Informazioni sul documento

Nome documento	Relazione Prima Parte del progetto PuzzleSolver
Versione documento	v.1.0.0
Data redazione	2014-12-04
Redattore	Tesser Paolo

Sommario

Lo scopo del documento è quello di fornire una presentazione del prima parte del progetto PuzzleSolver da realizzare, descrivendo e motivando le scelte attuate in questa fase.

Indice

1 Organizzazione strutturale delle classi

La struttura dell'applicativo è composta da diversi package, con all'esterno di essi la classe **PuzzleSolver** responsabile dell'esecuzione del programma e dell'istanziazione delle classi in oggetti per risolvere il puzzle.
Anche se tutte le classi all'interno di questi package lavorano bene assieme e tra loro esistono alcune dipendenze, si è deciso di tenere un minimo si separazione logica per le diverse componenti.
Nei sotto capitoli seguenti verranno illustarti i package utilizzati e le classi presenti in essi.

1.1 Puzzle

TO DO

1.2 Solver

TO DO

1.3 FileInputOutput

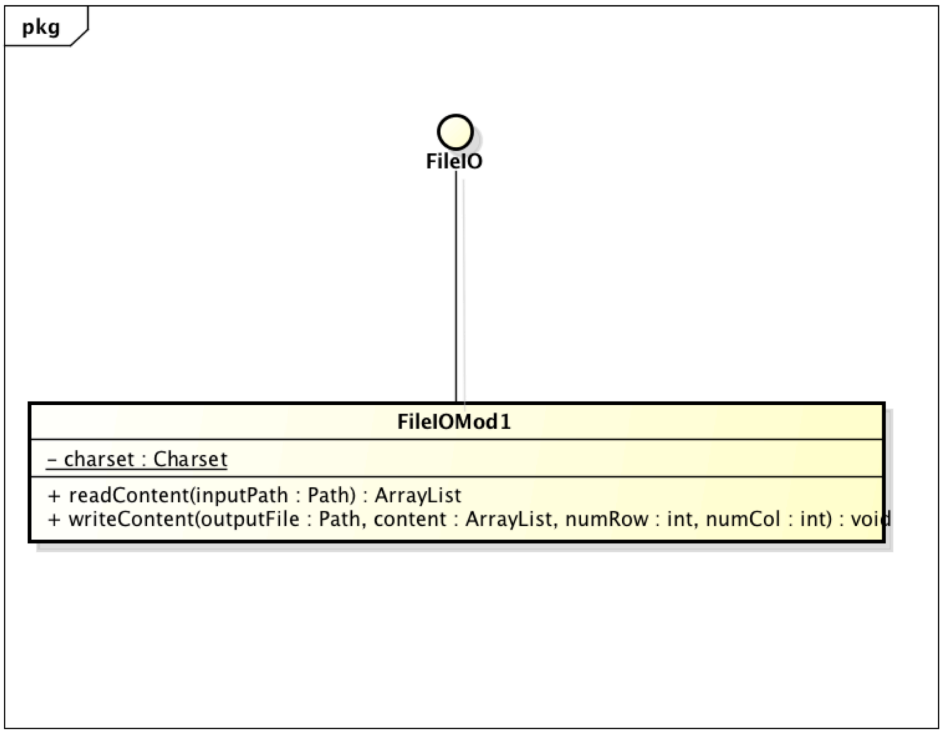

Figura 1: Package FileInputOutput

TO DO

2 Principi di Programmazione ad Oggetti

In questa capitolo vengono descritte le scelte effettuate per implementare i principi della programmazione ad oggeti, in particolare quelli di incapsulamento e di information hiding.

2.1 Principio di modularità

E' stato deciso di dividere la struttura dell'applicativo in diversi package.
Questo permette una migliore manutenzione del codice in quanto in ogni package sono contenute delle classi logicamente correlate e niente altro che possa essere di interferenza con il loro funzionamento.
Si è deciso di rendere inoltre il codice il più possibile estendibile e ciò è stato realizzato tramite l'utilizzo di interfacce e di classi astratte.
Maggiori dettagli su come è stato realizzato il tutto sono illustrati nel capitolo precedente, dedicato apposta alle scelte architetturali.

2.2 Principio di incapsulamento

Per avere un buon incapsulamento si è cercato di rendere le classi il più isolate possibili, permettendone l'accesso ai suoi membri solo attraverso degli opportuni metodi getter/setter e rendendo tali membri privati.
Questi marcatori di accesso inoltre sono quasi tutti firmati **final** per non consentire delle future ridefinizioni da sottoclassi che ne possano compromettere il naturale comportamento.
TO DO

2.3 Principio di information hiding

TO DO

2.4 Principio di sostituizione

Per garantire che l'utente possa scrivere codice basato sulla specifica TO DO

3 Algoritmo di risoluzione

L'algoritmo scelto per risolvere il puzzle è di tipo sequenziale, come richiesto dalla specifica di progetto.
Per arrivare alla soluzione vengono utilizzati due strutture dati come membri della classe PuzzleCharacter.
La prima struttura è la collezione HashMap, nella quale vengono salvati in ordine casuale i tasselli ricevuti in input dal file di testo. Come chiave inserisco l'id del tassello mentre come valore uso l'intero pezzo (Tile).
La seconda struttura dati è un array bidimensionale di oggetti Tile. In essa andranno memorizzati i vari pezzi in ordine corretto, questo contenitore infatti rappresenterà il puzzle nell'ordine corretto alla fine dell'algoritmo.
Di seguito vengono esposte le seguenze che vengo eseguite, correlate da dei grafici che mostrano come esso agisca sulle strutture dati utilizzate.

