12 Ottobre
2021

Esercitazione 2

Socket in JAVA
(connection oriented)

Bernardini Claudio
Corsetti Luca
Giardina Gianluca
Straccali Leonardo

Obiettivi esercitazione

Hard skills

1. Stabilire e mantenere una connessione tramite Socket
2. Gestione sequenziale e parallela lato servitore

Soft skills

1. Collaborazione per raggiungere un fine comune
2. Organizzazione del proprio lavoro in team

Introduzione

Client

1. Stabilisce una connessione con il servitore
2. Invia le informazioni di ogni file di un direttorio
3. Attende il comando di attivazione da parte del server

Servervitore Sequenziale

1. Attende richiesta di una connessione
2. Ricevute le informazioni dal client invia il comando di attivazione

Servervitore Concorrente

1. Crea un thread per ogni richiesta di commissione
2. Ogni thread ricevute le informazioni dal client invia il comando di attivazione
3.

Implementazione Client

```
File[] filesDirectory = directory.listFiles();

if ((numFileDir = filesDirectory.length) == 0) {
    System.out.println("Usage: Param directoryPath is empty");
    System.exit( status: 5);
}

System.out.println("Directory " + args[2] + ": file da trasferire " + numFileDir);

for (count = 0; count < numFileDir; count++){
    stateCheck = filesDirectory[count].isFile()
                 && filesDirectory[count].length() >= limitDimFile;
    if(stateCheck){
        if(nFDEff != count) filesDirectory[nFDEff] = filesDirectory[count];
        nFDEff++;
    }
}
```

A

```
for (count = 0; count < nFDEff; count++) {

    dimFile = filesDirectory[count].length();
    nomeFile = filesDirectory[count].getName();

    try {
        inFile = new FileInputStream(filesDirectory[count]);
    } catch (FileNotFoundException e) {...}

    try {
        outSock.writeUTF(nomeFile);
        System.out.println("Inviato il nome del file " + nomeFile);
    } catch (Exception e) {...}

    try {
        esito = inSock.readUTF();
        System.out.println("Esito trasmissione nome file: " + esito);
        if (esito.equalsIgnoreCase( anotherString: "salta")) {
            System.out.println("File " + nomeFile + " già presente sul server!");
        } else if (esito.equalsIgnoreCase( anotherString: "attiva")) {
            try {
                outSock.writeLong(dimFile);
                FileUtility.trasferisci_a_byte_file_binario(new DataInputStream(inFile),
                                                            outSock,
                                                            dimFile);
                inFile.close();
                System.out.println("Trasmissione di " + nomeFile + " terminata ");
            } catch (SocketTimeoutException ste) {...} catch (Exception e) {...}
        }
    } catch (SocketTimeoutException ste) {...}
    catch (Exception e) {...}
}
```

B

```
public static void trasferisci_a_byte_file_binario(DataInputStream src,
                                                    DataOutputStream dest,
                                                    long dimFile) throws IOException {
    int buffer;
    try {
        for (buffer = 0; buffer < dimFile; buffer++) {
            dest.write(src.read());
        }
        dest.flush();
    }
    catch (EOFException e) {...}
}
```

C

Implementazione

Server Sequenziale

A

```
try {
    nomeDir = inSock.readUTF();
    File dir = new File(nomeDir);
    esito = dir.exists() ? "saltaDir" : "attivaDir";

    if (esito.equals("attivaDir")) {
        dir.mkdirs();
        System.out.println("Ricevuta la cartella " + nomeDir);
    }
} catch (SocketTimeoutException ste) {...}
catch (IOException e) {...}

try {
    outSock.writeUTF(esito);
} catch (IOException e) {...}
```

B

```
try {
    nomeFile = nomeDir + "/" + inSock.readUTF();

    if (nomeFile == null) {...}
}
catch (SocketTimeoutException ste) {...}
catch (IOException e) {...}

curFile = new File(nomeFile);
esito = curFile.exists() ? "salta" : "attiva";

try {
    outSock.writeUTF(esito);
} catch (IOException e) {...}

if (esito.equals("attiva")) {
    outFile = new FileOutputStream(nomeFile);

    try {
        try {
            dimFile = inSock.readLong();
            System.out.println("dimensione di " + nomeFile + ": " + dimFile);
        } catch (SocketTimeoutException ste) {...}
        catch (IOException e) {...}

        System.out.println("Ricevo il file " + nomeFile + ":");
        FileUtility.trasferisci_a_byte_file_binario(inSock, new DataOutputStream(outFile), dimFile);
        System.out.println("Ricezione del file " + nomeFile + " terminata");
        outFile.close();
    } catch (SocketTimeoutException ste) {...}
    catch (Exception e) {...}
}
```

