, non sono ammessi troncamenti automatici
  - ⇨in altri termini, il linguaggio richiede che l'utente indichi chiaramente tutti i casi in cui ritiene ammissibile un troncamento

Principali Differenze

- Esempio
  - ⇨in C++ troncamenti ed ampliamenti sono automatici

```
#include <iostream.h>
void main() {
    short b; int i;
    b = 50000;
    cout << b << endl; // - 15536
    i = 200000;
    b = i;
    cout << b << endl; // 3392
}
```

in Java: errore sintattico "Possible loss of precision"

Principali Differenze

- Troncamenti in Java
  - ⇨devono essere richiesti esplicitamente dal programmatore
  - ⇨utilizzando un operatore di cast
  - ⇨in questo modo si evitano perdite di precisione non volute

Principali Differenze

- L'esempio precedente in Java

```
public static void main(String[] args) {
    short b;
    int i;
    b = (short) 50000;
    System.out.println(b); // - 15536
    i = 200000;
    b = (short) i;
    System.out.println(b); // 3392
}
```


Principali Differenze

ATTENZIONE
all'inizializzazione
obbligatoria

- III) Java richiede l'inizializz. obbligatoria
  - ⇨in Java una variabile deve essere stata necessariamente inizializzata prima di essere usata in un'espressione
  - ⇨altrimenti il compilatore solleva un errore sintattico
  - ⇨questo evita errori dovuti alla mancata inizializzazione e all'uso di valori casuali

Principali Differenze

- Esempio

```
public static void main (String[] args) {
    int i;
    while (i < 10) {
        System.out.println(i);
        i++;
    }
}
```

```
E:\tmp\Prova.java:6: variable i might not have
                                been initialized
        while (i < 10) {
                ^
1 error
```


Principali Differenze

- Gestione di stringhe e array
  - ⇨ulteriore differenza rispetto alla programmazione procedurale
- In sintesi
  - ⇨si tratta in entrambi i casi di oggetti
  - ⇨la sintassi relativa è volutamente ingannevole (lascia pensare di lavorare con valori di base)
  - ⇨la semantica ha delle peculiarità

Stringhe

- Le stringhe
  - ⇨oggetti della classe java.lang.String
- Sintassi
  - ⇨nella manipolazione delle stringhe la sintassi di Java è spesso ingannevole (come in C++)
  - ⇨si ha l'impressione di manipolare valori di un tipo di base
  - ⇨viceversa si sta lavorando con classi e riferimenti

Stringhe

- Esempio
  - ⇨creazione di una stringa
  - ⇨String s = "Prova";
  - ⇨sembrerebbe simile a int x = 1;
- Viceversa
  - ⇨equivale a creare un oggetto
  - ⇨ovvero a scrivere:
  - ⇨String s = new String("Prova");

-crea un oggetto di tipo java.lang.String
-gli assegna il valore "Prova"
-dichiara un riferimento s all'oggetto creato

Stringhe

riferimento implicito per l'oggetto costante, automaticamente creato dalla macchina virtuale

- Analogamente
  - ⇨le stringhe costanti sono oggetti costanti, con riferimenti impliciti
- Esempio: System.out.println("Prova");
  - ⇨equivale a: final String _rif = new String("Prova");
  - ⇨System.out.println(_rif);
- Infatti, è possibile scrivere anche
  - ⇨System.out.println("Prova".length()); // stampa 5

Stringhe

- Un'operazione importante sulle stringhe
  - ⇨concatenazione: +
- Esempio
  - ⇨String nome = new String("Mario");
  - ⇨String cognome = new String("Rossi");
  - ⇨String nomeECognome = "Sig." + nome + cognome;
- Utilizzata frequentemente con println()
  - ⇨System.out.println(nomeECognome + "\n");

Stringhe

concatena la stringa “Valore di x: “ con il valore intero 2 e stampa “Valore di x: 2”

- Caratteristica della concatenazione
  - ⇨qualsiasi dato può essere concatenato con una stringa per restituire una nuova stringa
- Esempio
  - ⇨int x = 2;
  - ⇨System.out.println(“Valore di x: “ + x);
- Caratteristica importante
  - ⇨println() ha un unico argomento di tipo String
  - ⇨per stampare vari valori bisogna concatenarli

