Programmazione Orientata agli Oggetti in Linguaggio Java

Sintassi e Semantica
Riferimenti

versione 2.1

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

Sommario

- Riferimenti
- Confronti
- Il Riferimento this
- Passaggio dei Parametri

Riferimenti

ATTENZIONE
gli oggetti non si manipolano mai direttamente, ma solo attraverso riferimenti

- Dati in un programma Java
  - ⇨dati dei tipi primitivi
  - ⇨riferimenti
- Riferimenti
  - ⇨dati che servono per manipolare gli oggetti
  - ⇨fungono da intermediari rispetto all'oggetto e consentono di inviare messaggi

Riferimenti

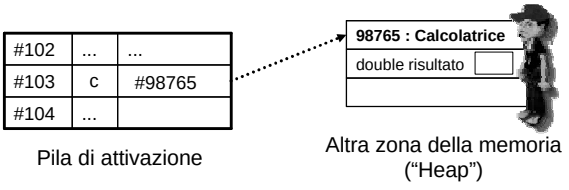
- Dichiarazione di un riferimento
  - ⇨tipo del riferimento e identificatore
- Tipo del riferimento
  - ⇨nome di una classe Java
  - ⇨i riferimenti sono tipati
- Valore per un riferimento
  - ⇨identificatore di un oggetto della classe corrispondente

Riferimenti

dichiara una variabile riferimento chiamata c che può assumere come valori identificatori di oggetti di tipo Calcolatrice

- Esempio
  - Calcolatrice c;
  - Partita p;
  - java.io.BufferedReader br;
- Semantica della dichiarazione
  - assegna alla variabile c di tipo riferimento uno spazio nella memoria
  - che può contenere l'identificatore di un oggetto della classe corrispondente

Riferimenti



- Attenzione
  - il riferimento e l'oggetto a cui si riferisce sono due dati completamente diversi
  - che corrispondono a zone di memoria diverse
- Esempio
  - variabile riferimento in un metodo: registro nello stack (nello stack frame del metodo)
  - oggetto corrispondente: heap

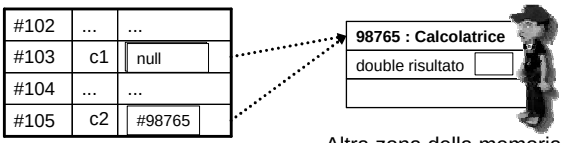
Riferimenti

- Operazioni possibili su un riferimento
  - ⇨assegnazione
  - ⇨invio di messaggi all'oggetto a cui si riferisce
  - ⇨passaggio dei parametri
- Assegnazione
  - ⇨con un costruttore
  - ⇨con il valore di un altro riferimento
  - ⇨con il valore speciale null

Riferimenti

- Tipicamente
  - ⇨l'assegnazione avviene chiamando un costruttore
  - ⇨viene creato un nuovo oggetto nello heap ed il suo id viene assegnato al riferimento
- Esempio
  - ⇨Calcolatrice c = new Calcolatrice(); // ogg #1234
  - ⇨Calcolatrice c1 = new Calcolatrice(); // ogg #678

Riferimenti



Pila di attivazione

Altra zona della memoria ("Heap")

- Esempi
  - Calcolatrice c1 = new Calcolatrice();
  - Calcolatrice c2;
  - c2 = c1; // i due riferim. puntano allo stesso oggetto
  - c1.somma(10, 20);
  - System.out.println(c2.getRisultato());
  - c1 = null;

in questo esempio sono mostrate tutte e tre le forme di assegnazione principali per un riferimento

Riferimenti

- La funzione di intermediazione
  - un riferimento funge da intermediario ogni volta che viene utilizzato il .
- Esempi
  - c1.somma(10, 20);
  - int[] array = new int[10];
  - System.out.println(array.length);

c1 chiede all'oggetto corrispondente di eseguire somma(10, 20)

array chiede all'oggetto corrispondente di utilizzare la proprietà length

Confronti

- Una annotazione importante
  - ⇨il riferimento funge da intermediario per l'oggetto solo quando si utilizza il punto
- In tutti gli altri casi
  - ⇨si lavora con il valore del riferimento
- Esempio
  - ⇨un pezzo di codice che, date due circonferenze, verifica se hanno gli stessi valori (centro e raggio)

