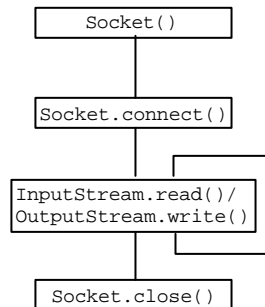


Implementazione di client e server in java

Avendo chiari i concetti delle socket in C
risulta tutto molto semplificato

Basta ricordare che un oggetto di classe
Socket è un peer di connessione con dei
“metodi attorno”



creazione socket

connessione

scambio dati

chiusura canale

Differenze rispetto al C

- Non abbiamo più il concetto di dominio (Java implementa solo indirizzi internet)
- Non dobbiamo più preoccuparci di gestire casi particolari e conversioni
“localhost” è identico a “127.0.0.1”
- Non ci sono più problemi di conversione di formato per i numeri, la virtual machine può essere considerato come un sistema operativo, quindi tutti i tipi hanno rappresentazione identica

Socket()

```
import java.net.Socket;  
  
Socket s = new Socket();
```

InetSocketAddress()

sottoclasse di SocketAddress

JDK 1.4

```
import java.net.SocketAddress;  
import java.net.InetSocketAddress;  
  
String host;  
int port;  
  
SocketAddress sa =  
    new InetSocketAddress(host, port);
```

Socket.connect()

JDK 1.4

- È il corrispettivo della chiamata a connect sulla socket incapsulata dall'oggetto
- La struttura sockadd_in è sostituita dall'istanza di una sottoclasse della classe virtuale SocketAddress

```
Socket.connect(SocketAddress);
```

Socket.connect()

JDK 1.4

```
import java.net.Socket;  
import java.net.SocketAddress;  
import java.net.InetSocketAddress;  
  
Socket s = new Socket();  
String host;  
int port;  
SocketAddress sa =  
    new InetSocketAddress(host, port);  
s.connect(sa);
```

getservbyname()

UNIMPLEMENTED

Comporterebbe la lettura di file/configurazioni specifiche per ogni architettura, e java non ammette queste cose.

Risultato: dobbiamo sapere il numero di porta a priori

read / write

```
import java.net.Socket;
import java.io.InputStream;
import java.io.OutputStream;

Socket s = new Socket();
InputStream is = s.getInputStream();
OutputStream os = s.getOutputStream();
```

read / write

A differenza del C non è possibile fare le operazioni logiche di read a write usando direttamente la socket.

Dobbiamo estrarre da socket due stream, uno di input e l'altro di output ed usare quelli.

Socket.close()

```
import java.net.Socket;

Socket s = new Socket();
s.close();
```

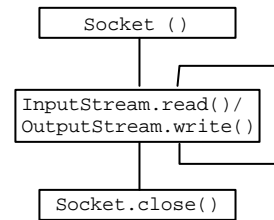
Le socket in java

BUONA NOTIZIA

qualcuno ha pensato al programmatore (finalmente), siamo svincolati da un sacco di dettagli tecnici ed esistono classi già pronte per client e server

BRUTTA NOTIZIA

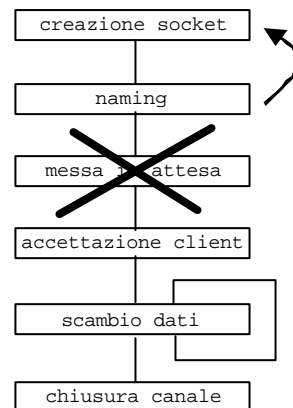
gestire le stringhe è **parecchio** più difficile, essendo java unicode

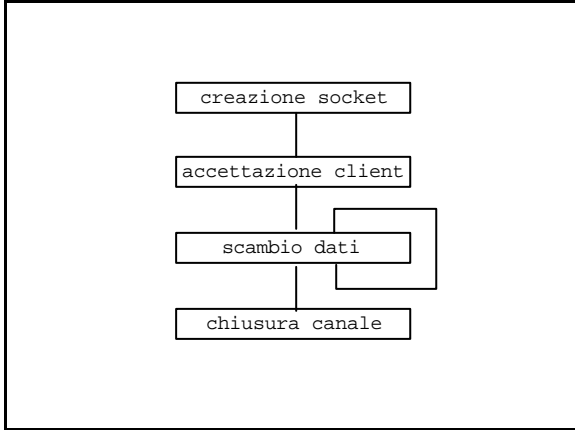


Rapidissimamente

```
import java.net.Socket;  
import java.net.SocketAddress;  
import java.net.InetSocketAddress;
```

```
String host;  
int port;  
Socket s = new Socket(host, port);
```

