LABORATORIO DI PROGRAMMAZIONE 2009–2010
EDIZIONE 2 – TURNI A E B
EDIZIONE 3 (SERALE)
PROVA D'ESAME
25.I.2010

VINCENZO MARRA

AVVERTENZA. *Non è ammesso l'uso delle classi del package* `prog.io` *allegato al libro di testo del corso.*

ESERCIZIO 1

La classe `VectorStringhe`.

Scrivete una classe di nome `VectorStringhe`, una cui istanza rappresenti un elenco di stringhe; sono ammesse le ripetizioni della stessa stringa, le stringhe vuote, e i riferimenti `null`. Dichiarate `VectorStringhe` sottotipo di `Vector<String>`. (La classe `Vector<E>` si trova in `java.util`). Dotate la classe dei due soli metodi seguenti.

- Un metodo di nome `numeroStringheNull`, di prototipo appropriato, che restituisca il numero di stringhe `null`[1] contenute nell'elenco al momento dell'invocazione del metodo.
- Un metodo di nome `numeroStringheVuote`, di prototipo appropriato, che restituisca il numero di stringhe vuote contenute nell'elenco al momento dell'invocazione del metodo.

ESERCIZIO 2

La classe `ListaStringhe`.

Scrivete una classe di nome `ListaStringhe`, una cui istanza rappresenti un elenco di stringhe; sono ammesse le stringhe vuote, e le ripetizioni della stessa stringa. Non sono invece ammessi nell'elenco riferimenti `null`. Dichiarate `ListaStringhe` sottotipo del tipo `VectorStringhe` implementato nell'Esercizio 1.

Ridefinite il metodo `public boolean add(E s)` ereditato da `Vector<E>` di modo che soddisfi le specifiche seguenti.

(1) Se il riferimento `s` di tipo `String` passato al metodo è `null`, il valore di `add(s)` è `false`, e l'effetto della chiamata è di lasciare invariato l'elenco di stringhe.

[1] *Errata corrige del 25 gennaio 2010.* La versione del testo distribuita all'esame riportava per errore 'non `null`' invece di '`null`'.

(2) Se il riferimento `s` di tipo `String` passato al metodo non è `null`, il valore di `add(s)` è `true`, e l'effetto della chiamata è di aggiungere `s` in coda all'elenco di stringhe.

Considerate infine l'opportunità di ridefinire tutti, alcuni, o nessuno dei due metodi di `VectorStringhe` implementati nell'Esercizio 1, e procedete di conseguenza.

Esercizio 3

La classe `InsiemeStringhe`.

Scrivete una classe di nome `InsiemeStringhe`, una cui istanza rappresenti un elenco di stringhe; sono ammesse le stringhe vuote. Non sono invece ammesse nell'elenco ripetizioni della stessa stringa, né sono ammessi riferimenti `null`. Dichiarate `InsiemeStringhe` sottotipo del tipo `ListaStringhe` implementato nell'Esercizio 2.

Ridefinite il metodo `public boolean add(E s)` ereditato da `Vector<E>`, già ridefinito in `ListaStringhe`, di modo che soddisfi le specifiche seguenti.

(1) Se la stringa passata al metodo è già contenuta nell'elenco, il metodo solleva un'eccezione non controllata di tipo `RipetizioneException`, che dovrete definire allo scopo.
(2) Altrimenti, il metodo si comporta come il metodo omonimo della superclasse.

Esercizio 4

La classe `Programma`.

Scrivete una classe di nome `Programma` contenente il solo metodo `main`. Il programma apre un file di testo specificato dall'utente, lo legge una riga alla volta, costruisce un elenco di stringhe contenente le righe lette, e visualizza in uscita l'elenco, con qualche informazione sulle righe ripetute e le righe vuote. Ecco le specifiche da seguire nel dettaglio.

Il programma accetta due argomenti passati dalla riga di comando, di cui il primo è obbligatorio, e il secondo facoltativo.

- Il primo argomento è il nome di un file di testo.
- Il secondo argomento può essere:
  - Assente, il che indica che è richiesta la costruzione di un elenco delle righe del file, incluse le righe ripetute o vuote.
  - L'opzione `-r`, il che indica che è richiesta la costruzione di un elenco delle righe del file, incluse le righe vuote, ma privo di righe ripetute.
- Se gli argomenti passati dalla riga di comando non sono conformi a queste specifiche, il programma termina con messaggi d'errore appropriati.
- Altrimenti, sia `nomefile` il nome del file specificato dall'utente. Allora:
  - Se `nomefile` non esiste, il programma termina con un messaggio d'errore appropriato.
  - Se `nomefile` esiste, ma è una directory, il programma termina con un messaggio d'errore appropriato.
  - Altrimenti, il programma procede.

- Il programma apre `nomefile`, costruisce un elenco delle sue righe (con o senza ripetizioni, a seconda dell'opzione specificata dall'utente), e visualizza in uscita l'elenco costruito. La visualizzazione deve essere fatta una riga per volta, con un ritorno a capo dopo ciascuna riga.
- Infine, il programma visualizza le informazioni seguenti, e termina.
  - Solo nel caso in cui l'utente specifichi l'opzione `-r`: Il numero di righe ripetute nel file.
  - In ogni caso: il numero di stringhe vuote nell'elenco. (*Nota Bene.* Nell'*elenco costruito dal programma*, non nel file originale.)
  - In ogni caso: il numero di stringhe `null` nell'elenco. (*Nota Bene.* Questo numero dovrebbe risultare sempre pari a zero, perchè in un file di testo una riga `null` indica la terminazione dello stream.)

Per esempio, si consideri il file di testo di nome `test.txt`, costituito da 8 righe, mostrato qui sotto:

```
Prima riga del file.
Un'altra riga del file.

Un'altra riga del file.
Un'altra riga del file.
Riga.
Riga.
Ultima riga del file.
```

L'esecuzione di `Programma test.txt` produce in uscita:

```
*** Elenco ***

Prima riga del file.
Un'altra riga del file.

Un'altra riga del file.
Un'altra riga del file.
Riga.
Riga.
Ultima riga del file.

*** Statistiche ***

Numero righe vuote nell'elenco:  1
Numero righe null nell'elenco:  0
```

L'esecuzione di `Programma test.txt -r` produce in uscita:

```
*** Elenco ***

Prima riga del file.
Un'altra riga del file.

Riga.
```

```
Ultima riga del file.

*** Statistiche ***

Numero righe ripetute nel file:  3
Numero righe vuote nell'elenco:  1
Numero righe null nell'elenco:  0
```

Classi da consegnare

(1) `VectorStringhe`
(2) `ListaStringhe`
(3) `InsiemeStringhe`
(4) `RipetizioneException`
(5) `Programma`

Per la consegna, eseguite l'upload dei singoli file sorgente dalla pagina web: `http://upload.dico.unimi.it`.

(V. Marra) Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano, via Comelico, 39-41, I-20135 Milan, Italy
E-mail address: marra@dico.unimi.it