

I socket

Corso di Laurea in Informatica, Programmazione Distribuita
Delfina Malandrino, dmalandrino@unisa.it
http://www.unisa.it/docenti/delfinamalandrino

1

Organizzazione della lezione

2

- Programmazione con i socket
  - Socket TCP
  - Stream
- Alcuni esempi di uso dei socket
  - Hello World
  - Un Registro di nomi con architettura client-server
- Conclusioni

2

Obiettivi della lezione

3

- Introdurre i socket TCP e la loro programmazione in Java

3

Organizzazione della lezione

4

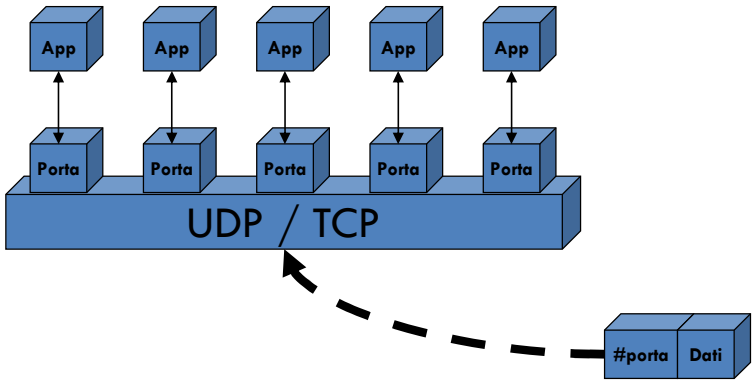
- Programmazione con i socket
  - Socket TCP
  - Stream
- Alcuni esempi di uso dei socket
  - Hello World
  - Un Registro di nomi con architettura client-server
- Conclusioni

4

Richiami di TCP: porte

5

- I protocolli TCP e UDP usano le porte per mappare i dati in ingresso con un particolare processo attivo su un computer
- Ogni socket è legato a un numero di porta così che il livello TCP può identificare l'applicazione a cui i dati devono essere inviati



5

Socket: definizione

6

- A livello programmatico, un Socket è definito come un "identificativo univoco che rappresenta un canale di comunicazione attraverso cui l'informazione è trasmessa" [RFC 147]
- La comunicazione basata su socket è indipendente dal linguaggio di programmazione
- Client e server devono concordare solo su protocollo (TCP o UDP) e numero di porta

6

I socket TCP in Java

7

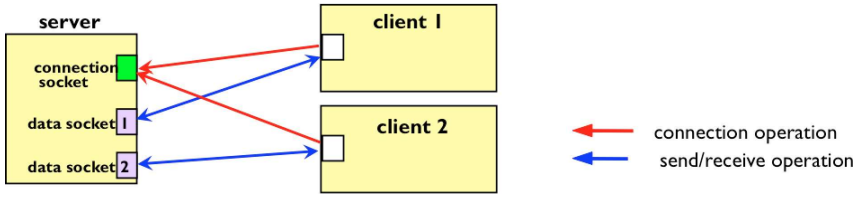
- Java fornisce le API per i socket in `java.net`
- Socket TCP:
  - astrazione fornita dal software di rete
  - permette di ricevere e trasmettere flussi dati
- Comunicazione bidirezionale
- Socket come endpoint (per il programmatore)
  - caratterizzato da un indirizzo IP e da una porta
- Client-server
  - naturale suddivisione dei compiti
    - chi in attesa di connessioni (server) e chi cerca di connettersi ad un servizio
  - asimmetrica

7

Come funzionano i socket

8

- I socket su stream sono supportati da due classi:
  - **`ServerSocket:`**
    - per accettare connessioni (socket di connessione)
  - **`Socket:`**
    - per scambio di dati (socket di dati)



8

I metodi di `Socket`

9

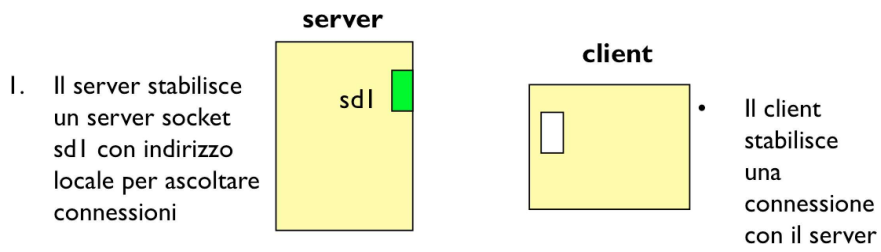
- **`ServerSocket(int port)`**
  - Crea un server socket su una specifica porta
- **`Socket accept()throws IOException`**
  - Aspetta connessioni sul ServerSocket e le accetta. Il metodo è bloccante fino a quando non viene fatta una connessione
- **`public void close() throws IOException`**
  - Chiude il socket
- **`void setSoTimeout(int timeout)throws SocketException`**
  - Setta un timeout (in ms) per una call ad accept. Se il tempo passa senza una connessione, viene lanciata una eccezione `java.io.InterruptedIOException`

9

La costruzione di uno stream socket - 1

10

- sd1 – socket di ascolto



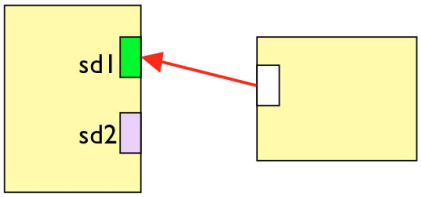
10

La costruzione di uno stream socket - 2

11

- sd2 – socket di connessione
- usato per lo scambio dei dati con un client

2. Il server accetta una connessione e crea un nuovo socket per dati sd2



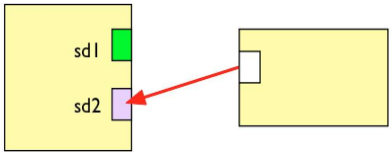
11

La costruzione di uno stream socket - 3

12

- Scambio di dati sul socket di connessione sd2
- Il client invia dati

3. Il server effettua una receive



- Il client effettua una send

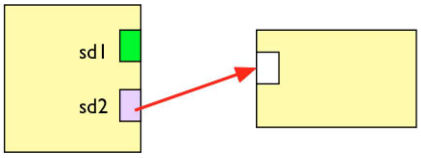
12

La costruzione di uno stream socket - 4

13

- Scambio di dati sul socket di connessione sd2
- Il server invia dati

4. Il server invia la risposta



- Il client effettua una receive

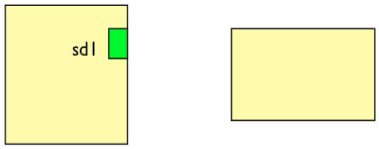
13

La costruzione di uno stream socket - 5

14

- Il server chiude la connessione

5. Quando il protocollo termina, il server chiude ed elimina il socket sd2



- Il client chiude il socket

14

Organizzazione della lezione

15

- Programmazione con i socket
  - Socket TCP
  - Stream
- Alcuni esempi di uso dei socket
  - Hello World
  - Un Registro di nomi con architettura client-server
- Conclusioni

15

Cosa fanno i programmi?

16

- "Prendi dati, fanne qualcosa, memorizza il risultato"
- Gli stream sono una maniera per prendere dati e trasferirli da qualche altra parte
  - i device sono spesso trattati come stream
- Esempio: un file viene visto come uno stream, un socket viene visto come uno stream, etc.

16

Cosa è uno stream?

17

- Uno stream è una sequenza ordinata di byte
  - **`java.io.InputStream`**: dati che da una sorgente esterna possono essere usati dal programma

```
int read()
int read(byte[]b)
int read(byte[] b,int off,int len)
void mark(int readlimit)
boolean markSupported()
void reset()
void close()
int available()
long skip(long n)
```

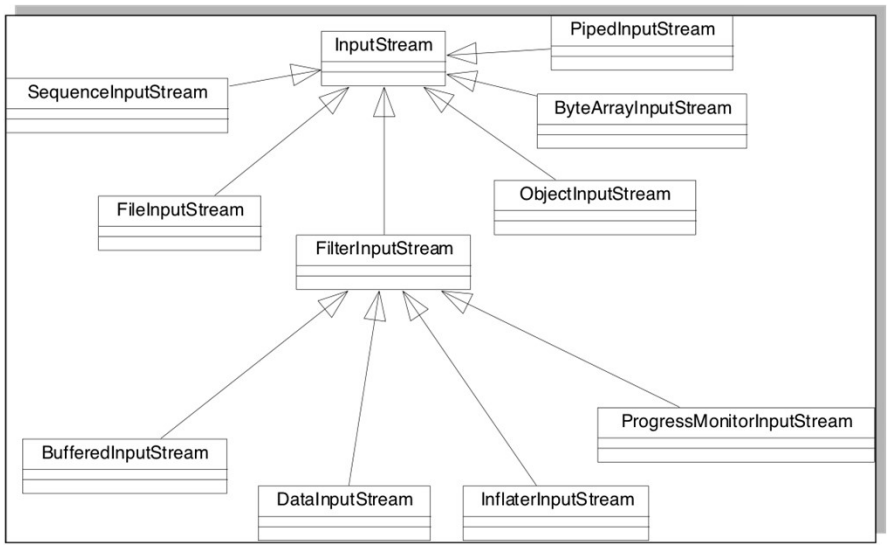
  - **`java.io.OutputStream`**: dati che il programma può inviare

```
void close()
void flush()
void write(byte[]b)
void write(byte[]b,int off,int len)
void write(int b)
```

17

Cosa è uno stream?

