Dipartimento di Informatica Sistemistica e Comunicazione
Corso di Laurea in Informatica

Programmazione 2

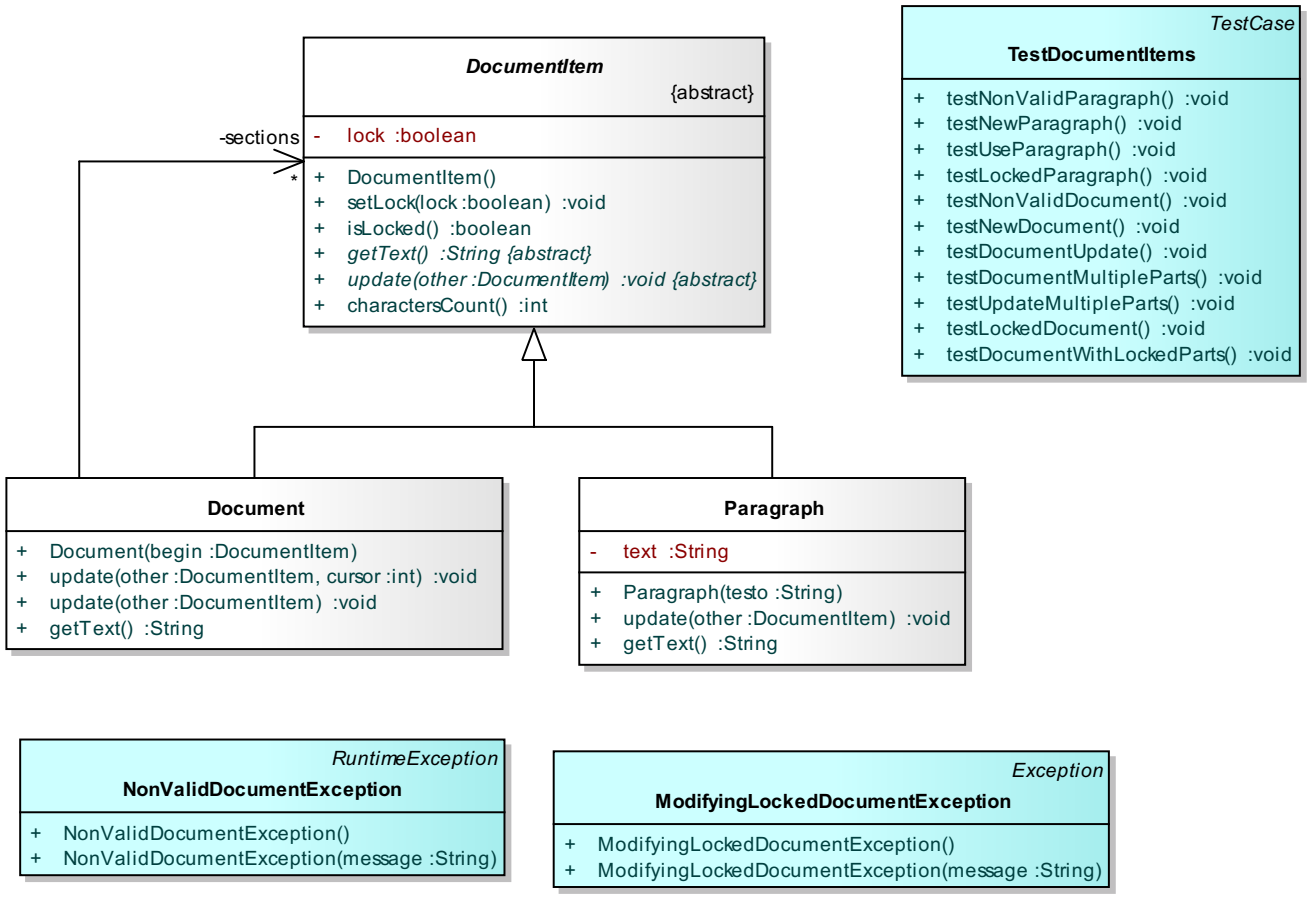
30 Giugno 2014 – Recupero Secondo compitino

Testo parte di pratica

Si realizzino le classi per la gestione di documenti di testo come da diagramma UML e specifiche che seguono. Si provino le classi realizzate con JUnit utilizzando i test forniti nella classe `TestDocumentItem`.