Confronti

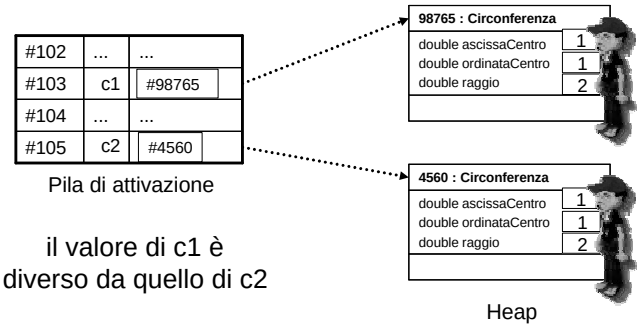
- Una prima soluzione (errata)

```
Circonferenza c1 = new Circonferenza (1, 1, 2);
Circonferenza c2 = new Circonferenza (1, 1, 2);
if (c1 == c2) {
   System.out.println("Circonferenze di valore uguale");
}
```

codice scorretto: non confronta i valori delle circonferenze
confronta i valori dei riferimenti e restituisce false

Confronti

```
Circonferenza c1 = new Circonferenza (1, 1, 2);
Circonferenza c2 = new Circonferenza (1, 1, 2);
if (c1 == c2) {
    System.out.println (“Circonferenze uguali”);
}
```



il valore di c1 è diverso da quello di c2

Confronti

- Il modo corretto di confrontare i valori

```
Circonferenza c1 = new Circonferenza (1, 1, 2);
Circonferenza c2 = new Circonferenza (1, 1, 2);
if (c1.getAscissaCentro() == c2.getAscissaCentro() &&
    c1.getOrdCentro() == c2.getOrdinataCentro() &&
    c1.getRaggio() == c2.getRaggio()) {
    System.out.println (“Circonferenze uguali”);
}
```

Confronti

- Viceversa
  - ⇨se volessi sapere se si tratta esattamente dello stesso oggetto

```
Circonferenza c1 = new Circonferenza (1, 1, 2);
Circonferenza c2 = c1;
if (c1 == c2) {
   System.out.println("Riferim. allo stesso oggetto");
}
```

codice corretto in questo caso
voglio sapere se i due rif. puntato allo stesso oggetto

Confronti

ATTENZIONE
al confronto tra oggetti

- Di conseguenza
  - ⇨attenzione ai confronti tra oggetti
- Due tipi di confronti
  - ⇨confronti tra i riferimenti (confrontano gli id. degli oggetti puntati): servono a verificare se due rif. puntano allo stesso oggetto
  - ⇨confronti tra i valori degli oggetti: servono a verificare se oggetti diversi hanno gli stessi valori per le proprietà

Confronti

- Un altro esempio
  - ⇨uguaglianza di stringhe
  - ⇨verificare se due nomi sono uguali a “Paolo”

```
String nome1 = new String (“Paolo”);
String nome2 = new String (“Paolo”);
if (nome1 == nome2 && nome1 == “Paolo”) {
    System.out.println(“Si chiamano entrambi Paolo”);
}
```

codice scorretto: non confronta i valori delle stringhe ma i valori dei riferimenti

Confronti

parametro: riferimento ad una stringa

- Il metodo equals() di java.lang.String
  - ⇨un metodo che consente di confrontare il valore di due stringhe e non i riferimenti
- Prototipo
  - ⇨public boolean equals (java.lang.String s);
- Semantica
  - ⇨confronta il valore dell’oggetto stringa (this) con il valore della stringa il cui riferimento è s

Confronti

- La versione corretta

```
String nome1 = new String (“Paolo”);
String nome2 = new String (“Paolo”);
if (nome1.equals(nome2) && nome1.equals(“Paolo”)) {
    System.out.println(“Si chiamano entrambi Paolo”);
}
```

confronto la stringa il cui riferimento è nome1
con la stringa costante “Paolo” (riferimento implicito;
in alternativa: “Paolo”.equals(nome1)