18



18

Stream di Input e di Output

19

- Gli stream sono come i carabinieri della famosa barzelletta: sono sempre presenti in coppia!
  - uno sa leggere (dallo stream) e l'altro sa scrivere (sullo stream)
- Per ogni tipo di InputStream (tranne SequenceInputStream) c'è il corrispondente OutputStream associato
  - FileInputStream ⇒ FileOutputStream
  - DataInputStream ⇒ DataOutputStream

19

Alcuni metodi utili di Stream

20

- Oggetto della classe Socket: il metodo getInputStream()
  - restituisce l'input stream associato al socket
  - esiste anche la versione per l'output stream
- Oggetto della classe ObjectInputStream: il metodo readObject()
  - bloccante
  - restituisce l'oggetto letto dallo stream
  - necessario il casting
- Metodo writeObject() di ObjectOutputStream

20

Organizzazione della lezione

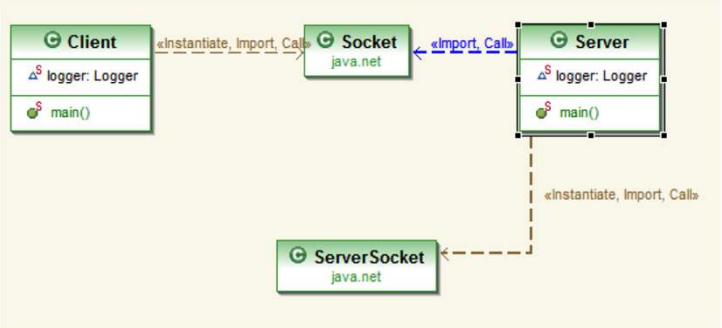
21

- Programmazione con i socket
  - Socket TCP
  - Stream
- Alcuni esempi di uso dei socket
  - Hello World
  - Un Registro di nomi con architettura client-server
- Conclusioni

21

Il diagramma delle classi

22



22

Il client

23

Import

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

 public static void main(String args[]) {
  try{
   Socket socket = new Socket ("localhost", 9000);
   ObjectOutputStream out = new
      ObjectOutputStream(socket.getOutputStream());
   ObjectInputStream in = new
      ObjectInputStream(socket.getInputStream());
   out.writeObject("Delfina");
   System.out.println(in.readObject());
   socket.close();
  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

23

Il client

24

Import

Definizione classe

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

 public static void main(String args[]) {
  try{
   Socket socket = new Socket ("localhost", 9000);
   ObjectOutputStream out = new
      ObjectOutputStream(socket.getOutputStream());
   ObjectInputStream in = new
      ObjectInputStream(socket.getInputStream());
   out.writeObject("Delfina");
   System.out.println(in.readObject());
   socket.close();
  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

24

Il client

25

Import

Definizione classe

Logger

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

 public static void main(String args[]) {
  try{
   Socket socket = new Socket ("localhost", 9000);
   ObjectOutputStream out = new
      ObjectOutputStream(socket.getOutputStream());
   ObjectInputStream in = new
      ObjectInputStream(socket.getInputStream());
   out.writeObject("Delfina");
   System.out.println(in.readObject());
   socket.close();
  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

25

Il client

26

Import

Definizione classe

Logger

Socket connessione

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

 public static void main(String args[]) {
  try{
   Socket socket = new Socket ("localhost", 9000);
   ObjectOutputStream out = new
      ObjectOutputStream(socket.getOutputStream());
   ObjectInputStream in = new
      ObjectInputStream(socket.getInputStream());
   out.writeObject("Delfina");
   System.out.println(in.readObject());
   socket.close();
  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

26

Il client

27

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

 public static void main(String args[]) {
  try{
   Socket socket = new Socket ("localhost", 9000);
   ObjectOutputStream out = new
      ObjectOutputStream(socket.getOutputStream());
   ObjectInputStream in = new
      ObjectInputStream(socket.getInputStream());
   out.writeObject("Delfina");
   System.out.println(in.readObject());
   socket.close();
  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

- Import
- Definizione classe
- Logger
- Socket connessione
- Stream output

27

Il client

28

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

 public static void main(String args[]) {
  try{
   Socket socket = new Socket ("localhost", 9000);
   ObjectOutputStream out = new
      ObjectOutputStream(socket.getOutputStream());
   ObjectInputStream in = new
      ObjectInputStream(socket.getInputStream());
   out.writeObject("Delfina");
   System.out.println(in.readObject());
   socket.close();
  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

- Import
- Definizione classe
- Logger
- Socket connessione
- Stream output
- Stream input

28

Il client

29

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

  public static void main(String args[]) {
   try{
    Socket socket = new Socket ("localhost", 9000);
    ObjectOutputStream out = new
       ObjectOutputStream(socket.getOutputStream());
    ObjectInputStream in = new
       ObjectInputStream(socket.getInputStream());
    out.writeObject("Delfina");
    System.out.println(in.readObject());
    socket.close();
   }catch(EOFException e) {
      logger.severe("Problemi con la connessione:"+
              e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
              t.getMessage());
     t.printStackTrace();
  }
 }
}
```

- Import
- Definizione classe
- Logger
- Socket connessione
- Stream output
- Stream input
- Si scrive il proprio nome

29

Il client

30

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

  public static void main(String args[]) {
   try{
    Socket socket = new Socket ("localhost", 9000);
    ObjectOutputStream out = new
       ObjectOutputStream(socket.getOutputStream());
    ObjectInputStream in = new
       ObjectInputStream(socket.getInputStream());
    out.writeObject("Delfina");
    System.out.println(in.readObject());
    socket.close();
  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
              e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
              t.getMessage());
     t.printStackTrace();
  }
 }
}
```

- Import
- Definizione classe
- Logger
- Socket connessione
- Stream output
- Stream input
- Si scrive il proprio nome
- si attende e stampa la risposta

30

Il client

31

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

  public static void main(String args[]) {
   try{
    Socket socket = new Socket ("localhost", 9000);
    ObjectOutputStream out = new
       ObjectOutputStream(socket.getOutputStream());
    ObjectInputStream in = new
       ObjectInputStream(socket.getInputStream());
    out.writeObject("Delfina");
    System.out.println(in.readObject());
    socket.close();
  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

- Import
- Definizione classe
- Logger
- Socket connessione
- Stream output
- Stream input
- Si scrive il proprio nome
- si attende e stampa la risposta
- si chiude il socket

31

Il client

32

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

 publicstaticvoidmain(String args[]) {
  try{
   Socket socket =newSocket ("localhost", 9000);
   ObjectOutputStream out =new
      ObjectOutputStream(socket.getOutputStream());
   ObjectInputStream in =new
      ObjectInputStream(socket.getInputStream());
   out.writeObject("Delfina");
   System.out.println(in.readObject());
   socket.close();
  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

- Import
- Definizione classe
- Logger
- Socket connessione
- Stream output
- Stream input
- Si scrive il proprio nome
- si attende e stampa la risposta
- si chiude il socket
- Eccezione di connessione

32

Il client

33

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class Client {
 static Logger logger = Logger.getLogger("global");

  public static void main(String args[]) {
   try{
    Socket socket = new Socket ("localhost", 9000);
    ObjectOutputStream out = new
       ObjectOutputStream(socket.getOutputStream());
    ObjectInputStream in = new
       ObjectInputStream(socket.getInputStream());
    out.writeObject("Delfina");
    System.out.println(in.readObject());
    socket.close();
   }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
   }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
   }
  }
}
```

Import

Definizione classe

Logger

Socket connessione

Stream output

Stream input

Si scrive il proprio nome

si attende e stampa la risposta

si chiude il socket

Eccezione di connessione

Altre eccezioni

33

Il server

34

Import

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;

public class Server {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
  try{
   ServerSocket serverSocket = new
         ServerSocket(9000);
   logger.info("Socketok,accetto conn...");

   Socket socket = serverSocket.accept();
   logger.info("Accettata una connessione....");

   ObjectOutputStream oS = new ObjectOutputStream
           (socket.getOutputStream());

   ObjectInputStream iS = new ObjectInputStream
           (socket.getInputStream());

    //...
```

