Metodologie di Programmazione - Canale P-Z
A.A. 2008/09 - Appello del 16/06/2009 - Prima parte

Avvertenze Le soluzioni devono essere scritte in modo chiaro e ordinato. Il codice deve essere indentato per facilitarne la lettura. Si possono consultare libri e appunti ma non è permesso usare computer.

Ogni esercizio o parte di esercizio ha un punteggio massimo. Il voto (in trentesimi) sarà determinato dalla somma dei punteggi ottenuti nei vari esercizi. (Notare che la somma dei punti degli esercizi proposti supera 30. Quindi, per ottenere il massimo non è necessario svolgere e ottenere il massimo dei punti in tutti gli esercizi.)

Esercizio 1 [8 punti]

Definire una classe `AVItem` che contiene le informazioni di base relative a un documento audio o video. Le informazioni che ogni classe mantiene sono: il titolo, la durata in secondi, l'autore e un commento (una stringa). Il commento può essere omesso ed è modificabile. Gli altri campi sono immutabili e obbligatori, tranne il campo autore che può essere omesso. **[4 punti]**

I costruttori della classe non devono essere accessibili a tutti, ma devono essere visibili alle sottoclassi.

I campi di istanza devono essere accessibili solo attraverso opportuni metodi getters o setters (scrivere solo quelli richesti dalle specifiche).

Riscrivere il metodo `toString` in modo che restituisca una stringa contenente tutti i dati memorizzati in un'istanza.

Definire due sottoclassi `AudioItem` e `VideoItem` che corrispondono rispettivamente a un documento audio e video. `AudioItem` contiene come informazione aggiuntiva il campo `esecutore`, immutabile ma non obbligatorio. Invece, `VideoItem` contiene come informazione aggiuntiva il campo `regista`, immutabile ma non obbligatorio. Tutte le classi devono poter creare istanze di `AudioItem` e `VideoItem`. Scrivere i metodi che completano queste due classi, in particolare estendere anche `toString` in modo di aggiungere le informazioni di queste classi a quelle della superclasse. **[4 punti]**

Esercizio 2 [22 punti]

Scrivere una classe `AVList` per la memorizzazione di una sequenza di elementi audio video, ovvero di `AVList`. Ogni istanza di questa classe deve mantenere una informazione relativa alla durata complessiva degli elementi nella lista, un titolo e un commento. Questi campi non devono essere accessibili direttamente dall'esterno della classe. Scegliere quali argomenti rendere immutabili e obbligatori e definire i metodi getters/setters necessari (tenendo anche conto delle altre specifiche che seguono). **[2 punti]**

In ogni caso, deve essere possibile aggiungere o cancellare elementi dalla lista. Si devono quindi prevedere le operazioni di inserimento in coda, di inserimento in una posizione data, di cancellazione del primo elemento della lista e di cancellazione di un elemento in una posizione data. **[5 punti]**

Una istanza di `AVList` può essere creata con una lista vuota, oppure a partire da una lista di elementi di tipo `AVList` (di cui non è fornita la durata complessiva) da copiare nella lista memorizzata in `AVList` (la classe `AVList` non deve dipendere da elementi non creati al suo interno). **[5 punti]**

Scrivere due sottoclassi della `AVList`:

1. la classe `SongList` che contiene solo `AudioItem`. Riscrivere ove necessario i costruttori e i metodi ereditati dalla superclasse (utilizzandoli comunque nelle nuove definizioni) per garantire che effettivamente al momento della costruzione delle istanze e dell'inserimento di nuovi item, le istanze di `SongList` contengono solo `AudioItem`. **[5 punti]**
2. la classe `Film` che contiene solo `VideoItem`. Questa classe prevede il campo aggiuntivo `regista`, il cui valore, se diverso da `null`, deve essere uguale al valore del campo regista di tutti i `VideoItem` contenuti nella lista. Riscrivere ove necessario i costruttori ereditati dalla superclasse (utilizzandoli comunque nelle nuove definizioni) per garantire che effettivamente le istanze di `Film` contengono solo liste di item del tipo corretto. Inoltre, per questa classe la lista dei `VideoItem` deve essere inizializzata al momento della creazione dell'istanza e non può essere successivamente modificata. A tale fine, si riscrivano tutti i metodi di inserimento e cancellazione ereditati dalla superclasse in modo che vengano semplicemente ignorate le corrispondenti operazioni. **[5 punti]**

Esercizio 3 [10 punti]

Scrivere un metodo che ha come parametri il nome di un file sorgente, il nome di un file destinazione e una stringa e copia il file sorgente (che supponiamo essere di testo) nel file destinazione eliminando tutte le parole presenti nella stringa.