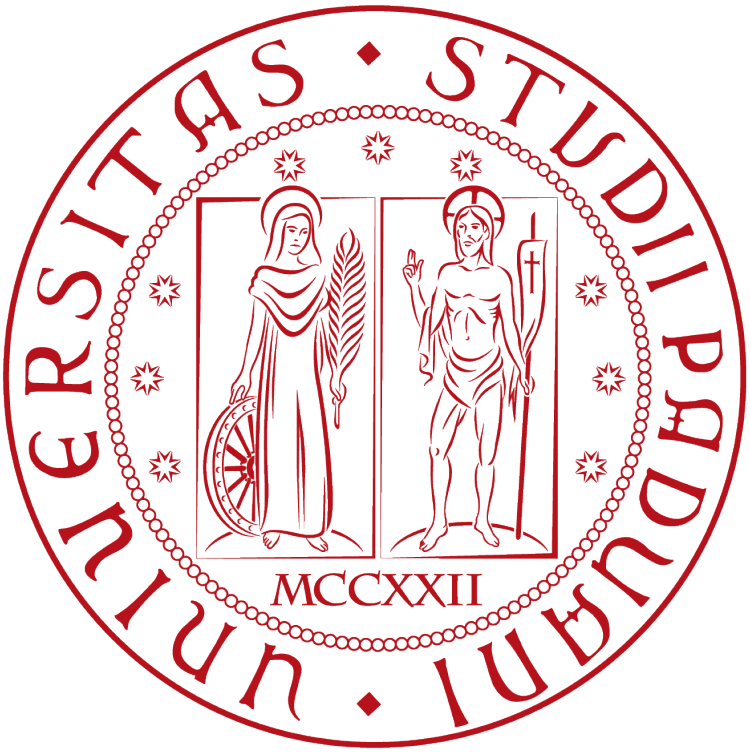

Informazioni sul documento

Nome documento	Relazione Terza Parte del progetto PuzzleSolver
Versione documento	v.3.0.0
Data redazione	2015-01-22
Redattore	Tesser Paolo

Sommario

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

Indice

1 Note introduttive **3**

2 Cambiamenti e Aggiunte **3**
2.1 Aggiunte 3
2.1.1 Package objrem 3

3 Logica di comunicazione client-server **5**

4 Robustezza **6**

5 Note sulla compilazione **7**
5.1 Versione JVM 7
5.2 Compilazione 7

1 Note introduttive

In questa terza parte, anche se la specifica non prevedeva come requisito la possibilità di avere più client che potessero eseguire la risoluzione di un puzzle sul server in maniera concorrente, si è deciso di implementare lo stesso tale funzionalità.
Il modo nel quale verrà effettuato ciò sarà illustrato nelle sezioni 2 e 3.

2 Cambiamenti e Aggiunte

In questa sezione verranno descritti i cambiamenti e le aggiunte apportate alla precedente versione per permettere al programma di essere distribuito tra un server e più client (come evidenziato nella sezione 1).
Non sono state effettuate particolare revisioni al codice sviluppato per soddisfare i requisiti della seconda parte.
É stata cancellata solo la classe **PuzzleSolver**, responsabile dell'esecuzione del programma, non più necessaria, a favore di due nuove classi: **PuzzleSolverServer** e **PuzzleSolverClient**, responsabili dell'esecuzione del programma sul server e di quello sul client. Nella sezione 3 verrà illustrato quale compito svolgono attraverso il loro main queste classi.

