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

Note ed esercizi aggiuntivi

10. Ereditarietà, file di testo

Esercizio 10.1
Scrivete un'applicazione che richieda all'utente di inserire i dati di un elenco di figure (rettangoli, quadrati, cerchi). L'inserimento può essere guidato mediante un menu. L'applicazione dovrà determinare l'area media delle figure inserite e produrre:

- Un elenco di tutte le figure la cui area è superiore alla media, nello stesso ordine in cui sono state inserite.
- Tre elenchi come il precedente, ma suddivisi per tipo di figura (cioè uno per i rettangoli, uno per i quadrati, uno per i cerchi).

Per svolgere l'esercizio sfruttate il meccanismo dell'ereditarietà (potete ispirarvi a un esempio simile presentato nel paragrafo 6.9 del libro di testo). Come cambierebbe la soluzione senza l'uso del tipo `Figura` e dell'ereditarietà?

Esempio. Conteggio del numero di lettere maiuscole presenti in una sequenza di stringhe fornita tramite tastiera. La fine della sequenza viene indicata inviando il segnale di *end-of-file*.

```
import prog.io.ConsoleInputManager;
import prog.io.ConsoleOutputManager;

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

    int nMaiuscole = 0;
    String s = in.readLine();
    while (s != null) {
      for (int i = 0; i < s.length(); i++)
        if (Character.isUpperCase(s.charAt(i)))
          nMaiuscole++;
      s = in.readLine();
    }
    out.println("Il testo contiene " + nMaiuscole + " lettere maiuscole");
  }
}
```

Note

- Nei sistemi Unix/Linux/Mac, l'*end-of-file* viene segnalato da tastiera premendo la combinazione di tasti `ctrl-d` all'inizio di una riga.
- Il metodo `readLine` della classe `ConsoleInputManager` restituisce il riferimento null quando riceve il segnale di *end-of-file* (ricordatevi che il riferimento `null` *non* è la stringa vuota!).

Esempio. Conteggio del numero di lettere maiuscole presenti in un file di testo il cui nome è fornito sulla linea di comando.

```
import prog.io.FileInputManager;
import prog.io.ConsoleOutputManager;

class ContaMaiuscole {
  public static void main(String[] args) {
    FileInputManager in = new FileInputManager(args[0]);
    ConsoleOutputManager out = new ConsoleOutputManager();

    int nMaiuscole = 0;
    String s = in.readLine();
    while (s != null) {
      for (int i = 0; i < s.length(); i++)
        if (Character.isUpperCase(s.charAt(i)))
          nMaiuscole++;
      s = in.readLine();
    }
    out.println("Il testo contiene " + nMaiuscole + " lettere maiuscole");
  }
}
```

Note

- L'applicazione legge il nome del file sulla riga di comando, ricevendolo tramite il parametro `args`. Pertanto l'invocazione sarà della forma

  ```
  java ContaMaiuscole <nomefile>
  ```

- Raggiunta la fine del file, il metodo `readLine` restituisce il riferimento `null`.

Esercizio 10.2
Eseguendo il codice dell'esempio precedente, se il nome del file non è specificato o se il file indicato non esiste, la Java Virtual Machine segnala un errore (provate a mandare in esecuzione l'applicazione per vedere il messaggio d'errore nei due casi). Modificate l'applicazione in modo che, nel caso si verifichi uno di questi due errori, segnali il problema con un messaggio comprensibile all'utente e termini l'esecuzione. Per prevenire il primo errore è sufficiente controllare la lunghezza dell'array riferito da `args`, per il secondo si può utilizzare un metodo fornito dalla classe `FileInputManager` (consultate la documentazione per scoprire come fare).

Esempio. Conteggio del numero di lettere maiuscole in file di testo (se il nome del file viene specificato sulla linea di comando) o nella sequenza di stringhe fornita da tastiera (se sulla riga di comando non vengono specificati argomenti).

```
import prog.io.InputManager;
import prog.io.FileInputManager;
import prog.io.ConsoleInputManager;
import prog.io.ConsoleOutputManager;

class ContaMaiuscole {
  public static void main(String[] args) {
    InputManager in = args.length > 0 ? new FileInputManager(args[0]) :
                                        new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    int nMaiuscole = 0;
    String s = in.readLine();
    while (s != null) {
      for (int i = 0; i < s.length(); i++)
        if (Character.isUpperCase(s.charAt(i)))
          nMaiuscole++;
      s = in.readLine();
    }
    out.println("Il testo contiene " + nMaiuscole + " lettere maiuscole");
  }
}
```

Note
Si osservi l'uso del supertipo `InputManager`, con il quale è possibile riferirsi sia a oggetti della classe `ConsoleInputManager` sia a oggetti della classe `FileInputManager`.