Stringhe

- Altre operazioni sulle stringhe
  - ⇨metodi di java.lang.String
- Lunghezza: public int length()
  - ⇨es: String nome = “Franco”;
  - ⇨System.out.println(nome.length()); // stampa 6
- i-esimo carattere: public char charAt(int i);
  - ⇨es: String nome = “Franco”;
  - ⇨System.out.println(nome.charAt(3)); // stampa ‘n’

Stringhe

ATTENZIONE
gli oggetti di tipo java.lang.String sono immutabili

- Ma...
  - ⇨la principale peculiarità delle stringhe in Java è che NON possono essere cambiate
  - ⇨java.lang.String non offre metodi per cambiare il valore di un oggetto di tipo String
  - ⇨ci sono ragioni complesse (design pattern>>)
- Stringhe modificabili
  - ⇨oggetti di java.lang.StringBuffer (>>)

Stringhe

- Esempio
  - ⇨non è possibile modificare un carattere in una stringa
- E la concatenazione ?
  - ⇨di fatto crea nuovi oggetti e non modifica i precedenti
- Esempio
  - ⇨String nome = “Franco”;
  - ⇨nome = nome + “Rossi”;

equivalente a:
String nome = new String(“Franco”);
nome = new String(nome + “Rossi”);

Array

ATTENZIONE
gli array di Java sono oggetti

- Gli array di Java sono oggetti
  - ⇨manipolati attraverso riferimenti
  - ⇨esistono due tipi di array
- Array di valori di un tipo primitivo
  - ⇨oggetto che mantiene una collezione di valori del tipo primitivo
- Array di riferimenti
  - ⇨oggetto che mantiene una collezione di riferimenti ad altri oggetti

Array

- Attenzione
  - ⇨la sintassi di Java cerca di rendere l'utilizzo degli array familiare ai programmatori C
  - ⇨come effetto collaterale "maschera" la vera natura degli array e può confondere le idee
  - ⇨bisogna viceversa tenere in mente che l'array è un oggetto e viene manipolato con metodi e proprietà come tutti gli altri oggetti

Array: Esempio

```
public class ProvaArray {
   // un modo "classico" per lavorare con un array
   final static int N = 5;
   public static void main(String[ ] args) {
      int[ ] array = new int[ N ];

      // un array di numeri pari
      for (int i = 0; i < N; i++) {
         array[ i ] = i * 2;
      }
      for (int i = 0; i < N; i++) {
         System.out.print(array[ i ] + " ");
      }
   }
}
```


Array

- Le caratteristiche comuni
  - ⇨si tratta di collezioni di variabili tutte dello stesso tipo; è necessario specificare la dimensione; es: int[ ] array = new int[N];
  - ⇨nome delle componenti: nome dell'array più un indice numerico; es: array[2]
  - ⇨numerazione a base 0
  - ⇨utilizzo delle componenti: possibilità di utilizzare espressioni; es: array[i] = i * 2;

Array

- Le differenze
  - ⇨si tratta di oggetti e quindi vengono creati usando new e usati attraverso i riferimenti
    es: int[ ] array = new int[3];
  - ⇨possono contenere sia variabili di tipi primitivi che variabili riferimento
    es: Circonferenza[ ] coll = new Circonferenza[3];
    es: String[ ] nomi = new String[N];

Array

- Le differenze (continua)
  - ⇨la dimensione specificata alla dichiarazione può essere il valore di una variabile (non necessariamente una costante)
    es: int x; x = it.unibas.Console.leggiIntero();
    float[ ] temperature = new float[x];
  - ⇨ma, una volta creato, la dimensione dell'array non può essere cambiata (non sono possibili aggiunte o elimin. di elementi)

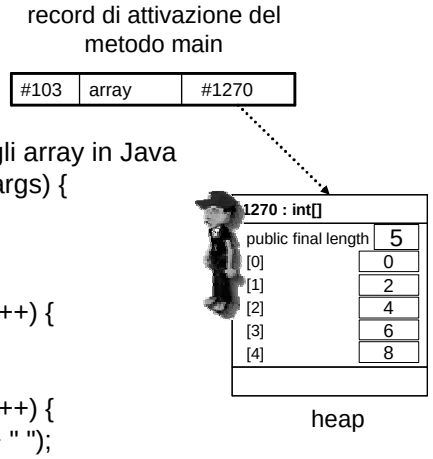
Array

- Le differenze (continua)
  - ogni oggetto array ha una proprietà pubblica chiamata length che consente di conoscerne la dimensione
    es: System.out.println(temperature.length);
  - questo è necessario proprio perchè la dimensione non deve necessariamente essere il valore di una costante
  - attenzione: il valore di length è costante