Confronti

- Per le circonferenze
  - ⇨potrei aggiungere un metodo equals simile a quello di String
  - ⇨in modo che ciascun oggetto di tipo Circonferenza sappia confrontarsi con un’altra Circonferenza
  - ⇨public boolean equals(Circonferenza c);
  - ⇨devo scriverlo esplicitamente

Confronti

```
public class Circonferenza {
   ...
   public boolean equals (Circonferenza c) {
      if (this.ascissaCentro == c.getAscissaCentro() &&
          this.ordCentro  == c.getOrdinataCentro() &&
          this.raggio == c.getRaggio()) {
          return true;
      }
      return false;
   }
   ...
}
```


Confronti

- A questo punto

```
Circonferenza c1 = new Circonferenza (1, 1, 2);
Circonferenza c2 = new Circonferenza (1, 1, 2);
if (c1.equals(c2)) {
    System.out.println (“Circonferenze uguali”);
}
```

Il Riferimento this

ATTENZIONE
al riferimento this

- Riferimento di un oggetto a sè stesso
  - ⇨non ha senso nei metodi statici
  - ⇨è particolare perchè si riferisce, a seconda del contesto, ad oggetti diversi
- Esempio: il metodo somma di Calcolatrice

```
public void somma(double a, double b) {
    this.risultato = a + b;
}
```


Il Riferimento this

```
public void somma(double a, double b) {
    this.risultato = a + b;
}
```

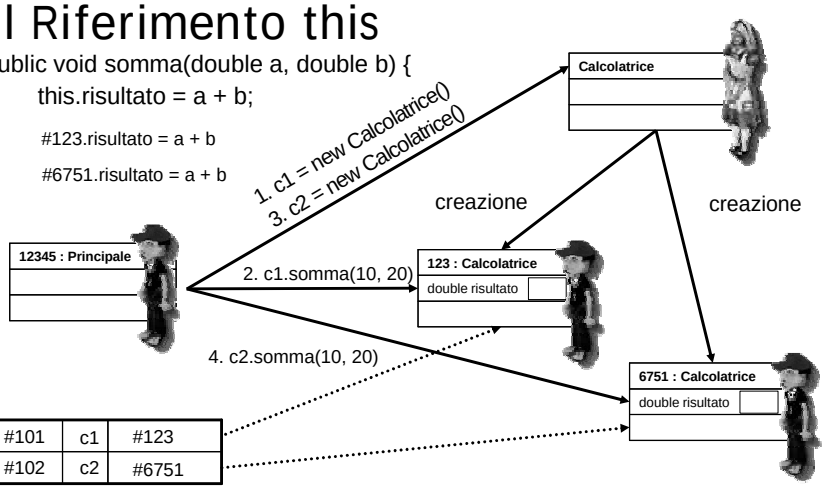
- Esempio
  - ⇨Calcolatrice c1 = new Calcolatrice(); // oggetto #123
  - ⇨c1.somma (10, 20);
  - ⇨Calcolatrice c2 = new Calcolatrice(); // oggetto #6751
  - ⇨c2.somma (10, 20);
- In questo caso
  - ⇨nell'esecuzione della prima chiamata (c1.somma(10, 20)), this ha il valore di c1 (#123), nella seconda chiamata ha il valore di c2 (#6751)

Il Riferimento this

```
public void somma(double a, double b) {
    this.risultato = a + b;
}
```

#123.risultato = a + b

#6751.risultato = a + b



#101	c1	#123
#102	c2	#6751

Il Riferimento this

- Regola generale
  - ⇨nel codice del metodo, il valore del riferimento this varia da chiamata a chiamata
  - ⇨ogni volta ha il valore del riferimento utilizzato per effettuare la chiamata del metodo (riferimento all'oggetto a cui è indirizzato il messaggio)
  - ⇨viene anche detto "parametro implicito" del metodo

Passaggio dei Parametri

- Il passaggio dei parametri in Java
  - ⇨diverse vedute sui libri di testo
  - ⇨cerchiamo di fare chiarezza
- Per cominciare
  - ⇨parametri e argomenti in Java possono essere di due tipi (come le variabili)
  - ⇨di tipo di base o di tipo riferimento
  - ⇨NON possono essere oggetti