1. **Ricerco il primo elemento del puzzle (quello in alto a sinistra).**
   Per fare ciò scorro una sola volta la tavola hash per cercare il tassello che ha id nord e id ovest uguale alla stringa VUOTO.
   Una volta trovato salvo il pezzo nella prima posizione dell'array bidimensionale.

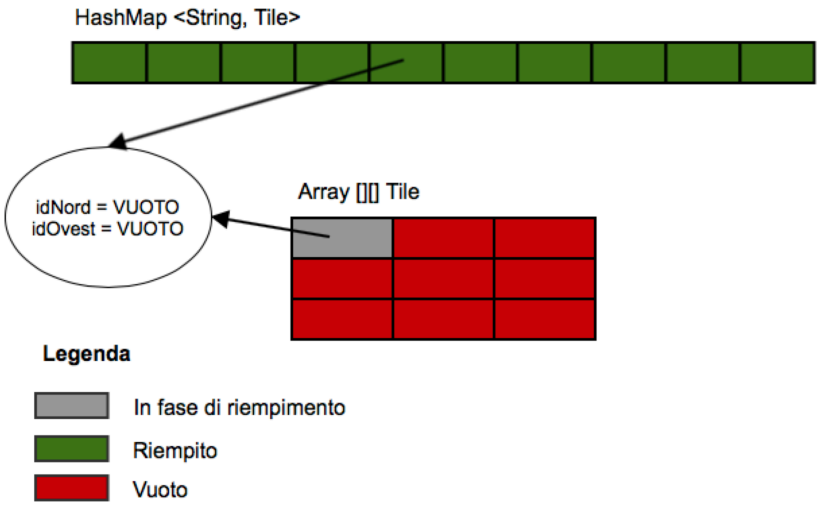

Figura 2: Algoritmo Step 1 - Ricerca del primo elemento

2. **Ordino la colonna più a sinistra (quella con i tasselli aventi id ovest uguale alla stringa VUOTO).**
   Prendo l'elemento trovato al passo successivo, mi estraggo l'id sud di esso e tramite i metodi della tavola hash mi ricavo il tassello reale a cui corrisponde.

Una volta estratto lo inserisco nella posizione corretta.
Continuo così da quello appena trovato fino a quando non trovo tutti quelli sottostanti.

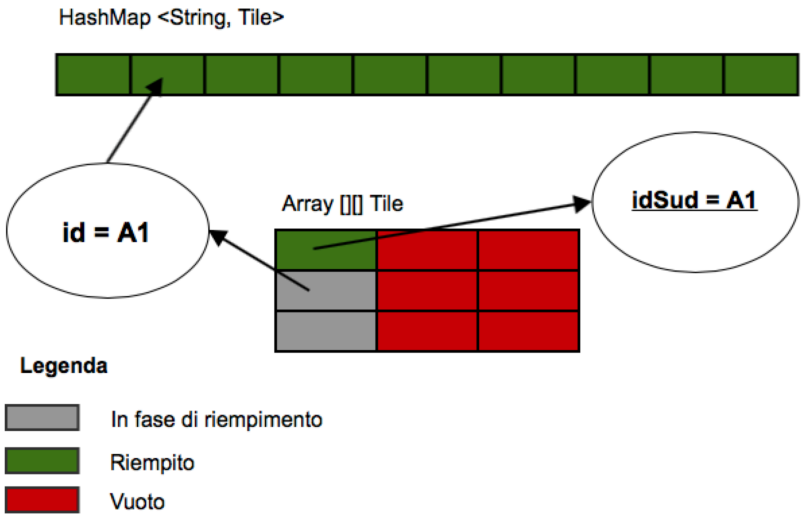

Figura 3: Algoritmo Step 2 - Ricerca della colonna più a sinistra

3. **Ordino tutte le righe.**
   Dopo aver ricavato tutta la prima colonna, eseguo, secondo lo stesso principio, anche la risoluzione per le righe, procedendo però sta volta con la ricerca del pezzo successivo a destra tramite l'id est.

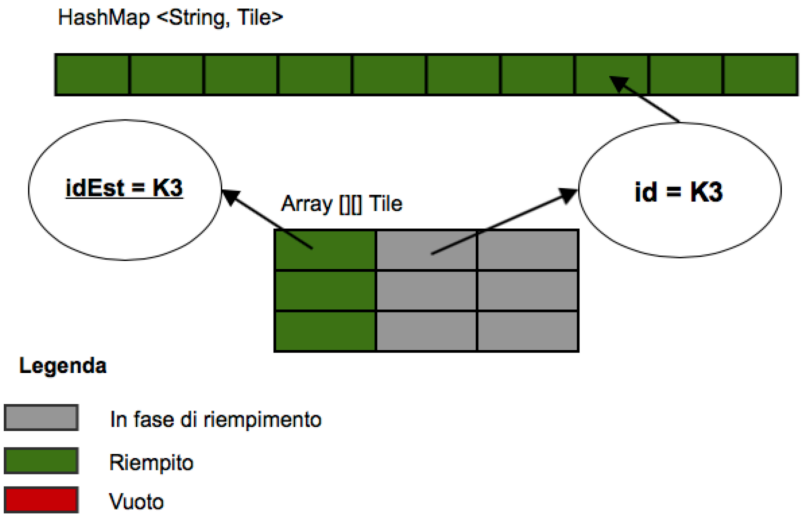

Figura 4: Algoritmo Step 3 - Ricerca delle righe

4 Test di correttezza

TO DO