Università di Pisa

Assignments di Laboratorio di Reti
A.A. 2020/21

Assignment 01 - PiGreco

Scrivere un programma che attiva un thread T che effettua il calcolo approssimato di π.

Il programma principale riceve in input da linea di comando un parametro che indica il grado di accuratezza (accuracy) per il calcolo di π ed il tempo massimo di attesa dopo cui il programma principale interrompe il thread T.

Il thread T effettua un ciclo infinito per il calcolo di π usando la *serie di Gregory-Leibniz* (π = 4/1 – 4/3 + 4/5 - 4/7 + 4/9 - 4/11 ...).

Il thread esce dal ciclo quando una delle due condizioni seguenti risulta verificata:

- il thread è stato interrotto
- la differenza tra il valore stimato di π ed il valore Math.PI (della libreria JAVA) è minore di accuracy

Assignment 02 - Ufficio Postale

Simulare il flusso di clienti in un ufficio postale che ha 4 sportelli. Nell'ufficio esiste:

- un'ampia sala d'attesa in cui ogni persona può entrare liberamente. Quando entra, ogni persona prende il numero dalla numeratrice e aspetta il proprio turno in questa sala.
- una seconda sala, meno ampia, posta davanti agli sportelli, in cui possono essere presenti al massimo k persone (oltre alle persone servite agli sportelli)
- una persona si mette quindi prima in coda nella prima sala, poi passa nella seconda sala.
- ogni persona impiega un tempo differente per la propria operazione allo sportello. Una volta terminata l'operazione, la persona esce dall'ufficio

Scrivere un programma in cui:

- l'ufficio viene modellato come una classe JAVA, in cui viene attivato un ThreadPool di dimensione uguale al numero degli sportelli
- la coda delle persone presenti nella sala d'attesa è gestita esplicitamente dal programma
- la seconda coda (davanti agli sportelli) è quella gestita implicitamente dal ThreadPool
- ogni persona viene modellata come un task, un task che deve essere assegnato ad uno dei thread associati agli sportelli
- si preveda di far entrare tutte le persone nell'ufficio postale, all'inizio del programma

Assignment 03 - Gestione di un lab di informatica

Il laboratorio di Informatica del Polo Marzotto è utilizzato da tre tipi di utenti, studenti, tesisti e professori ed ogni utente deve fare una richiesta al tutor per accedere al laboratorio. I computers del laboratorio sono numerati da 1 a 20. Le richieste di accesso sono diverse a seconda del tipo dell'utente:

- i professori accedono in modo esclusivo a tutto il laboratorio, poichè hanno necessità di utilizzare tutti i computers per effettuare prove in rete.
- i tesisti richiedono l'uso esclusivo di un solo computer, identificato dall'indice i, poichè su quel computer è istallato un particolare software necessario per lo sviluppo della tesi.
- gli studenti richiedono l'uso esclusivo di un qualsiasi computer.

I professori hanno priorità su tutti nell'accesso al laboratorio, i tesisti hanno priorità sugli studenti.

Nessuno può essere interrotto mentre sta usando un computer. Scrivere un programma JAVA che simuli il comportamento degli utenti e del tutor. Il programma riceve in ingresso il numero di studenti, tesisti e professori che utilizzano il laboratorio ed attiva un thread per ogni utente. Ogni utente accede k volte al laboratorio, con k generato casualmente. Simulare l'intervallo di tempo che intercorre tra un accesso ed il successivo e l'intervallo di permanenza in laboratorio

mediante il metodo sleep. Il tutor deve coordinare gli accessi al laboratorio. Il programma deve terminare quando tutti gli utenti hanno completato i loro accessi al laboratorio.

Assignment 04 - Laboratorio di informatica con Monitor

Risolvere il problema della simulazione del Laboratorio di informatica, dell'assignment precedente, utilizzando il costrutto di Monitor.

Assignment 05 - File Crawler

Si scriva un programma JAVA che

- riceve in input un filepath che individua una directory D
- stampa le informazioni del contenuto di quella directory e, ricorsivamente, di tutti i file contenuti nelle sottodirectory di D

iI programma deve essere strutturato come segue:

- attiva un thread produttore ed un insieme di k thread consumatori
- il produttore comunica con i consumatori mediante una coda
- il produttore visita ricorsivamente la directory data ed, eventualmente tutte le sottodirectory e mette nella coda il nome di ogni directory individuata
- i consumatori prelevano dalla coda i nomi delle directories e stampano il loro contenuto (nomi dei file)
- la coda deve essere realizzata con una LinkedList. Ricordiamo che una Linked List non è una struttura thread-safe. Dalle API JAVA *"Note that the implementation is not synchronized. If multiple threads access a linked list concurrently, and at least one of the threads modifies the list structurally, it must be synchronized externally"*

