Dipartimento di Informatica Sistemistica e Comunicazione
Corso di Laurea in Informatica

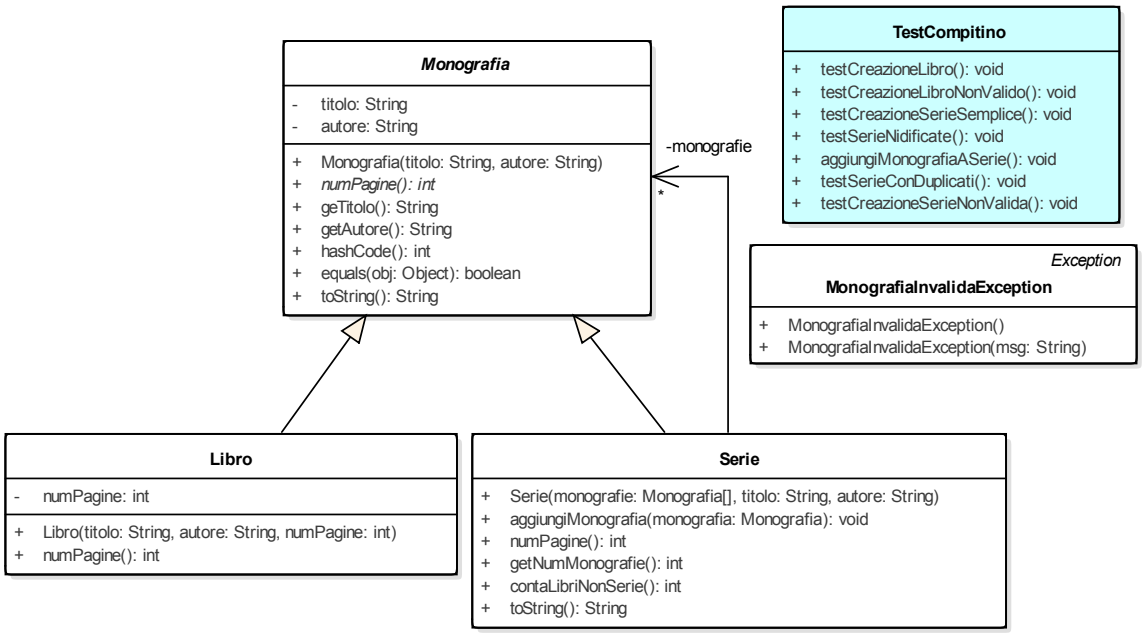
Programmazione 2

24 Giugno 2016 – Secondo Compitino

Testo parte di pratica

Una `Monografia` è un'opera letteraria caratterizzata da un titolo, un autore, e un certo numero di pagine. La `Monografia` può essere di due tipi (specializzazione): un `Libro` e una `Serie`. Un `Libro` è una `Monografia` consistente di un unico volume. Una `Serie` è una `Monografia` che può essere composta da singoli libri (oggetti di tipo `Libro`) e/o da altre serie (oggetti di tipo `Serie`). Ciò significa che una `Serie` consiste in un insieme di `Monografie`. Ad esempio la `Serie` "La terra di Mezzo" di Tolkien consiste di due `Monografie`: "Lo Hobbit" che è un `Libro` e da "Il Signore degli Anelli" che è una `Serie` composta dai volumi "La Compagnia dell'Anello", "Le due torri" e "Il ritorno del re", ciascuno dei quali è un oggetto di tipo `Libro`.

Implementare le classi come rappresentate dal seguente diagramma UML.



Infine, la classe `TestCompitino` (già fornita) contiene un insieme di test che devono essere fatti girare di volta in volta in modo da verificare la corretta realizzazione del software. **Come requisito minimo per ottenere una valutazione positiva, lo studente deve garantire che la sua implementazione non presenti errori di compilazione e superi almeno 3 casi di test fra quelli dati.**

Classe `Monografia`:

- ✓ Rappresenta una monografia. È una classe astratta caratterizzata da un titolo (attributo `titolo` e metodo getter `getTitolo()`), un autore (attributo `autore` e metodo getter `getAutore()`) e il numero di pagine (metodo astratto `numPagine()`)
- ✓ Il costruttore accetta due parametri `titolo` e `autore` e produce una eccezione di tipo `MonografiaInvalidaException` nel caso in cui uno dei due parametri sia `null` o uguale alla stringa vuota.
- ✓ Il metodo `equals()` confronta due monografie comparando solo il valore degli attributi `titolo` e `autore`, indipendentemente dal fatto che si tratti di un libro o di una serie. Ad esempio, il libro 'Il Signore degli Anelli' di Tolkien è uguale alla serie 'Il Signore degli Anelli' di Tolkien. Il metodo `hashcode()` deve essere coerente con il metodo `equals()`.
- ✓ Il metodo `toString()` stampa autore, titolo e numero delle pagine della monografia.

Classe `Libro`:

- ✓ rappresenta un libro. E' una classe concreta che estende `Monografia`. Definisce un attributo `numPagine` che viene assegnato nel costruttore.
- ✓ Il costruttore accetta tre parametri `titolo`, `autore` e `numPagine` e produce una eccezione di tipo `MonografiaInvalidaException` nel caso in cui `titolo` oppure `autore` siano `null` o uguale alla stringa vuota, oppure nel caso in cui `numPagine` sia minore o uguale a 0.
- ✓ Implementa il metodo `numPagine()` che ritorna il valore dell'attributo `numPagine`.

Classe `Serie`:

- ✓ Rappresenta un insieme di monografie. E' una classe concreta che estende `Monografia`. E' caratterizzata da un insieme (`HashSet`) di monografie memorizzato nell'attributo `monografie`.
- ✓ Il costruttore accetta tre parametri: un array di monografie, l'autore e il titolo della serie. Produce una eccezione di tipo `MonografiaInvalidaException` nel caso in cui `titolo` oppure `autore` siano `null` o uguale alla stringa vuota, oppure nel caso in cui l'array è `null`. Il costruttore memorizza nell'attributo `monografie` le monografie contenute nell'array passato come parametro. Tutte le monografie contenute nell'array devono avere lo stesso autore che deve essere coincidente con il valore dell'autore passato come parametro del costruttore. Se questa condizione non viene rispettata il costruttore produce l'eccezione `MonografiaInvalidaException`. Si intende l'array completamente riempito, cioè l'array contiene solo oggetti di tipo `Monografia` e non può contenere elementi di valore `null`.
- ✓ Il metodo `aggiungiMonografia(Monografia monografia)` aggiunge la monografia passata come parametro all'insieme di monografie della serie. L'autore della monografia aggiunta deve coincidere con l'autore della serie. Se questo non accade, il metodo produce l'eccezione `MonografiaInvalidaException`.
- ✓ Il metodo `numPagine()` ritorna il numero delle pagine calcolato sommando il numero delle pagine di tutte le monografie che compongono la serie.
- ✓ Il metodo `getNumMonografie()` ritorna il numero di monografie che compongono la serie.
- ✓ Il metodo `contaLibriNonSerie()` ritorna il numero di monografie che sono semplici libri (tipo dinamico `Libro`) e che fanno parte della serie.
- ✓ Il metodo `toString()` stampa autore, titolo e numero delle pagine della monografia. Stampa inoltre le medesime informazioni per tutte le monografie che compongono la serie.

Classe `MonografiaInvalidaException`:

- ✓ Rappresenta una eccezione che può essere sollevata dal programma quando i dati che sono utilizzati per creare una monografia non sono validi.
- ✓ Implementare il costruttore senza parametri e il costruttore con parametro di tipo stringa.