Lab Lesson 08

Giacomo Bergami

March 20, 2018

Esercizio

Si vuole gestire una community online di valutazione di film. La community è composta da utenti, identificati da un nome utente e provvisti di e-mail. Ogni utente effettua delle valutazioni su di un film, indicandone un nome ed un punteggio compreso tra zero e 10. Ogni film, contraddistinto da un unico nome, ha un genere, una data di rilascio, ed un attore principale. Di ogni attore, contraddistinto univocamente da nome e cognome, si vuole indicarne il sesso.

Ogni database può essere salvato su disco su un unico file di testo (`:dump`), e caricato in memoria (`:load`). Il formato con cui si vuole salvare e caricare il database in memoria è il seguente:

```
actor(Natalie,Portman,female)
movie(Free Zone,09/06/2005,Drama,Natalie Portman)
user(giangiotto23,giangiotto@studio.unibo.it)
score(giangiotto23,Free Zone,6)
```

Ogni riga corrisponde ad un oggetto di una classe corrispondente al nome dell'entità (es. `actor`, `movie`, ...). In particolare, il programma principale è una classe `Lez09.imdb.simple.Driver` che attende che l'utente fornisca i seguenti comandi:

- `:quit`

  esce dal programma driver.

- `:load` *nome.file*

  carica il database in memoria nel formato dato precedentemente.

- `:dump` *nome.file*

  salva il database nel modo specificato.

- `:add` *classe(argomento*$_1$*,...,argomento*$_n$*)*

  Se non esiste, crea un nuovo oggetto di tipo *classe* fornendo gli argomenti sopracitati al costruttore.

- `?mostProlificActor`

  Stampa in una nuova linea il nome di un attore/un'attrice che ha recitato in un numero di film maggiore (o uguale) ai suoi colleghi.

- `?noMoviesForGenre` *genere*

  Stampa in una nuova linea il numero di film che sono stati rilasciati di un determinato *genere*.

- `?genreWithHighestAverage`

  Dopo aver calcolato la media di tutte le valutazioni di film raggruppandole per genere, restituisce uno dei generi con media massima.

Per lo svolgimento dell'esercizio, si considera una soluzione ammissibile la rappresentazione del campo di ogni classe come una stringa.

Auto-Valutazione

Questo esercizio effettua il testing prendendo come input i files forniti nella cartella *tests*. In ogni cartella esiste un file di risposte attese *answer.txt*, un file *query.txt* di comandi da fornire al drive, e un file *db.txt* contentente il risultato finale atteso dell dump del database. In questo modo, posso fare in modo che alcuni field della classe siano oggetti effettivamente corrispondenti all'entità. L'oggetto `Lez09.imdb.simple.test.TestCorrectIMDB` utilizza automaticamente questi files e la classe driver, controllando che i risultati stampati e salvati su disco corrispondanto a quelli attesi.

Curiosità

1. Si consiglia il libro "How To Design Classes" di Felleisen per la progettazione orientata agli oggetti. Questo libro è liberamente disponibile in `http://www.ccs.neu.edu/home/matthias/HtDC/htdc.pdf`.

2. Il package `Lez09.imdb.data_generator` contiene il codice per mezzo del quale sono stati generati i files di test.

3. Il package `Lez09.imdb.better` contiene una soluzione completa ma più complicata, dove la creazione degli oggetti viene posticipata finché tutte le informazioni necessarie non vengono lette dal file del database. Questa soluzione non è richiesta ai fini dell'esercitazione, ma viene fornita come esempio di studio. La soluzione proposta utilizza un "design pattern" definito come *observer*: `https://it.wikipedia.org/wiki/Observer_pattern`.

   Per velocizzare il caricamento degli oggetti da file e per una pratica di buona programmazione, vengono utilizzate le `HashMap` invece degli `ArrayList`.

4. Le ultime versioni di Java tendono ad eliminare la gestione delle eccezioni per i casi di errore, e tendono di più ad una programmazione "state driven". Si fornisce un esempio avanzato per la gestione dei file in `https://github.com/jackbergus/LucenePdfIndexer`.