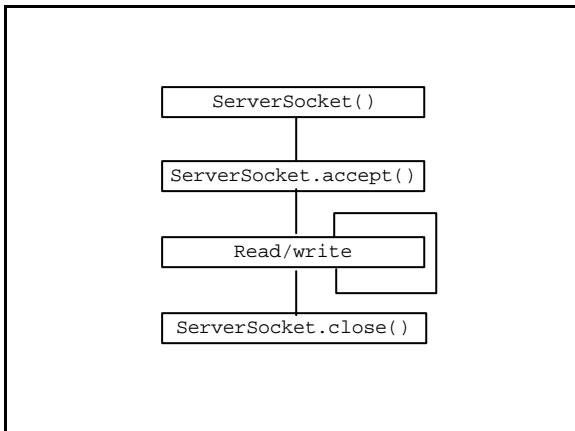




ServerSocket()

Abbiamo una classe specializzata per la socket sulla quale attendiamo i client

Possiamo crearla già collegata ad un indirizzo transport, oppure scollegata e poi fare bind in seguito



ServerSocket()

```

import java.net.ServerSocket;

int port;

ServerSocket ss = new ServerSocket(port);
  
```

JDK 1.4

ServerSocket()

```
import java.net.ServerSocket;
import java.net.SocketAddress;
import java.net.InetSocketAddress;

ServerSocket ss = new ServerSocket();
SocketAddress sa =
    new InetSocketAddress("localhost", port);
ss.bind(sa);
```

ServerSocket.accept()

Funzione identica al caso del C

```
import java.net.Socket;
import java.net.ServerSocket;

ServerSocket ss = new ServerSocket(0);
Socket data_sock = ss.accept();
```

getsockbyname()

- Quando faccio bind alla porta 0 posso recuperare l'indirizzo transport della socket con una chiamata a `getLocalPort`

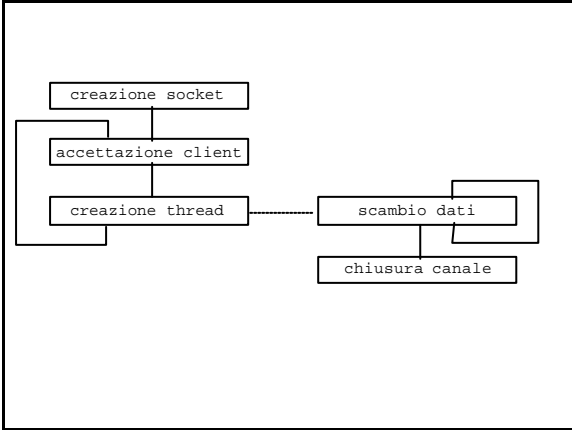
```
ServerSocket ss = new ServerSocket(0);
int port = ss.getLocalPort();
```

Server multithread

- È una soluzione analoga ad un server multiprocesso, l'unica differenza risiede nell'uso dei thread

IMPORTANTE

I thread condividono le variabili.

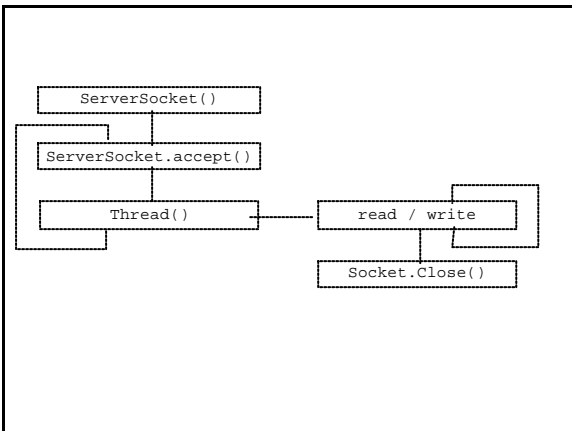


Thread()

La cosa migliore è che guardiate il "java tutorial"

<http://java.sun.com/docs/books/tutorial/>

Poi torneremo sull'argomento



Server concorrente

select()

UNIMPLEMENTED

Può essere simulata con i thread