



LABORATORIO di Reti di Calcolatori

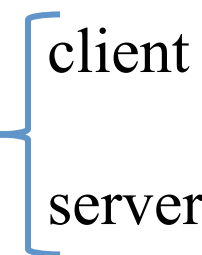
**Socket in linguaggio Java:
servizio connectionless**

Bibliografia

- ❖ slide della docente
- ❖ *testo di supporto*: D. Maggiorini, “Introduzione alla programmazione client-server”, Pearson Ed., 2009
 - ❑ cap.7 (tutto)
 - ❑ cap.8 (tutto)
- ❖ *Link utili*:
 - ❑ <http://docs.oracle.com/javase/tutorial/networking/index.html>
 - ❑ <http://docs.oracle.com/javase/6/docs/>

servizio connection-less

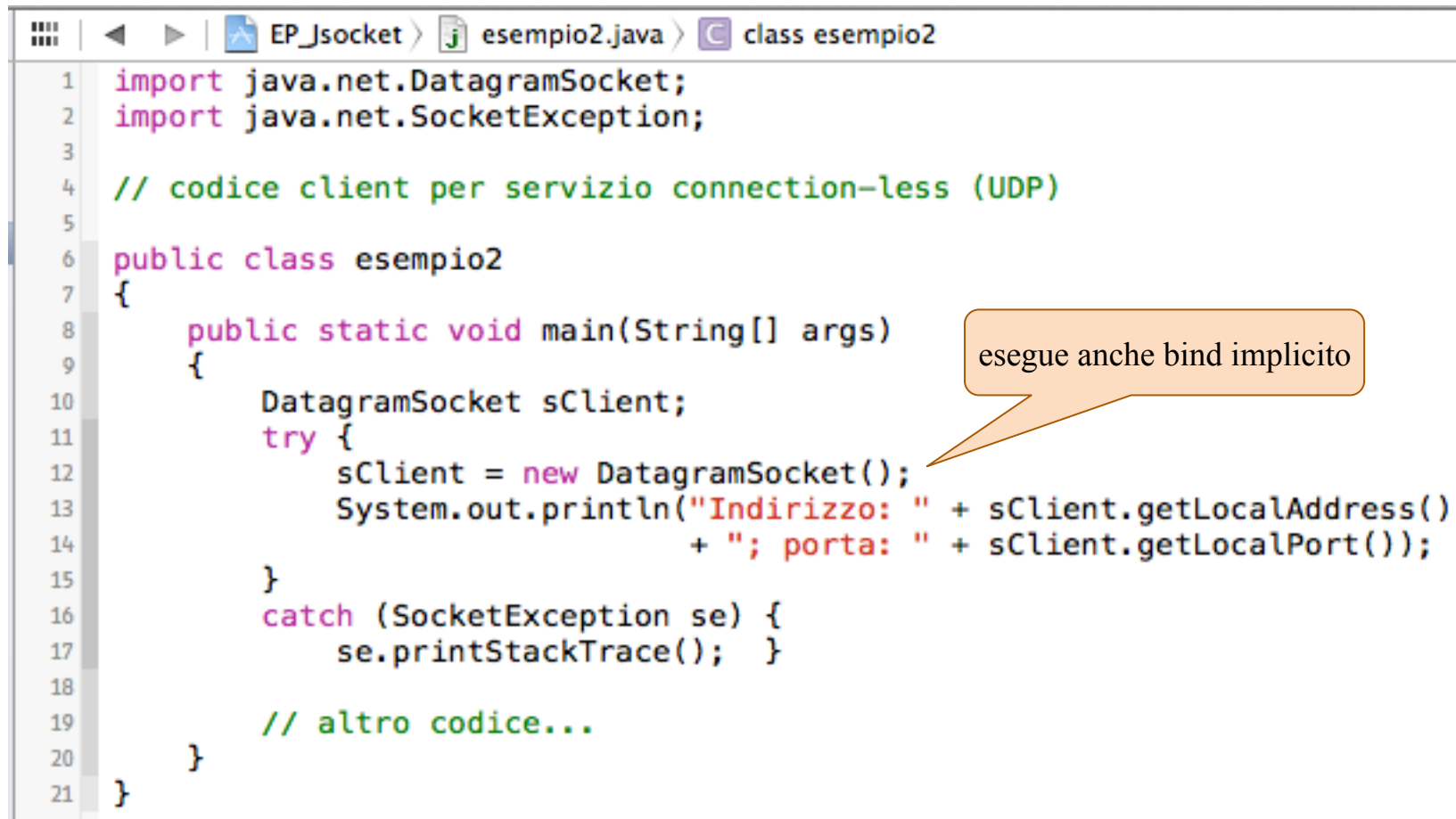
❖ ripasso fasi...