Assignment 06 - Conti correnti

- Creare un file contenente oggetti che rappresentano i conti correnti di una banca. Ogni conto corrente contiene il nome del correntista ed una lista di movimenti. I movimenti registrati per un conto corrente sono relativi agli ultimi 2 anni, quindi possono essere molto numerosi.
  - per ogni movimento vengono registrati la data e la causale del movimento.
  - l'insieme delle causali possibili è fissato: Bonifico, Accredito, Bollettino, F24, PagoBancomat.
  - NB: Scrivete un programma che crei il file
- Scrivere un programma che rilegge il file e trova, per ogni possibile causale, quanti movimenti hanno quella causale.
  - progettare un'applicazione che attiva un insieme di thread. Uno di essi legge dal file gli oggetti “conto corrente” e li passa, uno per volta, ai thread presenti in un thread pool.
  - ogni thread calcola il numero di occorrenze di ogni possibile causale all'interno di quel conto corrente ed aggiorna un contatore globale.
  - alla fine il programma stampa per ogni possibile causale il numero totale di occorrenze.
- Utilizzare NIO per l'interazione con il file e JSON per la serializzazione

Assignment 07 - Mini Web Server

Scrivere un programma JAVA che implementa un server HTTP che gestisce richieste di trasferimento di file di diverso tipo (es. immagini jpeg, gif) provenienti da un browser web.

Il server

- sta in ascolto su una porta nota al client (*es. 6789*)
- gestisce richieste HTTP di tipo GET alla Request URL *localhost:port/filename*

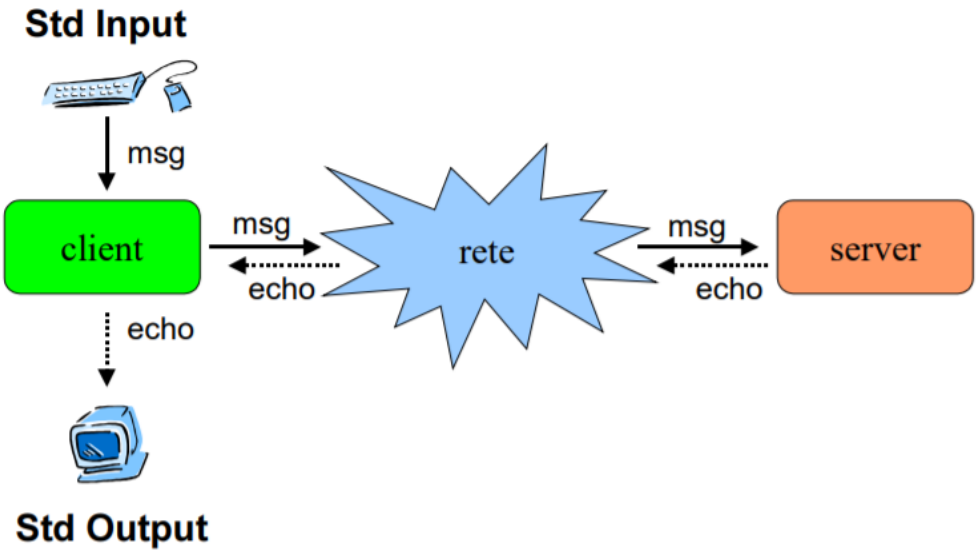
Ulteriori indicazioni

- le connessioni possono essere non persistenti.
- usare le classi Socket e ServerSocket per sviluppare il programma server
- per inviare al server le richieste, utilizzare un qualsiasi browser

Assignment 08 - NIO Echo Server

Scrivere un programma echo server usando la libreria java NIO e, in particolare, il Selector e canali in modalità non bloccante, e un programma echo client, usando NIO (va bene anche con modalità bloccante).

- Il server accetta richieste di connessioni dai client, riceve messaggi inviati dai client e li rispedisce (eventualmente aggiungendo "echoed by server" al messaggio ricevuto).
- Il client legge il messaggio da inviare da console, lo invia al server e visualizza quanto ricevuto dal server.



Assignment 09 - JAVA Pinger

PING è una utility per la valutazione delle performance della rete utilizzata per verificare la raggiungibilità di un host su una rete IP e per misurare il round trip time (RTT) per i messaggi spediti da un host mittente verso un host destinazione.

