Appello di Settembre 2009	**TV Serial Cataloger**
	Laurea Triennale in Comunicazione Digitale Laboratorio di Programmazione

1 Descrizione

Il progetto consiste nel realizzare un programma per gestire la catalogazione degli episodi di serial televisivi e conseguentemente la ricerca all'interno del catagolo costruito. Per chi non lo sapesse i serial televisivi (come lost, battlestar galactica, e star trek) sono suddivisi in stagioni ed ogni stagione è composta da diversi episodi.

Le informazioni che considereremo per la catalogazione dei vari episodi sono: titolo del serial di appartenenza, stagione, titolo dell'episodio da catalogare, numero dell'episodio, anno della prima trasmissione e genere del serial.

2 Struttura del Catalogo

Il catalogo degli episodi deve godere delle seguenti proprietà:

- deve essere permanente, cioè quando si esce dal programma il catalogo non viene perso,
- deve essere ordinabile, su richiesta, per serial, genere, numero di episodio e titolo degli episodi;

Su disco ogni singola riga del catalogo si riferisce ad un episodio e conterrà i seguenti dati (rigorosamente in quest'ordine e tutti in formato testuale):

- una stringa, rappresentante il titolo del serial;
- una stringa, rappresentante il titolo dell'episodio;
- un numero rappresentante la stagione di cui è parte l'episodio;
- un intero compreso tra 1900 e 2100, rappresentante l'anno in cui è stato trasmesso per la prima volta l'episodio;
- un intero compreso tra 1 e 1000, rappresentante il numero dell'episodio nella continuità del serial; e

- una stringa, rappresentante il genere del serial di appartenenza dell'episodio.

Le singole informazioni contenute in una generica riga del file sono separate tra loro dal carattere di barra verticale (`|`).
La seguente riga è un esempio di riga che può essere contenuta nel file:

```
Lost|Exposé|3|2007|63|Drama
```

Notate che, potenzialmente, le stringhe presenti sul file non hanno limite in lunghezza e non viene usata nessuna convenzione per i titoli.

3 Classi da Realizzare

È obbligatorio realizzare in Java il programma descritto nelle sezioni precedenti utilizzando le seguenti classi:

Episode

Descrive un episodio del serial in base alle informazioni che lo caratterizzano: *titolo, titolo serial, stagione, numero, anno* e *genere*. La classe implementa l'interfaccia `Comparable` per semplificare l'ordinamento del catalogo.

Attributi

- `String title`: variabile di istanza contenente il titolo dell'episodio;
- `String serial`: variabile di istanza contenente il titolo del serial di cui è parte l'episodio;
- **`int`** `season`: variabile di istanza contenente la stagione di appartenenza dell'episodio;
- **`int`** `epn`: variabile di istanza contenente il numero sequenziale dell'episodio;
- **`int`** `year`: variabile di istanza contenente l'anno di messa in onda dell'episodio;
- `String genre`: variabile di istanza contenente il riferimento al genere del serial di cui è parte l'episodio, ad esempio: drama, sci-fi, sit-com, anime e così via.

Notare che gli attributi della classe `Episode` rappresentano lo stato delle sue istanze e dovranno essere definite coerentemente coi dettami di buona programmazione insegnati (vedi *information hiding*) con tutto ciò che questo ne comporta. **L'osservazione è valida per tutte le classi anche se non ripetuta.**

Metodi e Costruttori Pubblici

- `Episode(String t, String ts,` **`int`** `s,` **`int`** `n,` **`int`** `y, String g)`: costruttore che interpreta i parametri come le informazioni caratterizzanti un episodio;

- `String toString()`: metodo che crea e ritorna una stringa rappresentante l'episodio;
- **`int`** `compareTo(Episode a)`: metodo che confronta l'episodio corrente (**`this`**) con l'episodio passato come parametro (`a`). Il confronto è basato su: titolo del serial, titolo dell'episodio, stagione e numero dell'episodio; perciò i due episodi sono uguali ed il metodo ritorna zero se e solo se hanno lo stesso titolo, appartengono allo stesso serial, hanno lo stesso numero progressivo e sono tratti dalla stessa stagione. Per il resto il confronto è lessicografico (similarmente al dizionario), se i due episodi appartengono allo stesso serial si passa a confrontare la stagione, se anche questo è uguale si passa a confrontare il numero progressivo ed infine il titolo dell'episodio, nel momento in cui uno degli attributi risulta diverso il metodo ritornerà -1 se l'attributo di **`this`** è minore del corrispondente attributo di `a`, 1 altrimenti.