Passaggio dei Parametri

- In sintesi
  - ⇨Java prevede esclusivamente il passaggio per valore
  - ⇨per cui non è possibile, in un metodo, modificare il valore di un argomento attraverso un parametro
- Ma
  - ⇨Java prevede i riferimenti, e con i riferimenti nei metodi si possono modificare gli oggetti

Passaggio dei Parametri

- In dettaglio
  - ⇨distinguiamo le due tipologie di parametri
- Passaggio dei parametri dei tipi di base
  - ⇨nel caso di valori dei tipi di base il passaggio avviene sempre per valore
  - ⇨quindi non è possibile in alcun modo modificare il valore di un argomento di un tipo di base attraverso un parametro in un metodo

Passaggio dei Parametri

- Esempio n.1: dati primitivi

```
public void somma(double a, double b) {
      this.risultato = a + b;
}

public static void main(String[ ] args) {
      double x, y;
      x = 20; y = 10;
      Calcolatrice c = new Calcolatrice();
      c.somma(x, y);
}
```

-nella pila viene allocato il record di attivazione per somma

-il valore di x viene copiato nello spazio di memoria di a

-il valore di y viene copiato nello spazio di memoria di b

Passaggio dei Parametri

- Se scrivessi

```
public void somma(double a, double b) {
      this.risultato = a + b;
      a++;
}
public static void main(String[ ] args) {
      double x, y;
      x = 20; y = 10;
      Calcolatrice c = new Calcolatrice();
      c.somma(x, y);
      System.out.println(x); // stampa 20
}
```

per via del passaggio per valore, il valore di x non viene in alcun modo modificato

Passaggio dei Parametri

- Passaggio dei riferimenti
  - ⇨anche i riferimenti vengono passati ai metodi per valore (copia del valore dell'argomento nello spazio di memoria del parametro)
  - ⇨ma, siccome il metodo riceve una copia del riferimento all'oggetto, attraverso il riferimento può inviare messaggi all'oggetto e quindi modificarne lo stato
  - ⇨si ottiene l'effetto del passaggio per riferimento

Passaggio dei Parametri

- Esempio n.2: riferimenti
  - ⇨acquisizione delle circonferenze

```
public void esegui() {
    int numCirconferenze = schermoLeggiNumCirconferenze();
    if (numCirconferenze > 0) {
        Circonferenza[] coll = new Circonferenza[numCirconferenze];
        for (int i = 0; i < coll.length; i++) {
            coll[i] = schermoLeggiDatiCirconferenza();
        }
    ...
}
```

nella versione attuale del programma, viene utilizzata una funzione che costruisce ciascun oggetto e restituisce il riferimento

Passaggio dei Parametri

- Ma, in alternativa, avrei potuto usare

```
public void esegui() {
    int numCirconferenze = schermoLeggiNumCirconferenze();
    if (numCirconferenze > 0) {
        Circonferenza[] coll = new Circonferenza[numCirconferenze];
        for (int i = 0; i < coll.length; i++) {
            coll[i] = new Circonferenza();
            schermoLeggiDatiCirconferenza(coll[i]);
        }
    ...
}
```

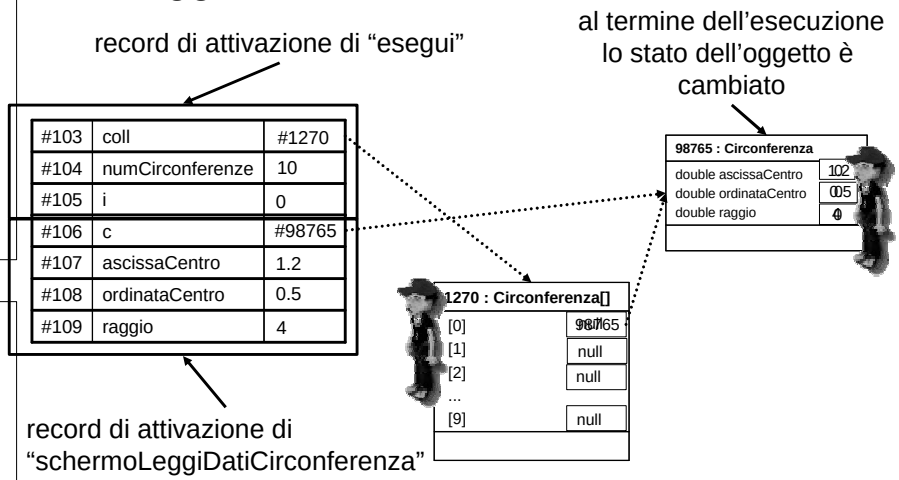
versione modificata del codice originale: voglio usare una procedura che riceve un riferimento alla circonferenza di cui leggere i valori

