Programmazione Concorrente e Distribuita 2014-2015. Seconda parte.

Andrea Baldan

August 26, 2015

Contents

1 Modifiche rispetto alla prima parte

Il design pattern scelto ha permesso di mantenere inalterata l'organizzazione delle classi, permane quindi l'albero della directory formato da **models**, **views** e **controllers** e il package **puzzlesolver** con l'unica differenza che la classe SortAlg ora è stata estesa da 2 sottoclassi, SortAlgFromTop e SortAlgFromBottom, questo per permettere la parallelizzazione dell'algoritmo di ordinamento ora non più sequenziale.
Le rimanente struttura del programma è rimasta pressochè invariata.

2 Algoritmo di ordinamento parallelizzato

Per consentire un approccio concorrente all'algoritmo ideato nella prima parte del progetto, la classe base astratta SortAlg.java questa volta è stata estesa in due sottoclassi che implementano l'interfaccia **Runnable**, in modo da potere essere incapsulate in un oggetto di tipo thread ed aver così la possibilità di lanciare il metodo sort in maniera concorrente.
SortAlgFromTop.java, la prima delle due sottoclassi si occupa di ordinare la prima metà del puzzle utilizzando essenzialmente lo stesso sistema impiegato nella prima parte del progetto, ovvero localizzazione del primo pezzo (nord e ovest "VUOTO"), ordinamento della riga mediante metodo nextInRow, localizzazione del pezzo a sud del primo precedentemente localizzato mediante metodo nextInCol e cosi via fino al raggiungimento della metà.
SortAlgFromBottom.java si occupa in maniera analoga di ordinare la metà inferiore del puzzle, ma questa volta l'algoritmo, per quanto si tratti delle stesse operazioni di SortAlgSeq, è stato implementato per ricostruire il puzzle al contrario. Il primo pezzo è quindi in questo caso l'ultimo del puzzle (sud ed est "VUOTO") e si procede in senso contrario a SortAlgSeq, fino al raggiungimento della metà prefissata.