Array: Esempio

```
public class ProvaArray {
   // il modo tipico di lavorare con gli array in Java
   public static void main(String[ ] args) {
      int[ ] array = new int[ 5 ];

      // un array di numeri pari
      for (int i = 0; i < array.length; i++) {
         array[ i ] = i * 2;
      }
      for (int i = 0; i < array.length; i++) {
         System.out.print(array[ i ] + " ");
      }
   }
}
```



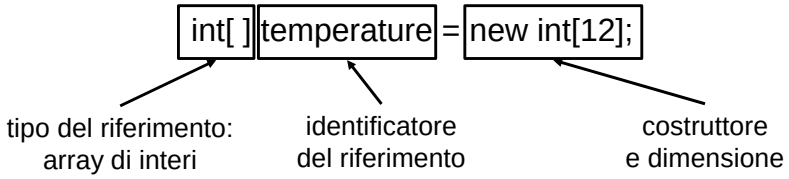
Array

ATTENZIONE
alla posizione
delle parentesi quadre

- Le differenze (continua)
  - ⇨la sintassi della dichiarazione è leggermente diversa da quella del C/C++
  - ⇨cambia la posizione delle parentesi quadre
- Sintassi
  - ⇨*<tipo>[ ] <identif> = new <tipo>[<dimensione>];*
  - ⇨es: int[ ] temperature = new int[12];

Array

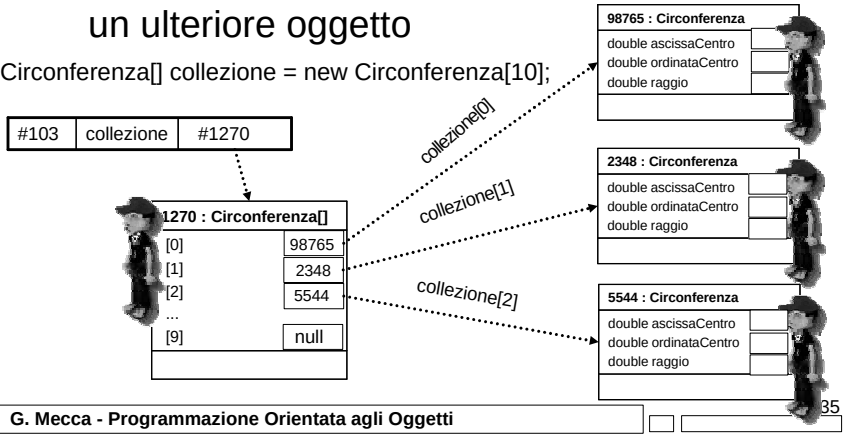
- In effetti
  - ⇨Java consente di utilizzare anche una sintassi tradizionale per le parentesi quadre
  - ⇨ma quella vista è decisamente più chiara
- Infatti



Array di Riferimenti

- Array di riferimenti
  - ⇨ogni variabile dell'array è un riferimento ad un ulteriore oggetto

Circonferenza[] collezione = new Circonferenza[10];



Array

Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 100
at Prova.main(Prova.java:15)

- La differenza principale
  - ⇨non è consentito lo sconfinamento
  - ⇨la macchina virtuale "solleva un'eccezione"
- Esempio
  - ⇨int[ ] temperature = new int[12];
  - ⇨System.out.println(temperature[100]);
  - ⇨System.out.println(temperature[12]);

nota: anche in questo caso viene sollevata l'eccezione

Inizializzazione degli Array

- Un aspetto tradizionale
  - ⇨inizializzazione alla dichiarazione
- Un aspetto nuovo
  - ⇨inizializzazione automatica
- Inizializzazione alla dichiarazione
  - ⇨analoga a quella del C/C++
  - ⇨float[ ] temperature = {12, 1.5, 22.4, 0};

