LAB1: Socket in Java
Sistemi Distribuiti

Claudia Raibulet

claudia.Raibulet@unimib.it

2022-2023

Java Socket Classes

- java.net.Socket
- java.net.ServerSocket

API Socket System Calls

- Many calls are provided to access TCP and UDP services
- The most relevant ones are in the table below

Primitive	Meaning
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket, set the queue length
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Write	Send some data over the connection
Read	Receive some data over the connection
Close	Release the connection

Esercizio 1: Socket 1

- Si progetti un semplice sistema client server per la visualizzazione dei film a calendario in un cinema multisala.
- Per semplicita' assumiamo che tutti gli spettacoli inizino alla stessa ora ogni giorno (che viene identificato da un numero da 1, lunedı, a 7, domenica), in un unico spettacolo serale (quindi non ci interessa l'orario, perche` e' di default sempre lo stesso).
- Server e client si scambiano byte (non usate astrazioni di piu`alto livello).
- Per semplicita ora assumiamo che tutti i giorni abbiano la stessa programmazione.

Esercizio 1: Protocollo

- Il protocollo da implementaree il seguente:
- Il client si connette al server (leggendo in input l‘IP, e se non fornito si connette a localhost) su porta nota
- Quando ottiene una connessione, manda il comando identificato dal numero 1, seguito dal carattere \n, poi dal numero identificativo del giorno di interesse, e poi \n
- Il server controlla che il comando sia corretto (1), che il giorno sia compreso fra 1 e 7, e se tutto risulta corretto risponde con il numero 1 (che rappresenta ”ok”) seguito da \n, e poi invia una stringa che e’ la concatenazione dell‘insieme dei film in sala quel giorno.
- Se il messaggio dal cliente malformato, risponde con 0 seguito da`\n
- Il server chiude la connessione (e anche il client)
- NB: i film sono identificati solo dalla stringa del titolo

Esercizio 1.1: Miglioramenti

- Lo scambio di comandi e contenuti in questa modalita' di byte non e' molto agevole... usiamo qualche piccolo aiuto di Java, e leggiamo lo stream riga per riga!
- Cosa uso per evitare di cercare a mano il \n...?
- In questa versione, i titoli dei film sono inviati uno per uno, su righe diverse, e poi si richiede di usare uno specifico comando (ad esempio "2-close") per indicare che la lista sia finita.

Esercizio 2: Socket 2

- Rendiamo piu realistico l‘esempio del cinema.
- Dopo che il client ha chiesto l‘elenco dei film del giorno, cerca sempre di prenotare dei posti per uno dei film in programmazione.
- Il server, se ha disponibilita, accetta la prenotazione (e diminuisce i posti disponibili per quel giorno per quel film).
- La sequenza prevista e’ quindi prima la richiesta della lista, e poi la prenotazione, e i due comandi sfruttano una sola connessione client -server.
- Iniziate a rendere la programmazione dei giorni differente, e create delle classi e strutture per memorizzare le sale e i film in programmazione, modellando anche il massimo numero di posti di una sala, e le prenotazioni per quel film in un dato giorno.
- Suggerimenti: potete usare unaHashMap<Integer,ArrayList<Sala>>o qualcosa di simile.

Esercizio 2: Protocollo

- Ipotizziamo di usare la lettura/scrittura per linea
- Il client si connette al server (leggendo in input l'IP) su porta nota
- Quando ottiene una connessione, manda il comando 1, poi il giorno di interesse
- Il server risponde come nel vecchio protocollo (in piu, se non ha titoli in`programmazione quel giorno risponde con -1)
- se il server ha risposto 0 o -1, il client termina
- se il server ha risposto 1, il client seleziona random uno dei film in programmazione, e un numero di posti random, e chiede di prenotare mandando:
  - 2 (comando)
  - numero del giorno desiderato
  - titolo del film desiderato
  - numero di posti desiderato
- se i dati inviati sono sensati, e c'e disponibilita' di posti, il server risponde con 1 (e` aggiorna internamente la disponibilita' per quel film). Se i dati sono errati risponde con 0, se non ci sono film in programmazione risponde con -1. Poi chiude la connessione

Esercizio 3: Da consegnare offline

- Si realizzi un sistema client server per la gestione di un catalogo online di libri. Il server tiene traccia di un elenco di autori, identificati univocamente dal loro nome e cognome (un‘unica stringa), cui viene anche associato un id unico (creato automaticamente dal server ogni volta che un client aggiunge un nuovo autore).
- Per ogni autore poi si tiene elenco dei sui libri, semplicemente identificati dal titolo.
- Assumiamo che non esistano due autori con lo stesso nome, e che non sia possibile per un autore pubblicare due libri con lo stesso titolo.

Esercizio 3: Comandi

- Il server accetta un solo comando per ogni connessione da un client, con la seguentesintassi:
- getAllAuthors: restituisce, linea per linea, i nomi degli autori
- getAuthorId: restituisce l‘id dell‘autore, se in elenco, o -1
- addAuthor: prende come successivo input il nome dell‘autore. Restituisce l‘id che gli viene assegnato, se nuovo, o -1 se gia presente
- addBook: seguito dal nome dell‘autore e poi dal titolo del libro. Restituisce true se l‘inserimento va a buon fine, false altrimenti
- getAllBooks: restituisce l‘elenco di tutti i libri, come coppia ”autore: titolo”
- getBooksOf: prende come successivo input il nome dell‘autore. Restituisce l‘elenco dei libri di quell‘autore, o -1 in caso di autore non presente

Esercizio 3: Server e Client

- Realizzare il server, che espone come attributo static pubblico la porta cui e' in ascolto.
- Realizzare un client che si connette al server e che chiede all'utente che operazione vuole eseguire, e i relativi parametri (se necessari), e che accetta comandi dall'utente finche` l'utente non seleziona uno specifico comando di terminazione.