Corso di Programmazione
Anno Accademico 2013–2014
Prof. Giovanni Pighizzini

Note ed esercizi aggiuntivi

Gli esercizi proposti sono utili per rivedere gli esempi riportati, che sono stati sviluppati e discussi in dettaglio a lezione.

5. Metodi statici, uso dei cicli

Esempio. Lettura di una sequenza di prezzi in lire e calcolo della somma dei prezzi equivalenti in euro. L'inserimento di `0` indica la fine della sequenza.

```
import prog.io.*;
import prog.utili.Importo;

class LireEuro {
  public  static void main(String[] args) {
    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    Importo sommaEuro = new Importo(0);

    //lettura ed elaborazione della sequenza
    int prezzoLire = in.readInt("Inserire un prezzo in lire (0 per terminare) ");
    while (prezzoLire != 0) {
      //calcola prezzo in euro
      Importo prezzoEuro = Importo.fromLire(prezzoLire);
      out.println("Euro : " + prezzoEuro);

      sommaEuro = sommaEuro.piu(prezzoEuro);

      prezzoLire = in.readInt("Inserire un prezzo in lire (0 per terminare) ");
    }

    //comunicazione risultato
    out.println("Somma prezzi in Euro = " + sommaEuro);
  }
}
```

Note
Prestate attenzione all'invocazione del metodo `fromLire`. *Tale metodo è statico.* Ciò significa che esso è un servizio offerto *direttamente* dalla classe `Importo`. Pertanto viene richiamato rivolgendosi direttamente alla classe, con l'invocazione

```
Importo.fromLire(...)
```

Sebbene il compilatore permetta di richiamare un metodo statico utilizzando il riferimento a un oggetto della classe, al posto del nome della classe stessa, tale scrittura *deve essere assolutamente evitata* in quanto l'oggetto non è in alcun modo coinvolto nell'esecuzione del metodo che, invece, è un compito della classe.

Esercizio 5.1
Modificate il codice dell'esempio precedente in modo che comunichi anche la somma dei prezzi in lire inseriti.

Esercizio 5.2
Scrivete un'applicazione che legga una sequenza di prezzi espressi in lire. Per ogni prezzo l'applicazione deve comunicare il corrispondente in euro. Supponete che l'inserimento di `0` indichi la fine della sequenza. L'applicazione deve poi comunicare:

- la somma dei prezzi in lire inseriti,
- la somma dei prezzi corrispondenti in euro e il suo corrispondente in lire.

A causa delle regole di arrotondamento nella trasformazione da lire in euro, le somme in lire ottenute ai due punti precedenti potrebbero differire. L'applicazione deve indicare se globalmente dall'arrotondamento si è ottenuto un guadagno o una perdita, fornendone l'ammontare in lire. Seguono due esempi di come dovrà essere l'esecuzione:

```
Inserire un prezzo (0 per terminare) 1500
Euro : 0,77
Inserire un prezzo (0 per terminare) 1500
Euro : 0,77
Inserire un prezzo (0 per terminare) 0
Somma prezzi in Lire = 3000
Somma prezzi in Euro = 1,54 - Corrispondente a Lire 2982
Perdita: Lire 18
```

```
Inserire un prezzo (0 per terminare) 2000
Euro : 1,03
Inserire un prezzo (0 per terminare) 4000
Euro : 2,07
Inserire un prezzo (0 per terminare) 0
Somma prezzi in Lire = 6000
Somma prezzi in Euro = 3,10 - Corrispondente a Lire 6002
Guadagno: Lire 2
```

Esercizio 5.3
Modificate l'applicazione costruita per l'esercizio 5.2 in modo che venga visualizzato anche il valore in euro corrispondente al guadagno o alla perdita in lire. Introducete poi ulteriori modifiche in modo che tutti i valori visualizzati vengano forniti anche in lettere, oltre che in cifre.

Esercizio 5.4
Scrivete un'applicazione che legga due numeri interi, ne calcoli la somma e la visualizzi incolonnata, nel formato presentato nei seguenti esempi di esecuzione:

```
Primo addendo? 54
Secondo addendo? 60
 54 +
 60 =
-----
114
```

```
Primo addendo? 1234567890
Secondo addendo? -987654321
1234567890 +
-987654321 =
------------
 246913569
```

Per la visualizzazione utilizzate gli usuali metodi `print` e `println`. L'incolonnamento può essere ottenuto calcolando il numero di cifre da cui è composto ciascun numero (cioè la lunghezza della stringa che rappresenta il numero), e facendo precedere il numero, se necessario, da spazi. La classe `String` offre un metodo *statico* di nome `valueOf` che riceve come argomento un `int` e restituisce il riferimento a un oggetto contenente la stringa corrispondente. Si può tenere conto del fatto che, in valore assoluto, il più grande intero rappresentabile è compreso tra due miliari e tre miliardi.