34

Il server

35

Import

Classe

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;

public class Server {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
  try{
   ServerSocket serverSocket = new
         ServerSocket(9000);
   logger.info("Socketok,accetto conn...");

   Socket socket = serverSocket.accept();
   logger.info("Accettata una connessione....");

   ObjectOutputStream oS = new ObjectOutputStream
           (socket.getOutputStream());

   ObjectInputStream iS = new ObjectInputStream
           (socket.getInputStream());

    //...
```

35

Il server

36

Import

Classe

Logger globale

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;

public class Server {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
  try{
   ServerSocket serverSocket = new
         ServerSocket(9000);
   logger.info("Socketok,accetto conn...");

   Socket socket = serverSocket.accept();
   logger.info("Accettata una connessione....");

   ObjectOutputStream oS = new ObjectOutputStream
           (socket.getOutputStream());

   ObjectInputStream iS = new ObjectInputStream
           (socket.getInputStream());

    //...
```

36

Il server

37

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;

public class Server {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
 try{
   ServerSocket serverSocket = new
         ServerSocket(9000);
   logger.info("Socketok,accetto conn...");

   Socket socket = serverSocket.accept();
   logger.info("Accettata una connessione....");

   ObjectOutputStream oS = new ObjectOutputStream
           (socket.getOutputStream());

   ObjectInputStream iS = new ObjectInputStream
           (socket.getInputStream());

    //...
```

- Import
- Classe
- Logger globale
- Main

37

Il server

38

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;

public class Server {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args)
 try{
   ServerSocket serverSocket = new
          ServerSocket(9000);
   logger.info("Socket ok,accetto conn...");

   Socket socket = serverSocket.accept();
   logger.info("Accettata una connessione....");

   ObjectOutputStream oS = new ObjectOutputStream
           (socket.getOutputStream());

   ObjectInputStream iS = new ObjectInputStream
           (socket.getInputStream());

    //...
```

- Import
- Classe
- Logger globale
- Main
- Dichiarazione del socket di connessione

38

Il server

39

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;

public class Server {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
  try{
   ServerSocket serverSocket = new
         ServerSocket(9000);
   logger.info("Socketok,accetto conn...");

   Socket socket = serverSocket.accept();
   logger.info("Accettata una connessione....");

   ObjectOutputStream oS = new ObjectOutputStream
           (socket.getOutputStream());

   ObjectInputStream iS = new ObjectInputStream
           (socket.getInputStream());

    //...
```

- Import
- Classe
- Logger globale
- Main
- Dichiarazione del socket di connessione
- Accettazione di connessioni

39

Il server

40

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;

public class Server {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
  try{
   ServerSocket serverSocket = new
         ServerSocket(9000);
   logger.info("Socketok,accetto conn...");

   Socket socket = serverSocket.accept();
   logger.info("Accettata una connessione....");

   ObjectOutputStream oS = new ObjectOutputStream
           (socket.getOutputStream());

   ObjectInputStream iS = new ObjectInputStream
           (socket.getInputStream());

    //...
```

- Import
- Classe
- Logger globale
- Main
- Dichiarazione del socket di connessione
- Accettazione di connessioni
- Quando c'è un client …

40

Il server

41

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;

public class Server {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
  try{
   ServerSocket serverSocket = new
         ServerSocket(9000);
   logger.info("Socketok,accetto conn...");

   Socket socket = serverSocket.accept();
   logger.info("Accettata una connessione....");

   ObjectOutputStream oS = new ObjectOutputStream
           (socket.getOutputStream());

   ObjectInputStream iS = new ObjectInputStream
           (socket.getInputStream());

    //...
```

- Import
- Classe
- Logger globale
- Main
- Dichiarazione del socket di connessione
- Accettazione di connessioni
- Quando c'è un client …
- …si prende l'output stream

41

Il server

42

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;

public class Server {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
  try{
   ServerSocket serverSocket = new
         ServerSocket(9000);
   logger.info("Socketok,accetto conn...");

   Socket socket = serverSocket.accept();
   logger.info("Accettata una connessione....");

   ObjectOutputStream oS = new ObjectOutputStream
           (socket.getOutputStream());

   ObjectInputStream iS = new ObjectInputStream
           (socket.getInputStream());

    //...
```

- Import
- Classe
- Logger globale
- Main
- Dichiarazione del socket di connessione
- Accettazione di connessioni
- Quando c'è un client …
- …si prende l'output stream
- …e l'input stream

42

Il server

43

```
   //...

   String nome= (String) iS.readObject();
   logger.info("Ricevuto:"+ nome);

   oS.writeObject("Hello"+ nome);
   socket.close();

  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

si riceve la stringa

43

Il server

44

```
   //...

   String nome= (String) iS.readObject();
   logger.info("Ricevuto:"+ nome);

   oS.writeObject("Hello"+ nome);
   socket.close();

  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

si riceve la stringa

si risponde salutando il client

44

Il server

45

```
   //...

   String nome= (String) iS.readObject();
   logger.info("Ricevuto:"+ nome);

   oS.writeObject("Hello"+ nome);
   socket.close();

  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

si riceve la stringa

si risponde salutando il client

eccezione di connessione

45

Il server

46

```
   //...

   String nome= (String) iS.readObject();
   logger.info("Ricevuto:"+ nome);

   oS.writeObject("Hello"+ nome);
   socket.close();

  }catch(EOFException e) {
     logger.severe("Problemi con la connessione:"+
             e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe("Lanciata Throwable:"+
             t.getMessage());
     t.printStackTrace();
  }
 }
}
```

si riceve la stringa

si risponde salutando il client

eccezione di connessione

altre eccezioni

46

Organizzazione della lezione

47

- Programmazione con i socket
  - Socket TCP
  - Stream
- Alcuni esempi di uso dei socket
  - Hello World
  - Un Registro di nomi con architettura client-server
- Conclusioni

47

Struttura

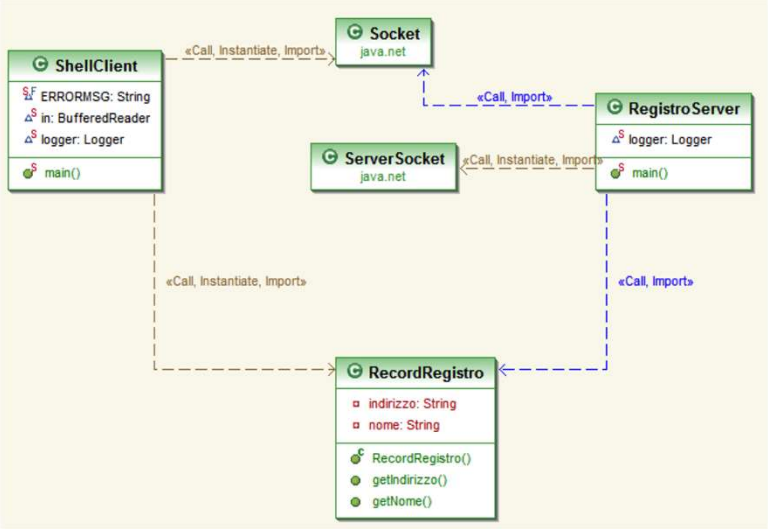
48

- Un semplice esempio
- Il server mantiene dei record (composti da nome e indirizzo)
- Il client può:
  - inserire un record: passando un oggetto da inserire
    - record verrà memorizzato
  - ricercare un record: passando un oggetto solamente con il campo nome inserito
    - ricerca di un record, restituzione di un valore (indirizzo)

48

Diagramma delle classi

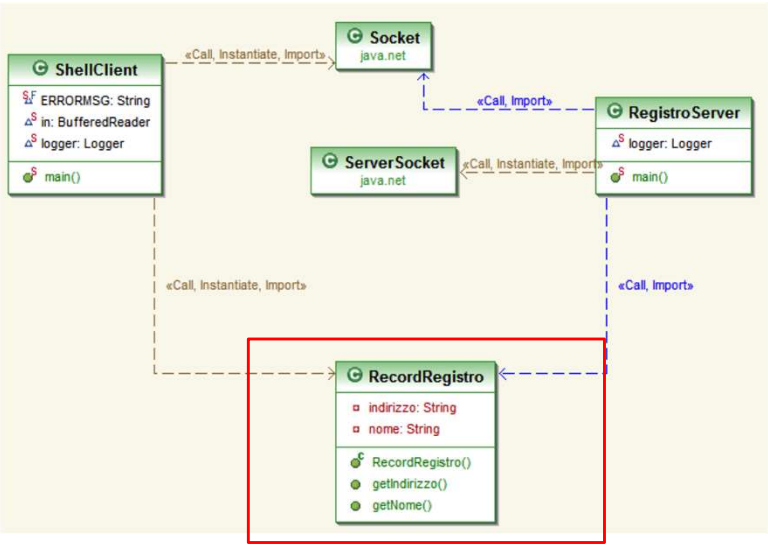
49



49

La classe RecordRegistro

50



50

La classe RecordRegistro

51