  Ad esempio, considerando il due episodi:

  ```
  Battlestar Galactica|Kobol's Last Gleaming: Part 1|1|2004|12|Sci-Fi
  Battlestar   GALACTICA|KOBOL's  Last Gleaming: Part 1|1|2005|12|SCI-FI
  ```

  il primo è rappresentato dall'oggetto **`this`** mentre il secondo è rappresentato dall'oggetto `a`, allora la chiamata:

  ```
  this.compareTo(a);
  ```

  ritornerà 0.

  Notare che il confronto deve essere fatto trascurando il numero di spazi che separano le parole che compongono i vari titoli e indipendentemente dall'uso di minuscole e maiuscole nelle parole stesse.

DB

La classe DB rappresenta il cuore del catalogo e dovrà permettere sia la costruzione del catalogo, che la sua manutenzione ed interrogazione. Gli episodi vengono aggiunti in testa al catalogo, senza mantenere un ordine, gli elementi già presenti nel catalogo vengono spostati in avanti di una posizione.

Attributi

- **`int`** `nepisodes`: variabile di istanza contenente il numero di episodi presenti in catalogo;
- `Episode[] episodes`: variabile di istanza contenente gli episodi presenti in catalogo, `episodes` è realizzato tramite un array ed avrà dimensione fissata pari a 1000 elementi;
- `String filename`: variabile di istanza contenente il nome del file su cui viene salvato e da cui viene ripristinato il catalogo.

Metodi e Costruttori Pubblici

- `DB(String fn)`: costruttore che inizializza il catalogo, `fn` è il nome del file su disco contenente il catalogo, se `fn` non è vuoto il contenuto del file corrispondente viene usato per inizializzare lo stato del catalogo (le variabili di istanza `nepisodes` e `episodes`);
- `String toString()`: metodo che crea e ritorna una stringa contenente l'intero catalogo con lo stesso formato usato in input;
- **`void`** `write()`: metodo che scrive il contenuto del catalogo nel file corrispondente, cancellando il contenuto precedente;
- **`void`** `reload()`: metodo che ricarica dal file i contenuti del catalogo nell'oggetto;
- `Object clone()`: metodo che crea una copia dell'oggetto e la ritorna;
- **`void`** `backup(String)`: metodo che effettua una copia di backup del catalogo, salvandola in un file il cui nome è specificato nella stringa passata come argomento;
- `Episode[] query(Query q)`: metodo che estrae dal catalogo tutti gli episodi che verificano la ricerca `q`; il metodo sfrutta il *late binding* per realizzare il confronto (vedi metodo `match(Episode)` della classe `Query` introdotta successivamente). Il risultato è un array contente gli episodi selezionati;
- **`void`** `squeeze()`: metodo pubblico che comprime il catalogo eliminando gli episodi potenzialmente doppi cioè quegli episodi che hanno lo stesso titolo e sono tratti dalla stessa stagione dello stesso serial, al solito trascurando spazi e differenze maiuscolo/minuscolo; ad esempio, gli episodi:

  ```
  Friends|The One Where Joey Speaks French|10|2004|231|Sit-Com
  Friends|  The One Where JOEY  Speaks French|10|2005|230|Sit-Com
  ```

  sono da considerarsi lo stesso episodio (notare che il numero progressivo, l'anno di messa in onda ed il genere sono ininfluenti), il primo episodio incontrato deve essere mantenuto in catalogo;
- **`void`** `sort()`: metodo che ordina il catalogo per serial, stagione e titolo episodio.