Esercizio 5.5
Modificate l'applicazione scritta per l'esercizio 5.4, in modo che le cifre vengano separate tre a tre da un punto, per evidenziare migliaia, milioni, ecc, come nel seguente esempio:

```
Primo addendo? 1234567890
Secondo addendo? -987654321
1.234.567.890 +
 -987.654.321 =
---------------
  246.913.569
```

Esercizio 5.6
Riscrivete l'applicazione dell'esercizio 5.4 in modo che sia in grado di trattare interi di grandezza arbitraria. A tale scopo, al posto del tipo primitivo `int`, rappresentate gli interi come oggetti della classe `java.math.BigInteger`, i cui oggetti rappresentano numeri interi di grandezza arbitraria. Consultate la documentazione per i dettagli.

Esempio. Calcolo del numero di lettere maiuscole in una stringa letta da input.

```
// VERSIONE 1: ciclo while

import prog.io.*;

class ContaMaiuscole {
  public  static void main(String[] args) {
    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    int nMaiuscole = 0;

    //lettura e calcolo
    String s = in.readLine("Inserisci una stringa ");

    int i = 0;
    int lung = s.length();
    while (i < lung) {
      char c = s.charAt(i);
      if (Character.isUpperCase(c))
        nMaiuscole = nMaiuscole + 1;
```

```
      i = i + 1;
    }

    //comunicazione risultato
    out.println("La stringa " + s + " contiene " +  nMaiuscole +
                " lettere maiuscole");
  }
}
```

Note

- Osservate che la condizione della selezione è il risultato fornito dal metodo statico `isUpperCase` di `Character`. L'invocazione dei metodi statici deve essere effettuata utilizzando il nome della classe che fornisce il metodo al posto del riferimento all'oggetto, utilizzato invece nel caso di invocazione dei metodi non statici.
- Osservate la struttura del ciclo, basato sulla variabile `i`, inizializzata a `0`, incrementata a ogni iterazione, sino a raggiungere il valore della lunghezza: in una situazione come questa è opportuno utilizzare un ciclo `for`, come mostrato nel codice seguente.

```
// VERSIONE 2: ciclo for
import prog.io.*;

class ContaMaiuscole {
  public  static void main(String[] args) {
    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    int nMaiuscole = 0;

    //lettura e calcolo
    String s = in.readLine("Inserisci una stringa ");

    for (int i = 0; i < s.length(); i = i + 1) {
      char c = s.charAt(i);
      if (Character.isUpperCase(c))
        nMaiuscole = nMaiuscole + 1;
    }

    //comunicazione risultato
    out.println("La stringa " + s + " contiene " +  nMaiuscole +
                " lettere maiuscole");
  }
}
```

Modifichiamo l'esempio in modo che conti il numero di lettere maiuscole in una sequenza di stringhe lette da input, anziché in una stringa sola. L'inserimento della stringa vuota indica la fine della sequenza.

```
import prog.io.*;

class ContaMaiuscole {
  public  static void main(String[] args) {
    ConsoleInputManager in = new ConsoleInputManager();
```

```
    ConsoleOutputManager out = new ConsoleOutputManager();

    int nMaiuscole = 0;

    //lettura e calcolo
    String s = in.readLine("Inserisci una stringa (<invio> per terminare) ");
    while (!s.equals("")) {
      //esame della stringa
      for (int i = 0; i < s.length(); i = i + 1) {
        char c = s.charAt(i);
        if (Character.isUpperCase(c))
          nMaiuscole = nMaiuscole + 1;
      }
      s = in.readLine("Inserisci una stringa (<invio> per terminare) ");
    }

    //comunicazione risultato
    out.println("Le stringhe lette contengono in totale " +  nMaiuscole +
                " lettere maiuscole");
  }
}
```

Note
Si osservi l'uso dei due cicli innestati: quello esterno per la lettura e l'esame delle stringhe, quello interno per l'esame di ciascuna stringa.

Esercizio 5.7
Modificate il codice dell'esempio precedente in modo che comunichi anche:

- il numero totale di stringhe lette,
- il numero medio di lettere maiuscole presenti nelle stringhe,
- la stringa, tra quelle lette, che contiene il maggior numero di lettere maiuscole.

Esercizio 5.8
Scrivete un'applicazione che legga un intero `n`, generi `n` numeri interi a caso, compresi tra `0` e `1000`, visualizzando ciascuno di essi in cifre, in lettere come numero cardinale, in lettere come numero ordinale. Se ad esempio viene letto `4`, una possibile esecuzione produrrà:

```
10 dieci decimo
8 otto ottavo
4 quattro quarto
1000 mille millesimo
```

Utilizzate la classe `Intero` del package `prog.utili` e la classe `Random` del package `java.util`.