Inizializzazione degli Array

- Inizializzazione automatica
  - ⇨anche per gli array vale la regola dell'inizializzazione obbligatoria
  - ⇨Java cerca di semplificare la vita al programmatore
  - ⇨ed inizializza automaticamente i valori dell'array
- Regola di inizializzazione autom. di Java
  - ⇨"regola del valore nullo"

Inizializzazione degli Array

- In particolare
  - ⇨i valori dei tipi primitivi vengono inizializzati, a seconda del tipo, al valore intero 0, al valore reale 0.0, al carattere di codice Unicode \u000, al valore booleano false
  - ⇨i riferimenti vengono inizializzati a null
- Di conseguenza
  - ⇨il valore di una componente di un array non è mai indefinito

Array Multidimensionali

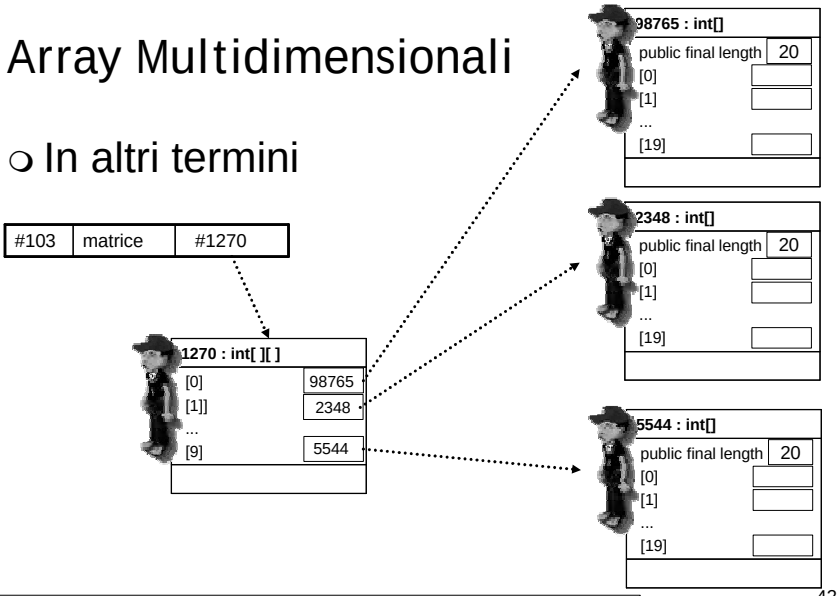
- In effetti
  - ⇨in Java non esistono veri e propri array multidimensionali
  - ⇨esistono però array di riferimenti ad array, che si comportano come se fossero array multidimensionali
- Sintassi
  - ⇨*<tipo>[ ][ ] <identif> = new <tipo>[<dim1>] [<dim2>];*
  - ⇨es: int[ ][ ] matrice = new int[10][20];

Array Multidimensionali

- In questo caso
  - ⇨int[ ][ ] matrice = new int[10][20];
  - ⇨ho dichiarato un array di 10 rifer. ad oggetti
  - ⇨ciascuno degli oggetti è un array di 20 interi
- Ho ottenuto
  - ⇨10 riferimenti ad array: matrice[0]...matrice[9]
  - ⇨ognuno dei quali contiene 20 interi
  - ⇨elemento i, j: matrice[i][j]

Array Multidimensionali

- In altri termini



Esempio: La Tabellina Pitagorica

```
public static void main(String[] args) {
        int[ ][ ] matrice = new int[10][10];

        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 10; j++) {
                matrice[i][j] = (i + 1) * (j + 1);
            }
        }

        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 10; j++) {
                System.out.print(matrice[i][j] + " ");
            }
            System.out.println();
        }
}
```


Riassumendo

- Introduzione
- Principali Differenze
- Stringhe
- Array
  - ⇨Array di Riferimenti
  - ⇨Inizializzazione degli Array
  - ⇨Array Multidimensionali

Termini della Licenza

- This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/1.0/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

- Questo lavoro viene concesso in uso secondo i termini della licenza “Attribution-ShareAlike” di Creative Commons. Per ottenere una copia della licenza, è possibile visitare http://creativecommons.org/licenses/by-sa/1.0/ oppure inviare una lettera all’indirizzo Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.