Nota: Gli elaborati che non superino almeno 5 test fra quelli dati saranno considerati insufficienti.

(Suggerimento: si eviti di eseguire i test solo alla fine del lavoro, quando ormai sarebbe tardi per apportare correzioni; Piuttosto si eseguano i test man mano che si procede con l'implementazione, per verificare incrementalmente il lavoro via via fatto.)



Classe `DocumentItem`:

- È una classe astratta che generalizza i possibili tipi di elementi che possono essere inclusi in un documento.
- Si massimi il riuso di codice, realizzando nella classe astratta tutte le caratteristiche in comune fra i possibili sotto tipi.
- Si tenga conto inoltre di come la classe `DocumentItem` viene usata polimorficamente nei test forniti.

Classe `Paragraph`:

- Realizza il concetto di paragrafo di testo (che può essere parte di un documento); il suo contenuto è rappresentato attraverso l'attributo `text` che si imposta con il costruttore.

- Un paragrafo può essere in stato di *locking*, nel qual caso esso non potrà essere modificato. L'attributo `lock` e i relativi metodi di accesso – `isLocked()` – e modifica – `setLock(boolean)` – permettono di gestire il meccanismo di lock/unlock.
- Il costruttore inizializza il testo del paragrafo in base alla stringa passata come parametro. Il costruttore gestisce gli errori sull'inizializzazione del testo sollevando l'eccezione `NonValidDocumentException` (la cui classe viene fornita e non va modificata) se il parametro passato ha il valore `null`. Inoltre il costruttore imposta lo stato di *locking* del paragrafo a `false`.
- Il metodo `update(DocumentItem)` modifica il testo del paragrafo sostituendolo con il testo del documento passato come parametro. La modifica può aver luogo solo se il paragrafo non è in stato di *locking*, altrimenti il metodo solleva l'eccezione `ModifyingLockedDocumentException` (la cui classe viene fornita e non va modificata).
- Il metodo `getText()` restituisce il contenuto del paragrafo come stringa di testo.
- Il metodo `charactersCount()` restituisce il numero di caratteri contenuti paragrafo.

Classe `Document`:

- Realizza il concetto di documento formato da *sezioni*: ogni sezione può essere un paragrafo o un altro documento. L'associazione viene gestita usando una collezione di tipo `ArrayList`.
- Un documento può essere in stato di *locking*, nel qual caso esso non potrà subire modifiche ad alcuna sua sezione. L'attributo `lock` e i relativi metodi di accesso – `isLocked()` – e modifica – `setLock(boolean)` – permettono di gestire il meccanismo di lock/unlock.
- Il costruttore imposta la prima sezione del documento: la prima sezione corrisponderà al parametro in ingresso. Il costruttore gestisce gli errori sull'inizializzazione del documento, sollevando l'eccezione `NonValidDocumentException` se il parametro passato ha il valore `null`. Inoltre il costruttore imposta lo stato di *locking* del documento a `false`.
- Il metodo `update(DocumentItem)` aggiunge una ulteriore sezione al documento: la nuova sezione corrisponderà al parametro in ingresso. La modifica può aver luogo solo se il documento non è in stato di *locking*, altrimenti il metodo solleva l'eccezione `ModifyingLockedDocumentException`.
- Il metodo `update(DocumentItem, int cursor)` modifica la sezione in posizione `cursor` della collezione (il conteggio delle posizioni comincia da 0) invocando un opportuno `update` della sezione indicata. Invece, se il valore di `cursor` non corrisponde numericamente a nessuna sezione del documento, la modifica va fatta aggiungendo una ulteriore sezione al documento: la nuova sezione corrisponderà al parametro in ingresso. Il metodo solleva l'eccezione `ModifyingLockedDocumentException` in tutti i casi in cui la modifica richiesta è inibita dallo stato di locking del documento o della sezione indicata.
- Il metodo `getText()` restituisce il contenuto del documento come stringa di testo. Il contenuto si ottiene concatenando il testo di tutte le sezioni del documento.
- Il metodo `charactersCount()` restituisce il numero di caratteri contenuti nel documento.