

---

# Socket in Java

## Client-Server

---

Programmazione in Ambienti  
Distribuiti

V 1.3 © Marco Torchiano 2006

## Socket

---

- Socket implementa i socket
  - `Socket(String host, int port)`
    - ♦ Apre una connessione ad host e port specificati
  - `InputStream getInputStream()`
    - ♦ Restituisce lo stream di uscita
  - `OutputStream getOutputStream()`
    - ♦ Restituisce lo stream di ingresso
-

---

## Indirizzi

---

- InetAddress rappresenta gli indirizzi
    - ◆ socket.getInetAddress() remoto
    - ◆ socket.getLocalAddress() locale
  - String getHostName()
    - ◆ Restituisce il nome dell'host
- 

## Connessione – lato client

---

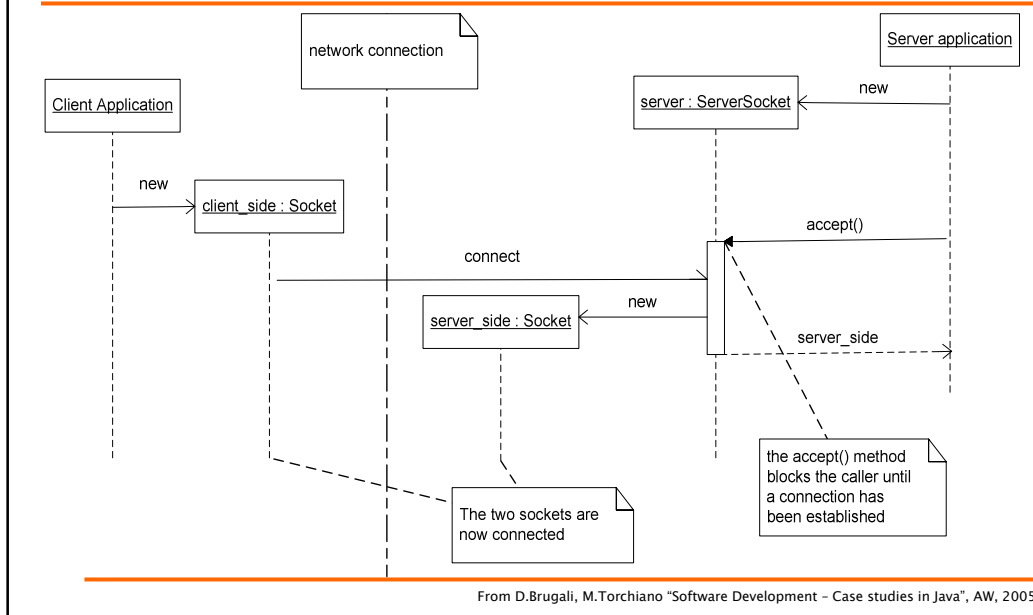
```
PrintWriter out;
BufferedReader in;
Socket sock;
sock = new Socket("host",3000);
System.out.println("Connected to " +
    sock.getInetAddress().getHostName());
out=new PrintWriter(
    sock.getOutputStream());
in=new BufferedReader(
    new InputStreamReader(
    sock.getInputStream()));
// ...
```

---

## Connessione – lato server

```
ServerSocket serv=new ServerSocket(port);
while(true){
    Socket conn = serv.accept();
    System.out.println("Connection from " +
        conn.getInetAddress().getHostName());
    out= new PrintWriter(
        conn.getOutputStream());
    in=new BufferedReader(
        new InputStreamReader(
            conn.getInputStream()));
    // ...
}
```

## Connessione tra socket



---

## Socket – lato server con thread

---

```
ServerSocket readServer = new ServerSocket(port);
while(true){
    Socket connection = readServer.accept();
    Server srv=new Server(connection)
}
```

```
class Server implements Runnable {
    Socket socket;
    public Server(Socket s){
        socket=s;
        Thread t = new Thread(this);
        t.start();
    }
    public void run(){ ... }
}
```

## Consigli

---

- **Data e ora:**

```
SimpleDateFormat dfmt = new SimpleDateFormat();
Date now = new Date();
System.out.println("Date:" + dfmt.format(now));
```

- **La semplice scrittura (es. println()) NON comporta l'invio di un pacchetto**

- ♦ Usare `outStream.flush()`
-

---

## Esempio

---

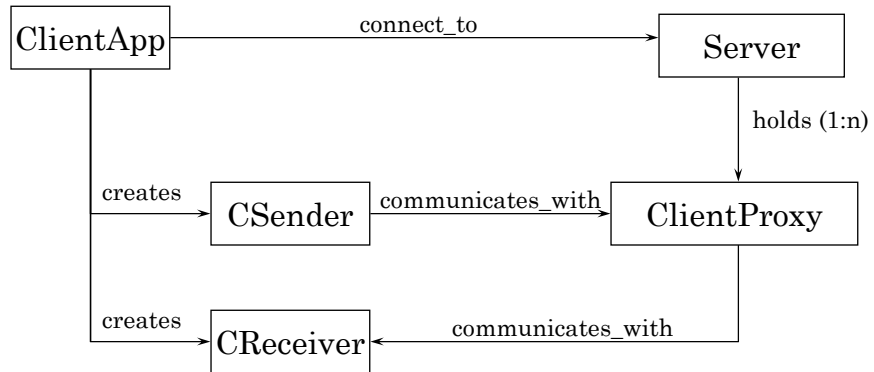
- Scrivere un programma in grado di collegarsi ad un web server
    - ♦ Richiede una pagina HTML
    - ♦ Stampa gli esiti della connessione
  - Scrivere un web server
    - ♦ Accetta connessioni da browser
    - ♦ Risponde usando il protocollo HTTP
    - ♦ Invia informazioni sulla richiesta HTTP
- 

## Comunicazione via socket

---

- Si vuole realizzare un sistema composto da un server al quale possono agganciarsi diversi client sia normali che applet.
  - Il server gestisce una stringa, che può essere modificata da un solo client per volta.
  - Ad ogni variazione della stringa, il server deve inviare a tutti i client connessi il nuovo valore.
-

## Relazione tra le classi



## Modello ad eventi