Lo scopo si questo assignment è quello di implementare un server PING ed un corrispondente client PING che consenta al client di misurare il suo RTT verso il server.

La funzionalità fornita da questi programmi deve essere simile a quella della utility PING disponibile in tutti i moderni sistemi operativi. La differenza fondamentale è che si utilizza UDP per la comunicazione tra client e server, invece del protocollo ICMP (Internet Control Message Protocol).

Inoltre, poiché l'esecuzione dei programmi avverrà su un solo host o sulla rete locale ed in entrambi i casi sia la latenza che la perdita di pacchetti risultano trascurabili, il server deve introdurre un ritardo artificiale ed ignorare alcune richieste per simulare la perdita di pacchetti

PING Client

- accetta due argomenti da linea di comando: **nome e porta del server**. Se uno o più argomenti risultano scorretti, il client termina, dopo aver stampato un messaggio di errore del tipo *ERR -arg x*, dove x è il numero dell'argomento.
- utilizza una comunicazione UDP per comunicare con il server ed invia 10 messaggi al server, con il seguente formato:

  ```
  PING seqno timestamp
  ```

  in cui *seqno* è il numero di sequenza del PING (tra 0-9) ed il *timestamp* (in millisecondi) indica quando il messaggio è stato inviato
- non invia un nuovo PING fino che non ha ricevuto l'eco del PING precedente, oppure è scaduto un timeout.
- Stampa ogni messaggio spedito al server ed il RTT del ping oppure un * se la risposta non è stata ricevuta entro 2 secondi
- Dopo che ha ricevuto la decima risposta (o dopo il suo timeout), il client stampa un riassunto simile a quello stampato dal PING UNIX

  ```
            ---- PING Statistics ----
  10 packets transmitted, 7 packets received, 30% packet loss
  round-trip (ms) min/avg/max = 63/190.29/290
  ```
- il RTT medio è stampato con 2 cifre dopo la virgola

PING Server

- è essenzialmente un echo server: rimanda al mittente qualsiasi dato riceve

- accetta un argomento da linea di comando: la porta, che è quella su cui è attivo il server. Se uno qualunque degli argomenti è scorretto, stampa un messaggio di errore del tipo *ERR -arg x*, dove x è il numero dell'argomento
- dopo aver ricevuto un PING, il server determina se ignorare il pacchetto (simulandone la perdita) o effettuarne l'eco. La probabilità di perdita di pacchetti di default è del 25%
- se decide di effettuare l'eco del PING, il server attende un intervallo di tempo casuale per simulare la latenza di rete
- stampa l'indirizzo IP e la porta del client, il messaggio di PING e l'azione intrapresa dal server in seguito alla sua ricezione (PING non inviato, oppure PING ritardato di x ms)

Assignment 10 - UDP Time Server

Definire un Server *TimeServer*, che

- invia su un gruppo di multicast *dategroup*, ad intervalli regolari, la data e l'ora.
- attende tra un invio ed il successivo un intervallo di tempo simulata mediante il metodo sleep().

L'indirizzo IP di *dategroup* viene introdotto da linea di comando.

Definire quindi un client *TimeClient* che si unisce a *dategroup* e riceve, per dieci volte consecutive, data ed ora, le visualizza, quindi termina.

Assignment 11 - Gestione Congresso

Si progetti un'applicazione Client/Server per la gestione delle registrazioni ad un congresso. L'organizzazione del congresso fornisce agli speaker delle varie sessioni un'interfaccia tramite la quale iscriversi ad una sessione, e la possibilità di visionare i programmi delle varie giornate del congresso, con gli interventi delle varie sessioni. Il server mantiene i programmi delle 3 giornate del congresso, ciascuno dei quali è memorizzato in una struttura dati, in cui ad ogni riga corrisponde una sessione (in tutto 12 per ogni giornata). Per ciascuna sessione vengono memorizzati i nomi degli speaker che si sono registrati (al massimo 5).

Il client può richiedere operazioni per:

- registrare uno speaker ad una sessione;
- ottenere il programma del congresso;

Il client inoltra le richieste al server tramite il meccanismo di **RMI**. Prevedere, per ogni possibile operazione una gestione di eventuali condizioni anomale (ad esempio la richiesta di registrazione ad una giornata e/o sessione inesistente oppure per la quale sono già stati coperti tutti gli spazi d'intervento)

Il client è implementato come un processo ciclico che continua a fare richieste sincrone fino ad esaurire tutte le esigenze utente. Stabilire una opportuna condizione di terminazione del processo di richiesta.

Va bene eseguire client, server e registry sullo stesso host.