  Notare che per definizione `Episode` implementa l'interfaccia `Comparable` è pertanto possibile sfruttare i metodi della classe `Arrays` per ordinare in modo agevole il catalogo.
- **`void`** `squeezeSpaces()`: metodo che permette di eliminare spazi e tabulazioni multipli (cioè adiacenti), rimpiazzare i caratteri di tabulazione con uno spazio, e rimuovere gli spazi inutili all'inizio o alla fine dei titoli di serial ed episodi presenti in catalogo.

- **void** `capitalizeTitles()`: metodo che formatta i titoli degli episodi e dei serial in catalogo in modo che ogni parola abbia l'iniziale (e solo l'iniziale) maiuscola.
- **void** `addEpisode(Episode e)`: metodo che aggiunge al catalogo presente in memoria l'episodio specificato come parametro, SENZA aggiornare la versione salvata su disco. Il nuovo brano va inserito all'inizio del catalogo senza rispettare un eventuale ordinamento del catalogo stesso; l'inserimento comporterà lo spostamento dei dati contenuti nel catalogo;
- **void** `removeEpisode(Episode e)`: metodo che cerca e rimuove dal catalogo tutte le occorrenze dell'episodio passato come parametro; l'uguaglianza tra l'episodio passato come parametro e gli episodi contenuti nel catalogo è verificata con gli stessi criteri usati per il metodo `squeeze()`.

Query

La classe astratta `Query` rappresenta una generica ricerca all'interno del catalogo. Fornisce la struttura di base per richieste polimorfe sul catalogo.

- **abstract boolean** `match(Episode e)`: metodo astratto che verifica che l'episodio `e` rispetti i criteri di ricerca selezionati.

SerialQuery

La classe `SerialQuery` estende la classe `Query`. Rappresenta la ricerca per gli episodi in catalogo dello stesso serial.

- `String serial`: variabile di istanza contenente il nome del serial di cui si vuole estrarre gli episodi dal catalogo.
- `SerialQuery(String s)`: costruttore della query, si limita ad inizializzare i campi della classe.
- **boolean** `match(Episode e)`: controlla se l'episodio `e` è parte del serial specificato dalla variabile di istanza `serial`; ritornerà **true** se è effettivamente è parte del serial cercato, **false** altrimenti.

TitleSubstringQuery

La classe `TitleSubstringQuery` estende la classe `Query`. Rappresenta la ricerca, in catalogo, degli episodi i cui titoli contengono la stringa specificata. Ad esempio, dato il catalogo:

```
Lost|Exposé|3|2007|63|Drama
Battlestar Galactica|Kobol's Last Gleaming: Part 1|1|2004|12|Sci-Fi
Friends|The One Where Joey Speaks French|10|2004|231|Sit-Com
Numb3ers|Where Credit's Due|6|2009|106|Crime Investigation
NCIS|Trojan Horse|4|2007|93|Crime Investigation
```

se definiamo una query per ricercare gli episodi i cui titoli contengono la parola `"Where"` otterremo:

```
Friends|The One Where Joey Speaks French|10|2004|231|Sit-Com
Numb3ers|Where Credit's Due|6|2009|106|Crime Investigation
```

Fate attenzione che la stringa deve essere trovata indipendentemente dagli spazi nel titolo, dalle differenze maiuscolo/minuscolo ed ovviamente dal fatto che sia parte di una stringa più grande.

- `String substring`: variabile di istanza contenente la stringa che si vuole cercare nei titoli degli episodi presenti nel catalogo.
- `TitleSubstringQuery(String s)`: costruttore della query, si limita ad inizializzare, di conseguenza, i campi della classe.
- **`boolean`** `match(Episode e)`: controlla se la stringa contenuta nella variabile di istanza `substring` è presente nel titolo dell'episodio `e`; ritornerà **`true`** se presente, **`false`** altrimenti.

BeforeYearQuery

La classe `BeforeYearQuery` estende la classe `Query`. Rappresenta la ricerca, in catalogo, degli episodi mandati in onda prima dell'anno specificato indipendentemente dal serial.

