Progetto Programmazione Concorrente e Distribuita
Prima Parte

Suierica Bogdan Ionut 1008089

December 22, 2014

1 Introduzione

1.1 Scopo del documento

Lo scopo del documento è quello di presentare le principali scelte architetturali del progetto.

2 Descrizione Progetto

Il progetto realizzato permette di ordinare un puzzle di caratteri. Il file in input contiene il testo rappresentato dal puzzle disordinato. Il file in output contiene il testo rappresentato dal puzzle disordinato organizzato in una riga, il puzzle ordinato in forma tabellare, e la dimensione della tabella.

3 Componenti e Classi

3.1 Introduzione

Tutti i campi dati delle classi utilizzate nel progetto sono privati e dotati di metodi pubblici per permettere l'accesso. Questo offre mantenibilità, flessibilità ed estensibilità del codice.
Nel progetto ho cercato di dividere tutti i vari compiti in classi.

3.2 Package puzzle

Per questa prima parte del progetto, puzzle rappresenta il package globale del progetto. Si potevano utilizzare due package differenti, uno per le oper-

azioni di input-output e uno per le funzionalità di ordinamento del puzzle, ma ho preferito tenere tutto insieme, in quanto tutte le classi erano relative allo stesso argomento e cioè a quello di ordinare un puzzle di stringhe da un file in input.

3.3 Interface SolverAlgorithm

Interfaccia che espone la funzionalità di ordinare un Puzzle. Il suo unico metodo **solve** viene realizzato nella classe **SolvedPuzzle**.

3.4 Classe IOReader

Questa classe rappresenta la lettura da file del puzzle disordinato in forma tabellare. Quando avviene la lettura , calcola la dimensione della tabella attraverso dei contatori che contano i Tile con idNord="VUOTO" e idOvest="VUOTO".

3.5 Classe IOWriter

Questa classe rappresenta la scrittura su file e come detto in precedenza, scrive il testo rappresentato dal puzzle disordinato, il puzzle ordinato in forma tabellare e la dimensione della tabella.

3.6 Classe Tile

Questa classe rappresenta il singolo pezzo del puzzle. Ogni pezzo ha un id univoco e gli id dei pezzi che li stanno vicino nelle quattro direzioni.

3.7 Classe Puzzle

Questa classe rappresenta l'insieme di pezzi del puzzle. E' caratterizzata da un ArrayList di **Tile** , due interi che indicano la dimensione della puzzle, una stringa che indica il path a cui si trova il file in input e un riferimento all'oggeto **IOReader** .
Inoltre alla costruzione dell'oggetto **Puzzle** avviene la lettura da file e l'ArrayList di **Tile** viene popolato.

3.8 Classe SolvedPuzzle

La classe **SolvedPuzzle** implementa l'interfaccia **SolverAlgorithm**. Ha il compito di salvarsi il puzzle disordinato e restituirlo ordinato attraverso il metodo realizzato solve, fornito dal interfaccia **SolverAlgorithm**.

4 Logica Algoritmo di ricostruzione del puzzle

Per l'algoritmo di ricostruzione uso l'interfaccia **SolverAlgorithm** in cui viene reso disponibile il metodo **solve** che accetta un **Puzzle** come parametro. Il metodo **solve** implementato nella classe **SolvedPuzzle** chiama altri due metodi : **SolveFirstCol** e **SolveRemainingTile**. Il primo metodo serve a ordinare solamente la prima colonna del puzzle. Il secondo serve a ordinare il resto del puzzle partendo dalla prima colonna. Ho scelto di fare cosi pensando alla seconda parte del progetto che tratta la concorrenza.