Implementazione

Server Concorrente

A

```
try {
    nomeDir = inSock.readUTF();
    File dir = new File(nomeDir);
    synchronized (dir.getCanonicalPath().intern()) {
        esito = dir.exists() ? "saltaDir" : "attivaDir";

        if (esito.equals("attivaDir")) dir.mkdirs();
    }
} catch (SocketTimeoutException ste) {...}
catch (IOException e) {...}

try {
    outSock.writeUTF(esito);
}
catch (IOException e) {...}
```

Synchronized statements

B

```
try {
    nomeFile = nomeDir+"/"+inSock.readUTF();
    if (nomeFile == null) {...}
} catch (SocketTimeoutException ste) {...}
catch (IOException e) {...}

curFile = new File(nomeFile);
synchronized (curFile.getCanonicalPath().intern()) {
    esito = curFile.exists() ? "salta" : "attiva";
    if (esito.equals("attiva")) outFile = new FileOutputStream(nomeFile);
}

try {
    outSock.writeUTF(esito);
} catch (IOException e) {...}

if (esito.equals("attiva")) {
    try {
        try {
            dimFile = inSock.readLong();
            System.out.println("dimensione di " + nomeFile + ": " + dimFile);
        } catch (SocketTimeoutException ste) {...}
        catch (IOException e) {...}

        System.out.println("Ricevo il file " + nomeFile + ":");
        FileUtility.trasferisci_a_byte_file_binario(inSock, new DataOutputStream(outFile), dimFile);
        System.out.println("Ricezione del file " + nomeFile + " terminata");
        outFile.close();
    }
    catch (SocketTimeoutException ste) {...}
    catch (Exception e) {...}
}
```

Benchmarks

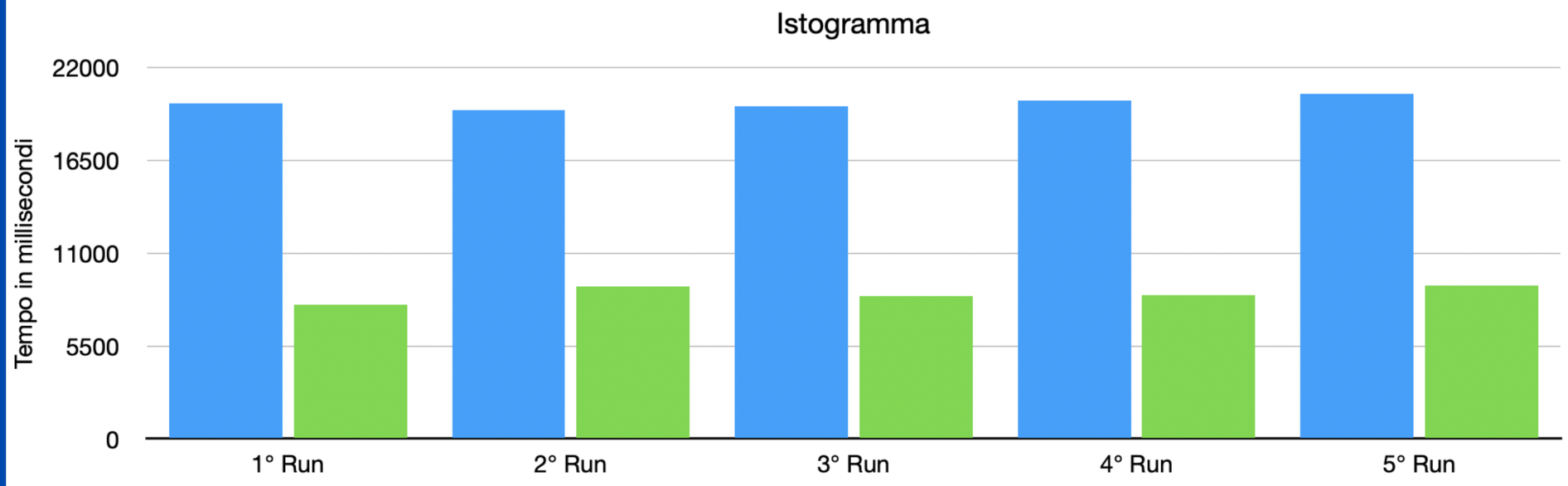
Riferimenti benchmark:

cpu: i7 7700k - ram 32gb ddr4 3200mhz

6 file da:

- 35kb
- 70kb
- 139kb
- 278kb
- 417kb
- 833kb

	1° Run	2° Run	3° Run	4° Run	5° Run
Server Sequenziale	19871	19494	19700	20057	20411
Server Concorrente	7969	9032	8475	8500	9078



Gestione progetto

Coding

Team