```
import java.io.Serializable;

public class RecordRegistro implements
      Serializable {
    private static final long serialVersionUID
                        = -4147133786465982122L;

    public RecordRegistro(String n, String i) {
        nome = n;
        indirizzo = i;
    }

    public String getNome() {
        return nome;
    }

    public String getIndirizzo() {
        return indirizzo;
    }

    private String nome;
    private String indirizzo;
}
```

Import

51

La classe RecordRegistro

52

```
import java.io.Serializable;

public class RecordRegistro implements
      Serializable {
    private static final long serialVersionUID
                        = -4147133786465982122L;

    public RecordRegistro(String n, String i) {
        nome = n;
        indirizzo = i;
    }

    public String getNome() {
        return nome;
    }

    public String getIndirizzo() {
        return indirizzo;
    }

    private String nome;
    private String indirizzo;
}
```

Import

Deve essere trasmessa sulla rete

52

La classe RecordRegistro

53

```
import java.io.Serializable;

public class RecordRegistro implements
      Serializable {
    private static final long serialVersionUID
                        = -4147133786465982122L;

    public RecordRegistro(String n, String i) {
        nome = n;
        indirizzo = i;
    }

    public String getNome() {
        return nome;
    }

    public String getIndirizzo() {
        return indirizzo;
    }

    private String nome;
    private String indirizzo;
}
```

Import

Deve essere trasmessa sulla rete

Versione della classe

53

La classe RecordRegistro

54

```
import java.io.Serializable;

public class RecordRegistro implements
      Serializable {
    private static final long serialVersionUID
                        = -4147133786465982122L;

    public RecordRegistro(String n, String i) {
        nome = n;
        indirizzo = i;
    }

    public String getNome() {
        return nome;
    }

    public String getIndirizzo() {
        return indirizzo;
    }

    private String nome;
    private String indirizzo;
}
```

Import

Deve essere trasmessa sulla rete

Versione della classe

Costruttore

54

La classe RecordRegistro

55

```
import java.io.Serializable;

public class RecordRegistro implements
      Serializable {
    private static final long serialVersionUID
                        = -4147133786465982122L;

    public RecordRegistro(String n, String i) {
        nome = n;
        indirizzo = i;
    }

    public String getNome() {
        return nome;
    }

    public String getIndirizzo() {
        return indirizzo;
    }

    private String nome;
    private String indirizzo;
}
```

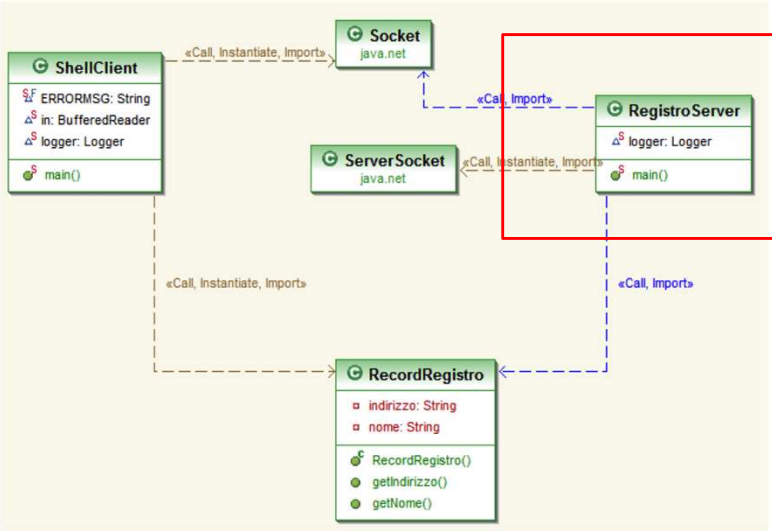
- Import
- Deve essere trasmessa sulla rete
- Versione della classe
- Costruttore

Il resto della classe non merita commenti ☺

55

La classe RegistroServer

56



56

La classe RegistroServer

57

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.util.logging.Logger;

public class RegistroServer {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
  HashMap<String, RecordRegistro> hash =
          new HashMap<String, RecordRegistro>();
  Socket socket = null;
  System.out.println ("In attesa...");
  try{
   ServerSocket serverSocket = new
                               ServerSocket(7000);

   while(true) {
    socket = serverSocket.accept();
    ObjectInputStream inStream = new
            ObjectInputStream
                    (socket.getInputStream());
    RecordRegistro record = (RecordRegistro)
                            inStream.readObject();
    //...
```

Import

57

La classe RegistroServer

58

```
import java.io. *;
import java.net. *;
import java.util. *;
import java.util.logging.Logger;

public class RegistroServer {
 static Logger logger = Logger.getLogger("global");

 public static void main(String[] args) {
  HashMap<String, RecordRegistro> hash =
          new HashMap<String, RecordRegistro>();
  Socket socket = null;
  System.out.println ("In attesa...");
  try{
   ServerSocket serverSocket = new
                               ServerSocket(7000);

   while(true) {
    socket = serverSocket.accept();
    ObjectInputStream inStream = new
            ObjectInputStream
                    (socket.getInputStream());
    RecordRegistro record = (RecordRegistro)
                            inStream.readObject();
    //...
```

Import

Classe

58

La classe RegistroServer

59

```
import java.io. *;
import java.net. *;
import java.util. *;
import java.util.logging.Logger;

public class RegistroServer {
 static Logger logger = Logger.getLogger("global");

  public static void main(String[] args) {
   HashMap<String, RecordRegistro> hash =
          new HashMap<String, RecordRegistro>();
   Socket socket = null;
   System.out.println ("In attesa...");
   try{
    ServerSocket serverSocket = new
                                ServerSocket(7000);

    while(true) {
     socket = serverSocket.accept();
     ObjectInputStream inStream = new
             ObjectInputStream
                   (socket.getInputStream());
     RecordRegistro record = (RecordRegistro)
                             inStream.readObject();
     //...
```

Import

Classe

Logger

59

La classe RegistroServer

60

```
import java.io. *;
import java.net. *;
import java.util. *;
import java.util.logging.Logger;

public class RegistroServer {
 static Logger logger = Logger.getLogger("global");

  public static void main(String[] args) {
   HashMap<String, RecordRegistro> hash =
          new HashMap<String, RecordRegistro>();
   Socket socket = null;
   System.out.println ("In attesa...");
   try{
    ServerSocket serverSocket = new
                                ServerSocket(7000);

    while(true) {
     socket = serverSocket.accept();
     ObjectInputStream inStream = new
             ObjectInputStream
                   (socket.getInputStream());
     RecordRegistro record = (RecordRegistro)
                             inStream.readObject();
     //...
```

Import

Classe

Logger

Mantiene le coppie nome, valore

60

La classe RegistroServer

61

```
import java.io. *;
import java.net. *;
import java.util. *;
import java.util.logging.Logger;

public class RegistroServer {
 static Logger logger = Logger.getLogger("global");

  public static void main(String[] args) {
   HashMap<String, RecordRegistro> hash =
          new HashMap<String, RecordRegistro>();
   Socket socket = null;
   System.out.println ("In attesa...");
   try{
    ServerSocket serverSocket = new
                                ServerSocket(7000);

   while(true) {
    socket = serverSocket.accept();
    ObjectInputStream inStream = new
            ObjectInputStream
                  (socket.getInputStream());
    RecordRegistro record = (RecordRegistro)
                            inStream.readObject();
    //...
```

Import

Classe

Logger

Mantiene le coppie nome, valore

Socket di connessione

61

La classe RegistroServer

62

```
import java.io. *;
import java.net. *;
import java.util. *;
import java.util.logging.Logger;

public class RegistroServer {
 static Logger logger = Logger.getLogger("global");

  public static void main(String[] args) {
   HashMap<String, RecordRegistro> hash =
          new HashMap<String, RecordRegistro>();
   Socket socket = null;
   System.out.println ("In attesa...");
   try{
    ServerSocket serverSocket = new
                                ServerSocket(7000);

   while(true) {
    socket = serverSocket.accept();
    ObjectInputStream inStream = new
            ObjectInputStream
                  (socket.getInputStream());
    RecordRegistro record = (RecordRegistro)
                            inStream.readObject();
    //...
```

Import

Classe

Logger

Mantiene le coppie nome, valore

Socket di connessione

Ciclo

62

La classe RegistroServer

63

```
import java.io. *;
import java.net. *;
import java.util. *;
import java.util.logging.Logger;

public class RegistroServer {
 static Logger logger = Logger.getLogger("global");

  public static void main(String[] args) {
   HashMap<String, RecordRegistro> hash =
          new HashMap<String, RecordRegistro>();
   Socket socket = null;
   System.out.println ("In attesa...");
   try{
    ServerSocket serverSocket = new
                                ServerSocket(7000);

   while(true) {
    socket = serverSocket.accept();
    ObjectInputStream inStream = new
            ObjectInputStream
                  (socket.getInputStream());
    RecordRegistro record = (RecordRegistro)
                           inStream.readObject();
    //...
```

Import

Classe

Logger

Mantiene le coppie nome, valore

Socket di connessione

Ciclo

accetta connessioni

63

La classe RegistroServer

64

```
import java.io. *;
import java.net. *;
import java.util. *;
import java.util.logging.Logger;

public class RegistroServer {
 static Logger logger = Logger.getLogger("global");

  public static void main(String[] args) {
   HashMap<String, RecordRegistro> hash =
          new HashMap<String, RecordRegistro>();
   Socket socket = null;
   System.out.println ("In attesa...");
   try{
    ServerSocket serverSocket = new
                                ServerSocket(7000);

   while(true) {
    socket = serverSocket.accept();
    ObjectInputStream inStream = new
            ObjectInputStream
                  (socket.getInputStream());
    RecordRegistro record = (RecordRegistro)
                           inStream.readObject();
    //...
```

Import

Classe

Logger

Mantiene le coppie nome, valore

Socket di connessione

Ciclo

accetta connessioni

crea stream input

64

La classe RegistroServer

66

```
import java.io. *;
import java.net. *;
import java.util. *;
import java.util.logging.Logger;

public class RegistroServer {
 static Logger logger = Logger.getLogger("global");

  public static void main(String[] args) {
   HashMap<String, RecordRegistro> hash =
          new HashMap<String, RecordRegistro>();
   Socket socket = null;
   System.out.println ("In attesa...");
   try{
    ServerSocket serverSocket = new
                                ServerSocket(7000);

   while(true) {
    socket = serverSocket.accept();
    ObjectInputStream inStream = new
            ObjectInputStream
                  (socket.getInputStream());
    RecordRegistro record = (RecordRegistro)
                           inStream.readObject();
    //...
```

Import

Classe

Logger

Mantiene le coppie nome, valore

Socket di connessione

Ciclo

accetta connessioni

crea stream input

legge l'oggetto (bloccante)

65

La classe RegistroServer

66

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e) {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket
    try{ socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
   }
  }

 }
}
```

Record non vuoto

66

La classe RegistroServer

67

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e) {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket
    try{ socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
   }
  }

 }
}
```

Record non vuoto

Record vuoto: ricerca

67

La classe RegistroServer

68

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e) {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket
    try{ socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
   }
  }

 }
}
```

Record non vuoto

Record vuoto: ricerca

Cerca il nome

68

La classe RegistroServer

69

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e) {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket
    try{ socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
    }
  }

 }
}
```

- Record non vuoto
- Record vuoto: ricerca
- Cerca il nome
- Crea l'output

69

La classe RegistroServer

70

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e)  {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t) {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket
    try{ socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
    }
  }
 }
}
```

- Record non vuoto
- Record vuoto: ricerca
- Cerca il nome
- Crea l'output
- Il record trovato (`null` se non c'è)

70

La classe RegistroServer

71

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e)  {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t)  {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket  try{
    socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
    }
   }
  }
}
```

- Record non vuoto
- Record vuoto: ricerca
- Cerca il nome
- Crea l'output
- Il record trovato (`null` se non c'è)
- Necessario

71

La classe RegistroServer

72

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e)  {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t)  {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket
    try{ socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
    }
   }
  }
}
```

- Record non vuoto
- Record vuoto: ricerca
- Cerca il nome
- Crea l'output
- Il record trovato (`null` se non c'è)
- Necessario
- Chiusura

72

La classe RegistroServer

73

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e)  {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t)  {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket
    try{ socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
    }
   }
  }
}
```

- Record non vuoto
- Record vuoto: ricerca
- Cerca il nome
- Crea l'output
- Il record trovato (`null` se non c'è)
- Necessario
- Chiusura
- Eccezione di chiusura socket

73

La classe RegistroServer

74

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e)  {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t)  {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket  try{
    socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
    }
   }
  }
}
```

- Record non vuoto
- Record vuoto: ricerca
- Cerca il nome
- Crea l'output
- Il record trovato (`null` se non c'è)
- Necessario
- Chiusura
- Eccezione di chiusura socket
- Altre eccezioni

74

La classe RegistroServer

75

```
    if(record.getIndirizzo()!=null) {//scrittura
      hash.put(record.getNome(), record);
    }else{//ricerca
      RecordRegistro res =
            hash.get(record.getNome());
      ObjectOutputStream outStream =new
              ObjectOutputStream
                    (socket.getOutputStream());
      outStream.writeObject(res);
      outStream.flush();
    }//fine else
    socket.close();
   }//fine while
  }catch(EOFException e)  {
     logger.severe(e.getMessage());
     e.printStackTrace();
  }catch(Throwable t)  {
     logger.severe(t.getMessage());
     t.printStackTrace();
  }
  finally{//chiusura del socket
    try{ socket.close();
    }catch(IOException e) {
      e.printStackTrace();
      System.exit(0);
    }
   }
  }
}
```

Record non vuoto

Record vuoto: ricerca

Cerca il nome

Crea l'output

Il record trovato (`null` se non c'è)

Necessario

Chiusura

Eccezione di chiusura socket

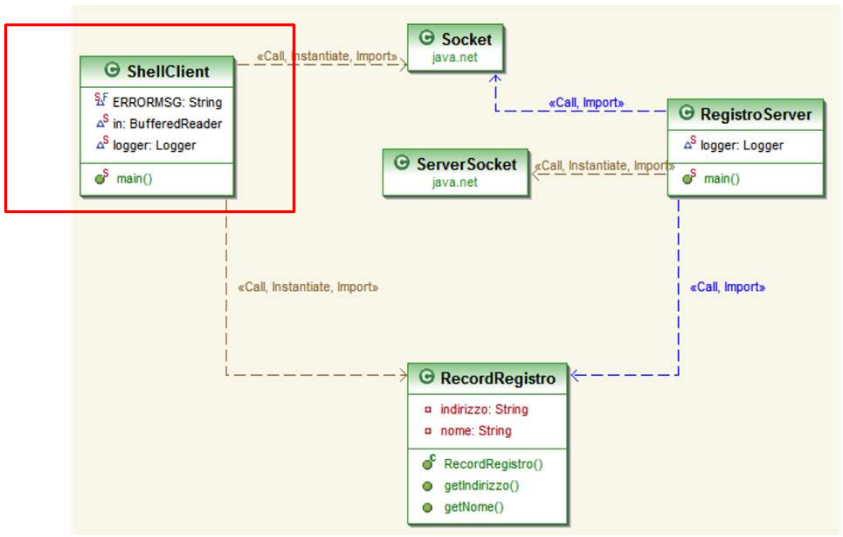
Altre eccezioni

Comunque …

75

La classe ShellClient

76



76

La classe ShellClient

77

```
public class ShellClient {
 public static void main(String args[]) {
  //...
  in = new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("xxxxx")) {
        //...
    }elseif(cmd.equals ("yyyyy")) {
        //...
    }elseif(cmd.equals ("zzzzz")) {
        //...
    }elseSystem.out.println (ERRORMSG);
  }catch(Throwable t) {
    //...
  }

  private static String ask(String prompt)throws
       IOException {
    System.out.print(prompt+"");
    return(in.readLine());
  }
 //...
 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

Input da stdin (`Scanner`)

77

La classe ShellClient

78

```
public class ShellClient {
 public static void main(String args[]) {
  //...
  in = new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("xxxxx")) {
        //...
    }elseif(cmd.equals ("yyyyy")) {
        //...
    }elseif(cmd.equals ("zzzzz")) {
        //...
    }elseSystem.out.println (ERRORMSG);
  }catch(Throwable t) {
    //...
  }

  private static String ask(String prompt)throws
       IOException {
    System.out.print(prompt+"");
    return(in.readLine());
  }
 //...
 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

Input da stdin (`Scanner`)

Loop infinito con uscita

78

La classe ShellClient

79

```
public class ShellClient {
 public static void main(String args[]) {
  //...
  in = new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("xxxxx")) {
        //...
    }elseif(cmd.equals ("yyyyy")) {
        //...
    }elseif(cmd.equals ("zzzzz")) {
        //...
    }elseSystem.out.println (ERRORMSG);
  }catch(Throwable t) {
    //...
  }

  private static String ask(String prompt)throws
       IOException {
    System.out.print(prompt+"");
    return(in.readLine());
  }
 //...
 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

Input da stdin (`Scanner`)

Loop infinito con uscita

Primo comando …

79

La classe ShellClient

80

```
public class ShellClient {
 public static void main(String args[]) {
  //...
  in = new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("xxxxx")) {
        //...
    }elseif(cmd.equals ("yyyyy")) {
        //...
    }elseif(cmd.equals ("zzzzz")) {
        //...
    }elseSystem.out.println (ERRORMSG);
  }catch(Throwable t) {
    //...
  }

  private static String ask(String prompt)throws
       IOException {
    System.out.print(prompt+"");
    return(in.readLine());
  }
 //...
 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

Input da stdin (`Scanner`)

Loop infinito con uscita

Primo comando . . .

Secondo comando …

80

La classe ShellClient

81

```
public class ShellClient {
 public static void main(String args[]) {
  //...
  in = new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("xxxxx")) {
        //...
    }elseif(cmd.equals ("yyyyy")) {
        //...
    }elseif(cmd.equals ("zzzzz")) {
        //...
    }elseSystem.out.println (ERRORMSG);
  }catch(Throwable t) {
    //...
  }

  private static String ask(String prompt)throws
        IOException {
    System.out.print(prompt+"");
    return(in.readLine());
  }
 //...
 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

- Input da stdin (`Scanner`)
- Loop infinito con uscita
- Primo comando . . .
- Secondo comando . . .
- Terzo comando . . .

81

La classe ShellClient

82

```
public class ShellClient {
 public static void main(String args[]) {
  //...
  in = new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("xxxxx")) {
        //...
    }elseif(cmd.equals ("yyyyy")) {
        //...
    }elseif(cmd.equals ("zzzzz")) {
        //...
    }elseSystem.out.println (ERRORMSG);
  }catch(Throwable t) {
    //...
  }

  private static String ask(String prompt)throws
        IOException {
    System.out.print(prompt+"");
    return(in.readLine());
  }
 //...
 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

- Input da stdin (`Scanner`)
- Loop infinito con uscita
- Primo comando . . .
- Secondo comando . . .
- Terzo comando . . .
- Messaggio di errore

82

La classe ShellClient

83

```
public class ShellClient {
 public static void main(String args[]) {
  //...
  in = new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("xxxxx")) {
        //...
    }elseif(cmd.equals ("yyyyy")) {
        //...
    }elseif(cmd.equals ("zzzzz")) {
        //...
    }elseSystem.out.println (ERRORMSG);
  }catch(Throwable t) {
    //...
  }

   private static String ask(String prompt)throws
        IOException {
     System.out.print(prompt+"");
     return(in.readLine());
   }
  //...
  static final String ERRORMSG ="Cosa?";
  static BufferedReader in =null;
}
```

- Input da stdin (`Scanner`)
- Loop infinito con uscita
- Primo comando . . .
- Secondo comando . . .
- Terzo comando . . .
- Messaggio di errore
- Eccezioni

83

La classe ShellClient

84

```
public class ShellClient {
 public static void main(String args[]) {
  //...
  in = new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("xxxxx")) {
        //...
    }elseif(cmd.equals ("yyyyy")) {
        //...
    }elseif(cmd.equals ("zzzzz")) {
        //...
    }elseSystem.out.println (ERRORMSG);
  }catch(Throwable t) {
    //...
  }

   private static String ask(String prompt)throws
        IOException {
     System.out.print(prompt+"");
     return(in.readLine());
   }
  //...
  static final String ERRORMSG ="Cosa?";
  static BufferedReader in =null;
}
```

- Input da stdin (`Scanner`)
- Loop infinito con uscita
- Primo comando . . .
- Secondo comando . . .
- Terzo comando . . .
- Messaggio di errore
- Eccezioni
- Metodo di input

84

La classe ShellClient

85

```
import java.io.*;
import java.net.*;
import java.util.logging.Logger;
public class ShellClient {
 static Logger logger = Logger.getLogger("global");
 public static void main(String args[]) {
  String host = args[0];
  String cmd;
  in =new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("inserisci")) {
    System.out.println ("Inserire i dati.");
    String nome = ask("Nome:");
     String indirizzo = ask ("Indirizzo:");
     RecordRegistro r = new RecordRegistro(
                        nome, indirizzo);
     Socket socket = new Socket (host, 7000);
     ObjectOutputStream sock_out = new
       ObjectOutputStream
             (socket.getOutputStream());
     sock_out.writeObject(r);
     sock_out.flush();
     socket.close();
    }elseif(cmd.equals ("cerca")) {
    //...
```

Import

85

La classe ShellClient

86

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class ShellClient {
 static Logger logger = Logger.getLogger("global");
 public static void main(String args[]) {
  String host = args[0];
  String cmd;
  in =new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("inserisci")) {
    System.out.println ("Inserire i dati.");
    String nome = ask("Nome:");
     String indirizzo = ask ("Indirizzo:");
     RecordRegistro r = new RecordRegistro(
                        nome, indirizzo);
     Socket socket = new Socket (host, 7000);
     ObjectOutputStream sock_out = new
       ObjectOutputStream
             (socket.getOutputStream());
     sock_out.writeObject(r);
     sock_out.flush();
     socket.close();
    }elseif(cmd.equals ("cerca")) {
    //...
```

Import

Hostname su linea di comando

86

La classe ShellClient

87

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class ShellClient {
 static Logger logger = Logger.getLogger("global");
 public static void main(String args[]) {
  String host = args[0];
  String cmd;
  in =new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("inserisci")) {
    System.out.println ("Inserire i dati.");
    String nome = ask("Nome:");
     String indirizzo = ask ("Indirizzo:");
     RecordRegistro r = new RecordRegistro(
                        nome, indirizzo);
     Socket socket = new Socket (host, 7000);
     ObjectOutputStream sock_out = new
       ObjectOutputStream
             (socket.getOutputStream());
     sock_out.writeObject(r);
     sock_out.flush();
     socket.close();
    }elseif(cmd.equals ("cerca")) {
    //...
```

Import

Hostname su linea di comando

Lettura da stdin

87

La classe ShellClient

88

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class ShellClient {
 static Logger logger = Logger.getLogger("global");
 public static void main(String args[]) {
  String host = args[0];
  String cmd;
  in =new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("inserisci")) {
    System.out.println ("Inserire i dati.");
    String nome = ask("Nome:");
     String indirizzo = ask ("Indirizzo:");
     RecordRegistro r =new RecordRegistro(
                        nome, indirizzo);
     Socket socket =new Socket (host, 7000);
     ObjectOutputStream sock_out =new
       ObjectOutputStream
             (socket.getOutputStream());
     sock_out.writeObject(r);
     sock_out.flush();
     socket.close();
    }elseif(cmd.equals ("cerca")) {
    //...
```

Import

Hostname su linea di comando

Lettura da stdin

Finquando non si digita "`quit`"

88

La classe ShellClient

89

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class ShellClient {
 static Logger logger = Logger.getLogger("global");
 public static void main(String args[]) {
  String host = args[0];
  String cmd;
  in =new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("inserisci")) {
    System.out.println ("Inserire i dati.");
    String nome = ask("Nome:");
     String indirizzo = ask ("Indirizzo:");
     RecordRegistro r =new RecordRegistro(
                        nome, indirizzo);
     Socket socket =new Socket (host, 7000);
     ObjectOutputStream sock_out =new
       ObjectOutputStream
             (socket.getOutputStream());
     sock_out.writeObject(r);
     sock_out.flush();
     socket.close();
    }elseif(cmd.equals ("cerca")) {
    //...
```

- Import
- Hostname su linea di comando
- Lettura da stdin
- Finquando non si digita "`quit`"
- Se si chiede di inserire

89

La classe ShellClient

90

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class ShellClient {
 static Logger logger = Logger.getLogger("global");
 public static void main(String args[]) {
  String host = args[0];
  String cmd;
  in =new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("inserisci")) {
    System.out.println ("Inserire i dati.");
    String nome = ask("Nome:");
     String indirizzo = ask ("Indirizzo:");
     RecordRegistro r =new RecordRegistro(
                        nome, indirizzo);
     Socket socket = new Socket (host,
     7000); ObjectOutputStream sock_out =new

       ObjectOutputStream
             (socket.getOutputStream());
     sock_out.writeObject(r);
     sock_out.flush();
     socket.close();
    }elseif(cmd.equals ("cerca")) {
    //...
```

- Import
- Hostname su linea di comando
- Lettura da stdin
- Finquando non si digita "`quit`"
- Se si chiede di inserire
- …chiede i dati

90

La classe ShellClient

91

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class ShellClient {
 static Logger logger = Logger.getLogger("global");
 public static void main(String args[]) {
  String host = args[0];
  String cmd;
  in =new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("inserisci")) {
    System.out.println ("Inserire i dati.");
    String nome = ask("Nome:");
     String indirizzo = ask ("Indirizzo:");
     RecordRegistro r =new RecordRegistro(
                        nome, indirizzo);
     Socket socket =new Socket (host, 7000);
     ObjectOutputStream sock_out =new

       ObjectOutputStream
             (socket.getOutputStream());
     sock_out.writeObject(r);
     sock_out.flush();
     socket.close();
    }elseif(cmd.equals ("cerca")) {
    //...
```

- Import
- Hostname su linea di comando
- Lettura da stdin
- Finquando non si digita "`quit`"
- Se si chiede di inserire
- …chiede i dati
- Crea record

91

La classe ShellClient

92

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class ShellClient {
 static Logger logger = Logger.getLogger("global");
 public static void main(String args[]) {
  String host = args[0];
  String cmd;
  in =new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("inserisci")) {
    System.out.println ("Inserire i dati.");
    String nome = ask("Nome:");
     String indirizzo = ask ("Indirizzo:");
     RecordRegistro r =new RecordRegistro(
                        nome, indirizzo);
     Socket socket =new Socket (host, 7000);
     ObjectOutputStream sock_out =new
       ObjectOutputStream
             (socket.getOutputStream());
     sock_out.writeObject(r);
     sock_out.flush();
     socket.close();
    }elseif(cmd.equals ("cerca")) {

    //...
```

- Import
- Hostname su linea di comando
- Lettura da stdin
- Finquando non si digita "`quit`"
- Se si chiede di inserire
- …chiede i dati
- Crea record
- Crea stream

92

La classe ShellClient

93

```
import java.io. *;
import java.net. *;
import java.util.logging.Logger;
public class ShellClient {
 static Logger logger = Logger.getLogger("global");
 public static void main(String args[]) {
  String host = args[0];
  String cmd;
  in =new BufferedReader(new
          InputStreamReader(System.in));
  try{
   while(!(cmd = ask (">>>")).equals("quit")) {
    if(cmd.equals ("inserisci")) {
    System.out.println ("Inserire i dati.");
    String nome = ask("Nome:");
     String indirizzo = ask ("Indirizzo:");
     RecordRegistro r =new RecordRegistro(
                        nome, indirizzo);
     Socket socket =new Socket (host, 7000);
     ObjectOutputStream sock_out =new
       ObjectOutputStream
             (socket.getOutputStream());
     sock_out.writeObject(r);

     sock_out.flush();
     socket.close();
    }elseif(cmd.equals ("cerca")) {
    //...
```

- Import
- Hostname su linea di comando
- Lettura da stdin
- Finquando non si digita "`quit`"
- Se si chiede di inserire
- …chiede i dati
- Crea record
- Crea stream
- Scrive il record

93

La classe ShellClient

94

- Se si chiede di ricercare …

```
     //...
    }elseif(cmd.equals ("cerca")) {
     System.out.println ("Inserire il nome per
           la ricerca.");
     String nome = ask("Nome:");
     RecordRegistro r = new RecordRegistro(nome,
           null);
     Socket socket = new Socket (host, 7000);
     ObjectOutputStream sock_out = new
       ObjectOutputStream(socket.getOutputStream());
     sock_out.writeObject(r);
     sock_out.flush();
     ObjectInputStream sock_in = new
       ObjectInputStream(socket.getInputStream());
     RecordRegistro result = (RecordRegistro)
                          sock_in.readObject();
     if(result !=null)
         System.out.println ("Indirizzo:"+
                             result.getIndirizzo());
     else
         System.out.println ("Record assente");
     socket.close();
    }else System.out.println (ERRORMSG);
   }//end while

  //...
```

94

La classe ShellClient

95

```
    //...
  }elseif(cmd.equals ("cerca")) {
   System.out.println ("Inserire il nome per la
         ricerca.");
   String nome = ask("Nome:");
   RecordRegistro r = new RecordRegistro(nome,
         null);
   Socket socket = new Socket (host, 7000);
   ObjectOutputStream sock_out = new
     ObjectOutputStream(socket.getOutputStream());
   sock_out.writeObject(r);
   sock_out.flush();
   ObjectInputStream sock_in = new
     ObjectInputStream(socket.getInputStream());
   RecordRegistro result = (RecordRegistro)
                        sock_in.readObject();
   if(result !=null)
       System.out.println ("Indirizzo:"+
                           result.getIndirizzo());
   else
       System.out.println ("Record assente");
   socket.close();
  }else System.out.println (ERRORMSG);
}//end while
//...
```

Se si chiede di ricercare . . .

Si crea un record con campo indirizzo vuoto

95

La classe ShellClient

96

```
    //...
  }elseif(cmd.equals ("cerca")) {
   System.out.println ("Inserire il nome per
         la ricerca.");
   String nome = ask("Nome:");
   RecordRegistro r = new RecordRegistro(nome,
         null);
   Socket socket =new Socket (host, 7000);
   ObjectOutputStream sock_out =new
     ObjectOutputStream(socket.getOutputStream());
   sock_out.writeObject(r);
   sock_out.flush();
   ObjectInputStream sock_in = new
     ObjectInputStream(socket.getInputStream());
   RecordRegistro result = (RecordRegistro)
                        sock_in.readObject();
   if(result !=null)
       System.out.println ("Indirizzo:"+
                           result.getIndirizzo());
   else
       System.out.println ("Record assente");
   socket.close();
  }else System.out.println (ERRORMSG);
}//end while
//...
```

Se si chiede di ricercare . . .

Si crea un record con campo indirizzo vuoto

Si apre un socket . . .

96

La classe ShellClient

97

```
    //...
  }elseif(cmd.equals ("cerca")) {
   System.out.println ("Inserire il nome per la
         ricerca.");
   String nome = ask("Nome:");
   RecordRegistro r = new RecordRegistro(nome,
         null);
   Socket socket =new Socket (host, 7000);
   ObjectOutputStream sock_out =new
     ObjectOutputStream(socket.getOutputStream());
   sock_out.writeObject(r);
   sock_out.flush();
   ObjectInputStream sock_in = new
     ObjectInputStream(socket.getInputStream());
   RecordRegistro result = (RecordRegistro)
                        sock_in.readObject();
   if(result !=null)
       System.out.println ("Indirizzo:"+
                           result.getIndirizzo());
   else
       System.out.println ("Record assente");
   socket.close();
  }else System.out.println (ERRORMSG);
 }//end while
//...
```

Se si chiede di ricercare . . .

Si crea un record con campo indirizzo vuoto

Si apre un socket . . .

. . . e si preleva lo stream

97

La classe ShellClient

98

```
    //...
  }elseif(cmd.equals ("cerca")) {
   System.out.println ("Inserire il nome per
         la ricerca.");
   String nome = ask("Nome:");
   RecordRegistro r = new RecordRegistro(nome,
         null);
   Socket socket =new Socket (host, 7000);
   ObjectOutputStream sock_out =new
     ObjectOutputStream(socket.getOutputStream());
   sock_out.writeObject(r);
   sock_out.flush();
   ObjectInputStream sock_in = new
     ObjectInputStream(socket.getInputStream());
   RecordRegistro result = (RecordRegistro)
                        sock_in.readObject();
   if(result !=null)
       System.out.println ("Indirizzo:"+
                           result.getIndirizzo());
   else
       System.out.println ("Record assente");
   socket.close();
  }else System.out.println (ERRORMSG);
}//end while
//...
```

Se si chiede di ricercare . . .

Si crea un record con campo indirizzo vuoto

Si apre un socket . . .

. . . e si preleva lo stream

Si invia l'oggetto

98

La classe ShellClient

99

```
    //...
  }elseif(cmd.equals ("cerca")) {
   System.out.println ("Inserire il nome per la
         ricerca.");
   String nome = ask("Nome:");
   RecordRegistro r =new RecordRegistro(nome,
         null);
   Socket socket = new Socket (host, 7000);
   ObjectOutputStream sock_out =new
     ObjectOutputStream(socket.getOutputStream());
   sock_out.writeObject(r);
   sock_out.flush();
   ObjectInputStream sock_in = new
     ObjectInputStream(socket.getInputStream());
   RecordRegistro result = (RecordRegistro)
                        sock_in.readObject();
   if(result !=null)
       System.out.println ("Indirizzo:"+
                           result.getIndirizzo());
   else
       System.out.println ("Record assente");
   socket.close();
  }else System.out.println (ERRORMSG);
}//end while
//...
```

- Se si chiede di ricercare . . .
- Si crea un record con campo indirizzo vuoto
- Si apre un socket . . .
- . . . e si preleva lo stream
- Si invia l'oggetto
- . . . facendo il flush

99

La classe ShellClient

100

```
    //...
  }elseif(cmd.equals ("cerca")) {
   System.out.println ("Inserire il nome per la
         ricerca.");
   String nome = ask("Nome:");
   RecordRegistro r = new RecordRegistro(nome,
         null);
   Socket socket =new Socket (host, 7000);
   ObjectOutputStream sock_out =new
     ObjectOutputStream(socket.getOutputStream());
   sock_out.writeObject(r);
   sock_out.flush();
   ObjectInputStream sock_in = new
     ObjectInputStream(socket.getInputStream());
   RecordRegistro result = (RecordRegistro)
                        sock_in.readObject();
   if(result !=null)
       System.out.println ("Indirizzo:"+
                           result.getIndirizzo());
   else
       System.out.println ("Record assente");
   socket.close();
  }else System.out.println (ERRORMSG);
  }//end while
//...
```

- Se si chiede di ricercare . . .
- Si crea un record con campo indirizzo vuoto
- Si apre un socket . . .
- . . . e si preleva lo stream
- Si invia l'oggetto
- . . . facendo il flush
- Risposta

100

La classe ShellClient

101

```
    //...
  }elseif(cmd.equals ("cerca")) {
   System.out.println ("Inserire il nome per la
         ricerca.");
   String nome = ask("Nome:");
   RecordRegistro r = new RecordRegistro(nome,
         null);
   Socket socket = new Socket (host, 7000);
   ObjectOutputStream sock_out =new
     ObjectOutputStream(socket.getOutputStream());
   sock_out.writeObject(r);
   sock_out.flush();
   ObjectInputStream sock_in = new
     ObjectInputStream(socket.getInputStream());
   RecordRegistro result = (RecordRegistro)
                        sock_in.readObject();
   if(result != null)
       System.out.println ("Indirizzo:"+
                           result.getIndirizzo());
   else
       System.out.println ("Record assente");
   socket.close();
  }else System.out.println (ERRORMSG);
}//end while
//...
```

Se si chiede di ricercare . . .

Si crea un record con campo indirizzo vuoto

Si apre un socket . . .

. . . e si preleva lo stream

Si invia l'oggetto

. . . facendo il flush

Risposta

Se il risultato è non nullo si stampa

101

La classe ShellClient

102

```
    //...
  }elseif(cmd.equals ("cerca")) {
   System.out.println ("Inserire il nome per la
         ricerca.");
   String nome = ask("Nome:");
   RecordRegistro r =new RecordRegistro(nome,
         null);
   Socket socket = new Socket (host, 7000);
   ObjectOutputStream sock_out = new
     ObjectOutputStream(socket.getOutputStream());
   sock_out.writeObject(r);
   sock_out.flush();
   ObjectInputStream sock_in = new
     ObjectInputStream(socket.getInputStream());
   RecordRegistro result = (RecordRegistro)
                        sock_in.readObject();
   if(result !=n ull)
       System.out.println ("Indirizzo:"+
                           result.getIndirizzo());
   else
       System.out.println ("Record assente");
   socket.close();
  }else System.out.println (ERRORMSG);
 }//end while
//...
```

Se si chiede di ricercare . . .

Si crea un record con campo indirizzo vuoto

Si apre un socket . . .

. . . e si preleva lo stream

Si invia l'oggetto

. . . facendo il flush

Risposta

Se il risultato è non nullo si stampa

. . . o non esiste

102

La classe ShellClient

103

```
    //...
   }elseif(cmd.equals ("cerca")) {
    System.out.println ("Inserire il nome per
          la ricerca.");
    String nome = ask("Nome:");
    RecordRegistro r =newRecordRegistro(nome,
          null);
    Socket socket = new Socket (host, 7000);
    ObjectOutputStream sock_out =new
      ObjectOutputStream(socket.getOutputStream());
    sock_out.writeObject(r);
    sock_out.flush();
    ObjectInputStream sock_in =new
      ObjectInputStream(socket.getInputStream());
    RecordRegistro result = (RecordRegistro)
                        sock_in.readObject();
    if(result != null)
        System.out.println ("Indirizzo:"+
                            result.getIndirizzo());
    else
        System.out.println ("Recordassente");
    socket.close();
   }else System.out.println (ERRORMSG);
  }//end while
 //...
```

Se si chiede di ricercare . . .

Si crea un record con campo indirizzo vuoto

Si apre un socket . . .

. . . e si preleva lo stream

Si invia l'oggetto

. . . facendo il flush

Risposta

Se il risultato è non nullo si stampa

. . . o non esiste

Errore comando

103

La classe ShellClient

104

```
    //...

     }catch(Throwable t) {
    logger.severe("Lanciata Throwable:"+
          t.getMessage());
    t.printStackTrace();
   }
   System.out.println ("Bye bye");
  }//fine main

   private static String ask(String prompt)throws
        IOException {
   System.out.print(prompt+"");
   return(in.readLine());
  }


  static final String ERRORMSG ="Cosa?";
  static BufferedReader in =null;
}
```

Tutte le eccezioni

104

La classe ShellClient

105

```
    //...

     }catch(Throwable t) {
    logger.severe("Lanciata Throwable:"+
            t.getMessage());
    t.printStackTrace();
  }
  System.out.println ("Bye bye");
 }//fin main

  private static String ask(String prompt)throws
        IOException {
  System.out.print(prompt+"");
  return(in.readLine());
}


 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

Tutte le eccezioni

vengono loggate

105

La classe ShellClient

106

```
    //...

     }catch(Throwable t) {
    logger.severe("Lanciata Throwable:"+
            t.getMessage());
    t.printStackTrace();
  }
  System.out.println ("Bye bye");
 }//fine main

  private static String ask(String prompt)throws
        IOException {
  System.out.print(prompt+"");
  return(in.readLine());
}


 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

Tutte le eccezioni

vengono loggate

Metodo di servizio per input da tastiera

106

La classe ShellClient

107

```
    //...

     }catch(Throwable t) {
    logger.severe("Lanciata Throwable:"+
            t.getMessage());
    t.printStackTrace();
  }
  System.out.println ("Bye bye");
 }//fine main

  private static String ask(String prompt)throws
        IOException {
  System.out.print(prompt+"");
  return(in.readLine());
}


 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

Tutte le eccezioni

vengono loggate

Metodo di servizio per input da tastiera

Legge da stdin (e può lanciare eccezione)

107

La classe ShellClient

108

```
    //...

     }catch(Throwable t) {
    logger.severe("Lanciata Throwable:"+
            t.getMessage());
    t.printStackTrace();
  }
  System.out.println ("Bye bye");
 }//fine main

  private static String ask(String prompt)throws
        IOException {
  System.out.print(prompt+"");
  return(in.readLine());
}


 static final String ERRORMSG ="Cosa?";
 static BufferedReader in =null;
}
```

Tutte le eccezioni

vengono loggate

Metodo di servizio per input da tastiera

Legge da stdin (e può lanciare eccezione)

Stringa di errore

108

Cosa si potrebbe fare con questo esempio

109

- Separare Client e Server in due progetti diversi
- Provare su rete locale
- Rendere il server concorrente

109

Conclusioni

110

- Programmazione con i socket
  - Socket TCP
  - Stream
- Alcuni esempi di uso dei socket
  - "Hello World"
  - Un Registro di nomi con architettura client-server
- Conclusioni

Nelle prossime lezioni:

Java Remote Method Invocation

110