1. creazione socket
 2. binding → gestione indirizzi host + #porta
 3. scambio dati (*datagram*)
 4. chiusura
- 

❖ classi utilizzate: *package java.net*

- ❑ class DatagramSocket; DatagramPacket
- ❑ class InetAddress (indirizzi host); InetSocketAddress

1. creazione socket lato client



```
1 import java.net.DatagramSocket;
2 import java.net.SocketException;
3
4 // codice client per servizio connection-less (UDP)
5
6 public class esempio2
7 {
8     public static void main(String[] args)
9     {
10         DatagramSocket sClient;
11         try {
12             sClient = new DatagramSocket();
13             System.out.println("Indirizzo: " + sClient.getLocalAddress()
14                               + "; porta: " + sClient.getLocalPort());
15         }
16         catch (SocketException se) {
17             se.printStackTrace();
18         }
19         // altro codice...
20     }
21 }
```

esegue anche bind implicito

- ❖ situazione molto diversa da precedente: client e server sono simmetrici (senza connessione); **non ci sono due classi diverse...**

2. gestione indirizzi

- ❖ il server può usare in alternativa il costruttore `DatagramSocket(int port)` per specificare porta ben nota
- ❖ quando il client deve specificare l'indirizzo del server a cui inviare dati:
 - ❑ medesime classi per la gestione indirizzi già viste a proposito del caso connesso
- ❖ complichiamolo un po': i dati del server con cui comunicare possono essere passati da linea di comando
 - ❑ così evitiamo ricompilazione di esempio connesso...

1-2. parametrizzato

```
1 import java.net.DatagramSocket;
2 import java.net.SocketException;
3
4 // codice client per servizio connection-less (UDP)
5
6 public class esempio2
7 {
8     public static void main(String[] args)
9     {
10         DatagramSocket sClient;
11         try {
12             // parametri default server
13             String nome_host = "localhost";
14             int porta = 7000;
15
16             // acquisizione parametri server
17             if (args.length != 2)
18             {
19                 throw new IllegalArgumentException ("num.parametri non corretto");
20             }
21             nome_host = args[0];
22             porta = Integer.parseInt(args[1]);
23             if (porta <=0)
24             {
25                 throw new IllegalArgumentException ("porta non valida");
26             }
27
28             sClient = new DatagramSocket();
```

3. scambio dati

- ❖ bisogna costruire un *pacchetto* che include il payload e l'indirizzo del destinatario

- ❑ costruttore `DatagramPacket`(byte[] buf, int offset, int length, InetAddress addr, int port)

- nella sua forma più complessa

indirizzo destinazione

- ❖ invio pacchetto con metodo

`DatagramSocket.send(DatagramPacket dp)`

- ❖ ricezione pacchetto con metodo

`DatagramSocket.receive(DatagramPacket dp)`

- ❑ bloccante finché non è ricevuto l'intero datagram
 - ❑ dp è la struttura in cui è inserito il pacchetto ricevuto

3. costruzione e invio pacchetto

```
1 import java.net.DatagramSocket;  
2 import java.net.DatagramPacket;  
3 import java.net.SocketException;  
4 import java.net.InetSocketAddress;  
5 import java.io.BufferedReader;  
6 import java.io.InputStreamReader;
```

con import di tutti gli opportuni package...

```
35  
36 InetSocketAddress isa = new InetSocketAddress(nome_host,porta);  
37 InputStreamReader tastiera = new InputStreamReader(System.in);  
38 BufferedReader br = new BufferedReader(tastiera);  
39 String frase = br.readLine();  
40 byte[] buffer = frase.getBytes();  
41 DatagramPacket dp = new DatagramPacket(buffer, buffer.length);  
42 dp.setSocketAddress(isa);  
43 sClient.send(dp);
```

❖ altri metodi utili per costruire il pacchetto:

- ❑ DatagramPacket.setAddress(InetAddress iaddr)
- ❑ DatagramPacket.setData(byte[] buf, int offset, int length)
- ❑ DatagramPacket.setPort(int port)

3. ricezione pacchetto

```
1 import java.net.DatagramSocket;
2 import java.net.SocketException;
3 import java.net.DatagramPacket;
4 import java.net.InetSocketAddress;
5 import java.net.InetAddress;
6
7 // codice server per servizio connection-less (UDP)
8
9 public class es2Srv
10 {
11     public static void main(String[] args)
12     {
13         DatagramSocket sSrv;
14         try {
15             sSrv = new DatagramSocket();
16             System.out.println("Indirizzo: " + sSrv.getLocalAddress() + "; porta: " + sSrv.getLocalPort());
17
18             int dim_buffer = 100;
19             byte[] buffer = new byte[dim_buffer];
20             DatagramPacket dpin = new DatagramPacket(buffer, dim_buffer);
21             sSrv.receive(dpin);
22             String stringa = new String(buffer, 0, dpin.getLength()); ←
23             System.out.println("ricevuto: " + stringa);
24             InetAddress ia = dpin.getAddress(); } ←
25             int porta = dpin.getPort();
26             System.out.println("Indirizzo " + ia.getHostAddress() + "; porta: " + porta);
27             sSrv.close();
28         }
29         catch (Exception e) {
30             e.printStackTrace(); }
31     }
32 }
33 }
```

e questo se ce lo ricordiamo è meglio...

ultime considerazioni

- ❖ lanciare server
 - ❑ output mostra porta scelta da S.O. per binding
- ❖ lanciare client passando come argomento la porta del server

```
Elena$ java esempio2 localhost 64640
```

- ❖ inserire nel client la stringa da inviare al server

```
Indirizzo: 0.0.0.0/0.0.0.0; porta: 64641  
prova frase invio
```

porta client

- ❖ output server

```
Indirizzo: 0.0.0.0/0.0.0.0; porta: 64640  
ricevuto: prova frase invio  
Indirizzo 127.0.0.1; porta: 64641
```

indirizzo server (IP any)

indirizzo client