Passaggio dei Parametri

```
private void schermoLeggiDatiCirconferenza (Circonferenza c) {
    System.out.println("\nImmetti i dati della circonferenza:");
    System.out.print("Ascissa del centro: --> ");
    double ascissaCentro = it.unibas.utilita.Console.leggiDouble();
    c.setAscissaCentro(ascissaCentro);
    System.out.print("Ordinata del centro: --> ");
    double ordinataCentro = it.unibas.utilita.Console.leggiDouble();
    c.setOrdinataCentro(ordinataCentro);
    System.out.print("Lunghezza del raggio: --> ");
    double raggio = it.unibas.utilita.Console.leggiDouble();
    c.setRaggio(raggio);
}
```

la procedura utilizza il riferimento per chiamare metodi sull'oggetto e modificarne lo stato

Passaggio dei Parametri



Passaggio dei Parametri

- Nota
  - ⇨nel metodo schermoLeggiDatiCirconferenza sto usando la copia del riferimento per modificare lo stato dell'oggetto
- Ma
  - ⇨non potrei cambiare il valore del riferimento, che viene passato per valore

Passaggio dei Parametri

- Esempio n.2: scambio di circonferenze

```
public static void scambia (Circonferenza a, Circonferenza b) {
    Circonferenza x = a;
    a = b;
    b = x;
}
```

CODICE ERRATO:
sto cambiando i valori dei riferimenti nello stack, non lo stato degli oggetti nello heap

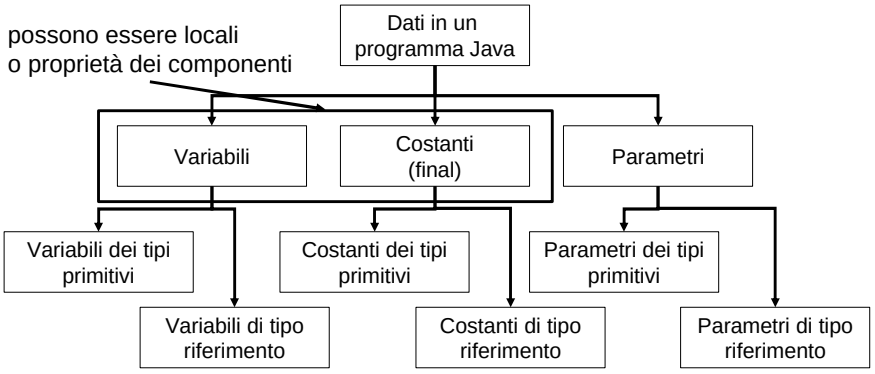
```
public static void main (String[ ] args) {
    Circonferenza a = new Circonferenza (1, 1, 2);
    Circonferenza b = new Circonferenza (1.3, 2.5, 5.2);
    scambia (a, b);
    System.out.println (a.getRaggio()); // stampa 2
}
```

Passaggio dei Parametri

- Riassumendo
  - ⇨Java prevede solo il passaggio per valore
  - ⇨non è possibile modificare un argomento di un tipo di base attraverso un parametro
  - ⇨non è possibile modificare un argomento di tipo riferimento attraverso un parametro
  - ⇨ma è possibile, usando i riferimenti, modificare lo stato di oggetti nello heap

Passaggio dei Parametri

- Riassumendo: dati in un programma Java



Riassumendo

- Riferimenti
- Confronti
- Il Riferimento this
- Passaggio dei Parametri

Termini della Licenza