ΠΑΡΑΛΛΗΛΟΣ ΚΑΙ ΚΑΤΑΝΕΜΗΜΕΝΟΣ ΥΠΟΛΟΓΙΣΜΟΣ

ΚΑΘΗΓΗΤΗΣ: ΜΑΡΓΑΡΙΤΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Αρχοντής-Εμμανουήλ Κωστής | ics21044

ΕΡΓΑΣΙΑ 11
RMI vs SOCKETS

ΜΕΤΡΗΣΕΙΣ:

ΥΛΟΠΟΙΗΣΗ ΜΕ ΧΡΗΣΗ SOCKETS	
FILENAME	**NCLOC (Lines of Code)**
ClientProtocolCalculator.java	42
EchoClientTCP.java	28
EchoServerTCP.java	32
ServerProtocolCalculator.java	35
TOTAL	**137**

ΥΛΟΠΟΙΗΣΗ ΜΕ ΧΡΗΣΗ RMI	
FILENAME	**NCLOC (Lines of Code)**
CLIENT SIDE	
Calculator.java	4
CalculatorClient.java	29
SERVER SIDE	
Calculator.java	4
CalculatorImpl.java	17
CalculatorServer.java	16
CalculatorService.java	46
TOTAL	116

Από τις μετρήσεις παρατηρούμε πως η υλοποίηση με χρήση του μηχανισμού RMI έχει ελαφρώς χαμηλότερο αριθμό Γραμμών Κώδικα, σε σχέση με την υλοποίηση με Sockets. Αυτό αντικατοπτρίζει την αφαίρεση και τον υψηλού επιπέδου χειρισμό που μας παρέχει αυτή η τεχνική, κάτι που μας “γλιτώνει” αρκετή δουλειά και κάνει τον κώδικα πιο συντηρήσιμο. Πρακτικά η τεχνική RMI αφαιρεί μεγάλο μέρος της επικοινωνίας που γίνεται μέσω δικτύου, δίνοντας περισσότερο βάση στον ορισμό των απομακρυσμένων κλήσεων και στο business logic. Τα σφάλματα και τα προβλήματα που μπορεί να προκύψουν σχετικά με το δίκτυο τα χειρίζεται ο μηχανισμός του Remote Method Invocation, κάτι που απλοποιεί τον χειρισμό σφαλμάτων και edge cases τόσο στον κώδικα του πελάτη όσο και του διακομιστή. Τέλος

στην υλοποίηση μας παρέχεται ένα επιπλέον επίπεδο αφαίρεσης μέσω της κλάσης CalculatorService η οποία είναι η κλάση που είναι πραγματικά υπεύθυνη για τον χειρισμό του υπολογισμού ενώ η κλάση CalculatorImpl ασχολείται με τα θέματα που αφορούν το RMI. Ο διαχωρισμός αυτός μας βοηθάει να έχουμε ακόμη πιο καθαρό κώδικα που κάθε κλάση έχει την δική της αρμοδιότητα (κάτι που ακολουθεί την SRP).
Στην υλοποίηση χωρίς RMI πέρα από την αύξηση στις γραμμές κώδικα, παρατηρούμε και ότι η πολυπλοκότητα της διαχείρισης σφαλμάτων είναι μεγαλύτερη. Πρακτικά καθώς ο χειρισμός της επικοινωνίας μέσω δικτύου απαιτεί “χειρωνακτική” εργασία (δηλ. πρέπει να γράψουμε ένα σαφώς ορισμένο πρωτόκολλο) όπου χειριζόμαστε τις διαφορετικές συνδέσεις, μηνύματα και σφάλματα. Τα σφάλματα απαιτούν έναν πιο ρητό χειρισμό όσον αφορά τις εξαιρέσεις δικτύου και την “τήρηση” των πρωτοκόλλων.

Συμπερασματικά ο μηχανισμός RMI προσφέρει μια υψηλού επιπέδου προσέγγιση, που μειώνει την πολυπλοκότητα του κώδικα, το μέγεθος του και τον χειρισμό σφαλμάτων. Από την άλλη, η χρήση Sockets παρέχει μεν περισσότερο έλεγχο (καθώς μπορούμε π.χ. να ορίσουμε δικά μας custom πρωτόκολλα), αλλά απαιτεί περισσότερο κώδικα και πιο ρητή και λεπτομερή διαχείριση σφαλμάτων.

Η διαφορά μεταξύ των Γραμμών Κώδικα κάθε υλοποίησης φαίνεται στα παρακάτω γραφήματα: