ESERCITAZIONE 6
Java RMI

- Bernardi Daniel
- Chichifoi Karina
- Gjura Endri
- Ivan Andrei Daniel
- Pizzini Cavagna Hiari

Introduzione

Si è deciso, per quanto riguarda il servizio di conteggio delle righe con più parole di una certa soglia inserita dall'utente, di chiedere sempre all'utente quali delimitatori usare per le parole e questi vengono passati come argomento in una stringa.

Per quanto riguarda il secondo servizio offerto dal Server, di eliminazione di una riga da un file di testo, si crea una copia del file e, se la riga esiste, la si elimina. Infine si elimina il file originale e si rinomina la copia con il nome originale del file.

Codice Client (conteggio righe)

```
while ((service = stdIn.readLine()) != null) {
      if (service.equals("C")) {
            String fileName, delim;
            int numParole;

            System.out.print("Inserire il nome del file: ");
            fileName = stdIn.readLine();

            System.out.print("Inserire il numero di parole per riga da ricercare: ");
            numParole = Integer.parseInt(stdIn.readLine());

            System.out.print("Inserire i separatori: ");
            delim = stdIn.readLine();

            System.out.println("Conto le righe del file " + fileName + " con oltre " + numParole + " parole");

            try {
                  System.out.println("Sono presenti " + serverRMI.conta_righe(fileName, numParole, delim) + " righe valide");
            } catch (RemoteException re) {
                  System.err.println("Possibile errore nel file; " + re.getMessage());
                  re.printStackTrace();
      }
```

Si passa al Client la lettera C o la lettera E a seconda che si voglia il servizio di conteggio righe o eliminazione riga e da qui si divide il codice.

Codice Client (eliminazione riga)

```
    } else if (service.equals("E")) {
        String fileName;
        int numRiga, ris;

        System.out.print("Inserire il nome di un file: ");
        fileName = stdIn.readLine();

        System.out.print("Inserire la riga da eliminare: ");
        numRiga = Integer.parseInt(stdIn.readLine());

        try {
            System.out.println("La linea è stata eliminata e il file ha " + serverRMI.elimina_riga(fileName, numRiga) + " righe");
        } catch (RemoteException re) {
            System.err.println("Possibile errore nel file; " + re.getMessage());
            re.printStackTrace();
        }
    } else {
        System.out.println("Servizio non esistente!");
        System.out.println("Servizi disponibili: C -> Conta parole; E -> Elimina riga, CTRL+D per uscire: ");
    }
    System.out.print("Servizio (C = Conta parole - E = Elimina riga), CTRL+D per uscire: ");
}
```

Algoritmo Server (eliminazione riga)

```
public int elimina_riga(String fileName, int rowNum) throws RemoteException {
      File file = new File(fileName);
      try {
            BufferedReader br = new BufferedReader(new FileReader(file));
            String line;
            int numRowNew = 0, index = 0;
            boolean checkRow = false;
            File fileR = new File("alias");
            PrintWriter pw = new PrintWriter(fileR);
            while ((line = br.readLine()) != null) {
                  if (index != rowNum) {
                        pw.println(line);
                        numRowNew++;
                  } else {
                        checkRow = true;
                  }
                  index++;
            }
            pw.close();
            br.close();
            file.delete();
            fileR.renameTo(file);

            return numRowNew;
      }
}
```

Si crea una copia del file su cui lavorare e alla fine si elimina l'originale e rinomina la copia.

Conclusione

Per concludere il progetto si era pensato di fare una prova comparativa di performance dell'algoritmo di lettura da file nel primo servizio offerto dal Server (conteggio righe).

```
public int conta_righe(String fileName, int threshold, String delim) throws RemoteException, FileNotFoundException;
public int conta_righe_char(String fileName, int threshold, String delim) throws RemoteException, FileNotFoundException;
```

Le due varianti dell'algoritmo, tramite StringTokenizer e carattere per carattere, sono state sviluppate ed entrambe funzionanti ma a causa di alcuni problemi nella misurazione dei tempi non sono ritenibili affidabili.

Ad esempio, su file di grandi dimensioni, aumentando il numero minimo di parole per riga, si otteneva un (circa) dimezzamento dei tempi, cosa alquanto improbabile. Per ovviare a questo problema si è deciso di chiudere e aprire Client e Server a ogni prova, ma questo forniva risultati troppo sparsi e poco attendibili.