2.1 Aggiunte

2.1.1 Package objrem

Questo package contiene le classi TO DO.
TO DO

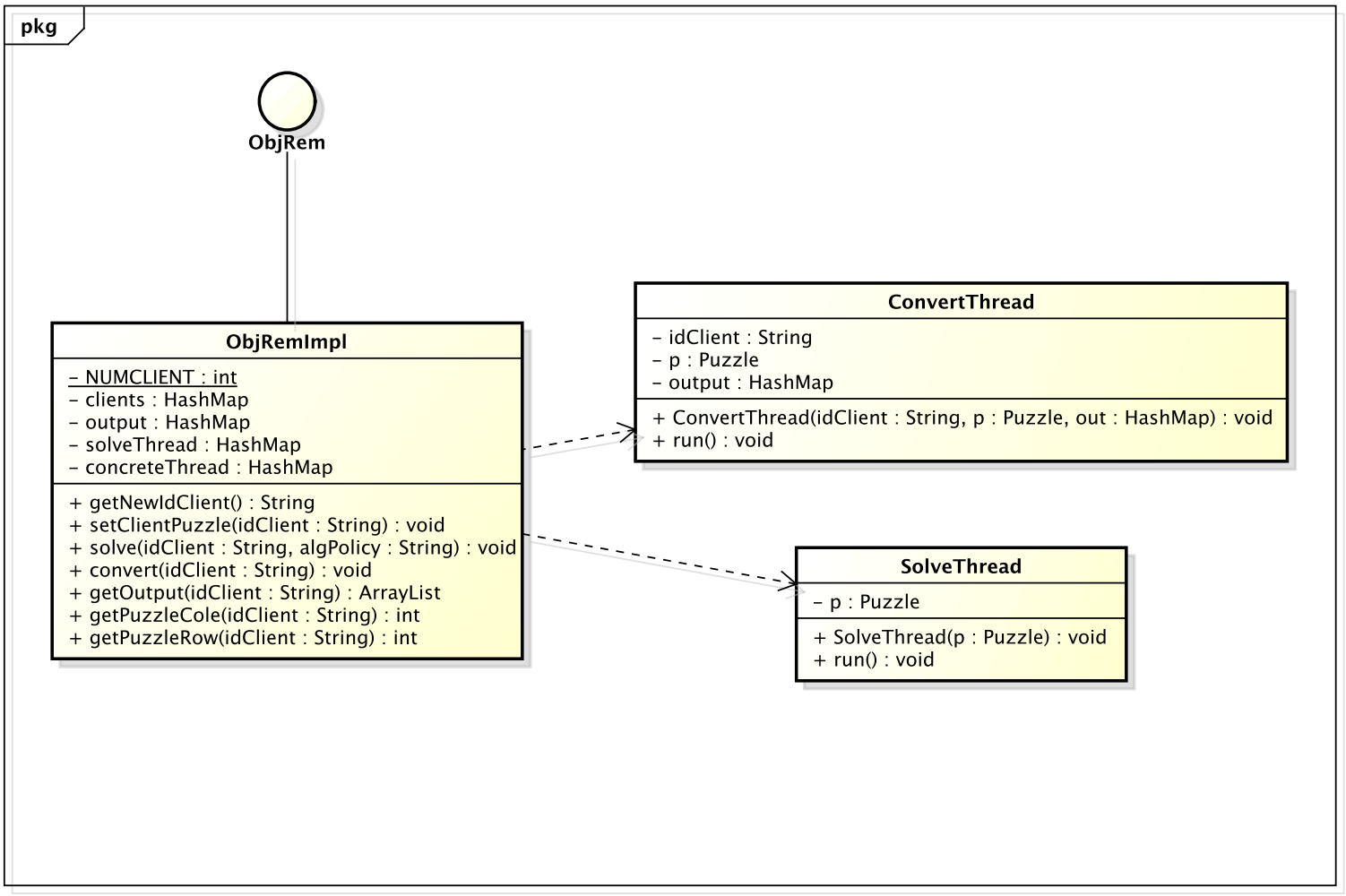

Figura 1: Package objrem

3 Logica di comunicazione client-server

Per la comunicazione tra i diversi client e il server il programma adotta la libreria RMI che offre un livello di astrazione più elevato rispetto al meccanismo di comunicazione tramite socket.
Il suo scopo è quello di rendere trasparenti al programmatore quasi tutti i dettagli della comunicazione su rete.
Per implementare il meccanismo offerto da RMI, ci serviamo del package **objrem** citato nella sezione precedente.
TO DO

4 Robustezza

TO DO

5 Note sulla compilazione

5.1 Versione JVM

La versione di Java presente nella macchina utilizzata per l'implementazione del codice è quella 1.8.0_20.
Non sono stati usati però costrutti particolari di questa versione e per l'utilizzo di API si è sempre fatto riferimento alla documentazione ufficiale della 1.7.
Per essere sicuri di rispettare la specifica che richiedeva al massimo la versione 1.7, si è testata la compilazione e l'esecuzione dell'applicativo sulle macchine di laboratorio dopo avere eseguito da terminale le istruzioni presenti al seguente link: http://www.studenti.math.unipd.it .

5.2 Compilazione

Dalla root principale consegnata è possibile avviare il comando per la compilazione sia dei file necessari al server sia quelli necessari al client attraverso l'istruzione **make**.
Se si vuole lanciare il programma, testandolo con degli input definiti dal fornitore per provare l'applicativo, è possibile eseguire sempre da root i seguenti comandi:

1. **make loadserver**: il quale come prima cosa avvierà il **registro rmi** e poi lancerà il programma lato server, prendendo come parametro una stringa di testo che rappresenta il nome del server;

2. **make loadclient**: il quale lancerà il programma lato client prendendo tre file fissi, quali il file di input, quello di output e una stringa di testo che rappresenta il nome del server.n

Se si vuole invece lanciare il programma con dei file di test personalizzati e un server qualunque è possibile eseguire i seguenti script bash che lanceranno l'applicativo (la compilazione andrà effettuata precedentemente come descritto nella prima parte di questa sezione):

1. **bash puzzlesolverserver.sh "nome del server"**: questo script riceve come input il nome del server. Lanciandolo verrà eseguito il comando per avviare il registro rmi e in seguito quello per avviare il main del server;

2. **bash puzzlesolverclient.sh input output "nome del server"**: questo script riceve come input tre parametri, quali file di input, file di output e il nome del server. Lanciandolo verrà eseguito il comando per avviare il main del client.

Attenzione: lo script bash: puzzlesolverclient.sh, che lancia l'esecuzione del comando *java* sul main del client principale va ad eseguire il comando in un livello inferiore rispetto ad esso. Bisogna fare quindi attenzione al percorso del file in input che potrebbe generare un'eccezione qualora non fosse corretto.