Esercizio 10.3
Modificate il codice precedente in modo che, in caso di lettura da tastiera, fornisca un messaggio preliminare per invitare l'utente a inserire il testo.

Esercizio 10.4
Modificate il codice precedente in modo che, nel caso l'utente fornisca il nome di un file inesistente, l'applicazione chieda all'utente se vuole terminare l'esecuzione o se vuole inserire il testo da analizzare direttamente da tastiera.

Esercizio 10.5
Scrivete un'applicazione `CopiaFile` che crei una copia di un file di testo. Se l'applicazione viene chiamata indicando due argomenti, nella forma

```
java CopiaFile <file1> <file2>
```

allora deve copiare il contenuto di `<file1>` in `<file2>` (se `<file2>` esiste già, il suo contenuto verrà cancellato). Se l'applicazione viene chiamata indicando un solo argomento, nella forma

```
java CopiaFile <file>
```

allora deve visualizzare il contenuto di `<file>` sul monitor. Infine, se l'applicazione viene chiamata senza indicare argomenti, nella forma

```
java CopiaFile
```

allora deve leggere righe da tastiera, riscrivendole man mano sul monitor. Ispiratevi agli esempi precedenti. Per la scrittura potete utilizzare le classi `FileOutputManager` e `ConsoleOutputManager` che implementano l'interfaccia `OutputManager`. Per saperne di più consultate la documentazione.

Esercizio 10.6
Modificate l'applicazione realizzata per l'esercizio 10.5 in modo che, nel caso di scrittura su un file già esistente, chieda all'utente se desidera cancellare il contenuto del file, se desidera scrivere in fondo al file, o se desidera rinunciare all'operazione. La classe `FileOutputManager` offre un costruttore che permette di specificare se si vuole sovrascrivere il file (cioè scriverci dopo avere cancellato il contenuto precedente) o se si vuole appendere al file (cioè scrivere alla fine). La classe `FileInputManager` fornisce un metodo statico per verificare l'esistenza di un file.

Esercizio 10.7
Scrivete un'applicazione `LeggiWeb` che visualizzi sul monitor il codice HTML di una pagina web remota. Scrivendo ad esempio

```
java LeggiWeb http://www.unimi.it
```

l'applicazione deve visualizzare il codice HTML della pagina `http://www.unimi.it`. L'esercizio è più semplice di quanto possa sembrare. È sufficiente ispirarsi alla soluzione data per l'esercizio 10.5, sapendo che le pagine HTML sono file di testo e che nel package `prog.io` c'è una classe di nome `WebPageInputManager` che implementa l'interfaccia `InputManager` e permette di leggere il codice sorgente di pagine web remote.

Esercizio 10.8
Scrivete un'applicazione `TrovaLink` che elenchi tutti i collegamenti (link) ad altre pagine che si trovano di una pagina HTML remota. Per la lettura della pagina HTML potete utilizzare quanto sviluppato per l'esercizio 10.7. Nel codice HTML i collegamenti sono individuati da speciali tag come

```
<A HREF="http://www.link.xx">testo</A>
```

La stringa tra virgolette dopo la parola `HREF` indica la pagina a cui il link conduce; il testo che viene visualizzato sulla pagina, tramite il quale è possibile seguire il collegamento con un click sul mouse, è delimitato da `<A HREF>` e `</A>` (`A` e `HREF` possono essere indifferentemente minuscoli o maiuscoli).

Esercizio 10.9
Scrivete altre applicazioni, come quella per l'esercizio 10.8, che estraggano informazioni da pagine HTML (ad esempio i titoli delle pagine, l'elenco delle figure presenti, l'elenco di tutte le parole in italico, ecc.). Se non conoscete il linguaggio HTML fate una ricerca in rete per scoprire quali sono i tag fondamentali.