- **`int`** `year`: variabile di istanza contenente l'anno interessato dalla ricerca.
- `BeforeYearQuery(`**`int`** `y)`: costruttore della query, si limita ad inizializzare, di conseguenza, i campi della classe.
- **`boolean`** `match(Episode e)`: controlla se l'episodio `e` è stato mandato in onda prima dell'anno specificato dalla variabile di istanza `year`; ritornerà **`true`** se è effettivamente così, **`false`** altrimenti.

Eccezioni.

Attenzione che i vari metodi potrebbero sollevare delle eccezioni; queste non sono state specificate nel testo ma dovranno essere gestite *cum grano salis*.

Tutte le eccezioni previste dall'uso di metodi Java devono filtrare ed essere gestite nel metodo `main()` anche se non espressamente indicato dalla segnatura dei metodi introdotti in questo documento.

A parte quanto espressamente richiesto, è lasciata piena libertà sull'implementazione delle singole classi e sull'eventuale introduzione di altre classi, a patto di seguire le regole del paradigma ad oggetti ed i principi di buona programmazione.

Non è richiesto e non verrà valutato l'utilizzo di particolari modalità grafiche di visualizzazione: è sufficiente una qualunque modalità di visualizzazione basata sull'uso dei caratteri.

È invece **espressamente richiesto** di non utilizzare package non standard di Java (si possono quindi utilizzare `java.util`, `java.io` e così via) e di rispettare l'interfaccia fornita. **Nota.** la libreria `prog` non fa parte delle librerie standard pertanto non ne è concesso l'utilizzo.

4 Modalità di Consegna

Il progetto deve essere svolto a gruppi di al massimo tre persone che intendono sostenere l'intero esame di Fondamenti di Architettura e Programmazione - Laboratorio di Programmazione nell'appello di settembre 2009, e deve essere consegnato **entro la mezzanotte tra domenica 20 e lunedì 21 settembre 2009** tramite il sito di sottoposizione:

`http://homes.dsi.unimi.it/~malchiod/LP/sottoposizione`

Per sottomettere un progetto dovrete registrarvi e creare un gruppo a cui affiliare voi ed i vostri compagni di gruppo. La sottoposizione è fatta a nome del gruppo e vale per ognuno dei componenti del gruppo stesso. Nota, sottoposizioni successive cancellano le sottoposizioni già fatte.

Dovranno essere consegnati tutti i sorgenti Java che permettano al programma di essere eseguito correttamente, compressi in un archivio di tipo ZIP che estragga i file nella directory in cui si trova l'archivio stesso. Nell'archivio dovrà anche essere accluso un breve documento in formato txt o rtf in cui:

- verrà descritto il modo in cui interfacciarsi con il programma;
- saranno illustrate le principali scelte implementative e le strategie utilizzate per svolgere il progetto

Le sottoposizioni che non seguono queste specifiche ed i programmi che non compilano correttamente non verranno valutati.

È inoltre richiesto di consegnare, **entro lunedì 21 settembre**, una copia cartacea della stampa del codice sorgente prodotto e della relazione in portineria del DSI/DICo indicando chiaramente nome, cognome e numero di matricola di tutti i componenti del gruppo, nonché il turno e il docente di riferimento.

5 Valutazione

Durante la prova orale con i singoli studenti saranno discusse le modalità implementative adottate e la padronanza di alcuni dei concetti necessari per preparare il progetto e/o spiegati a lezione. La valutazione del progetto sarà fatta in base alla:

- conformità dell'implementazione scelta per risolvere il problema con il paradigma di programmazione a oggetti;

- conformità del codice presentato alle regole di buona programmazione;
- adeguatezza della documentazione consegnata;
- assenza di errori nel programma;

Walter Cazzola
Dipartimento di Informatica e Comunicazione
Via Comelico 39/41 20135 Milano
Stanza P121 | Tel. +39.02.503.16300
e-mail: cazzola@dico.unimi.it

Dario Malchiodi
Dipartimento di Scienze dell'Informazione
Via Comelico 39/41 20135 Milano
Stanza S238 | Tel. +39.02.503.16338
e-mail: malchiodi@dsi.unimi.it