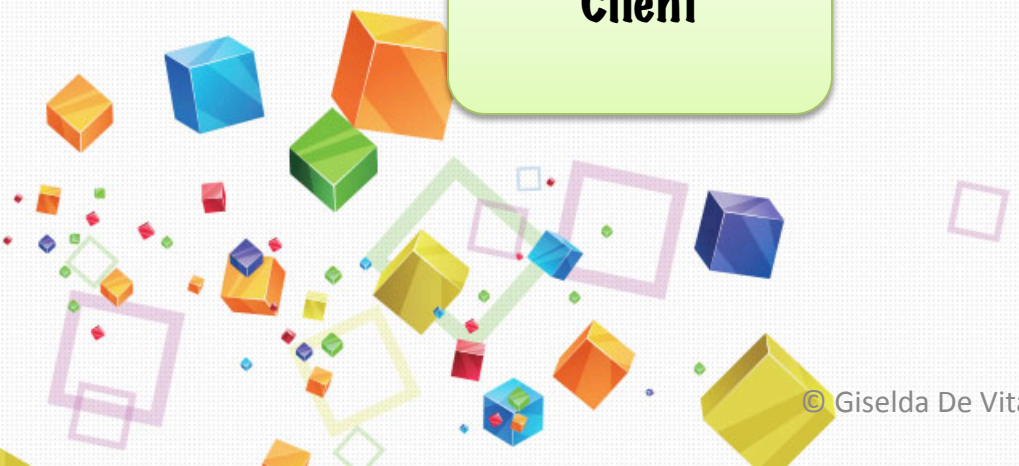
Socket in Java

Un socket

Per la programmazione in rete, un oggetto fondamentale è il **SOCKET**.

Per testare un programma che utilizza i socket in JAVA, abbiamo bisogno di creare due applicazioni distinte:

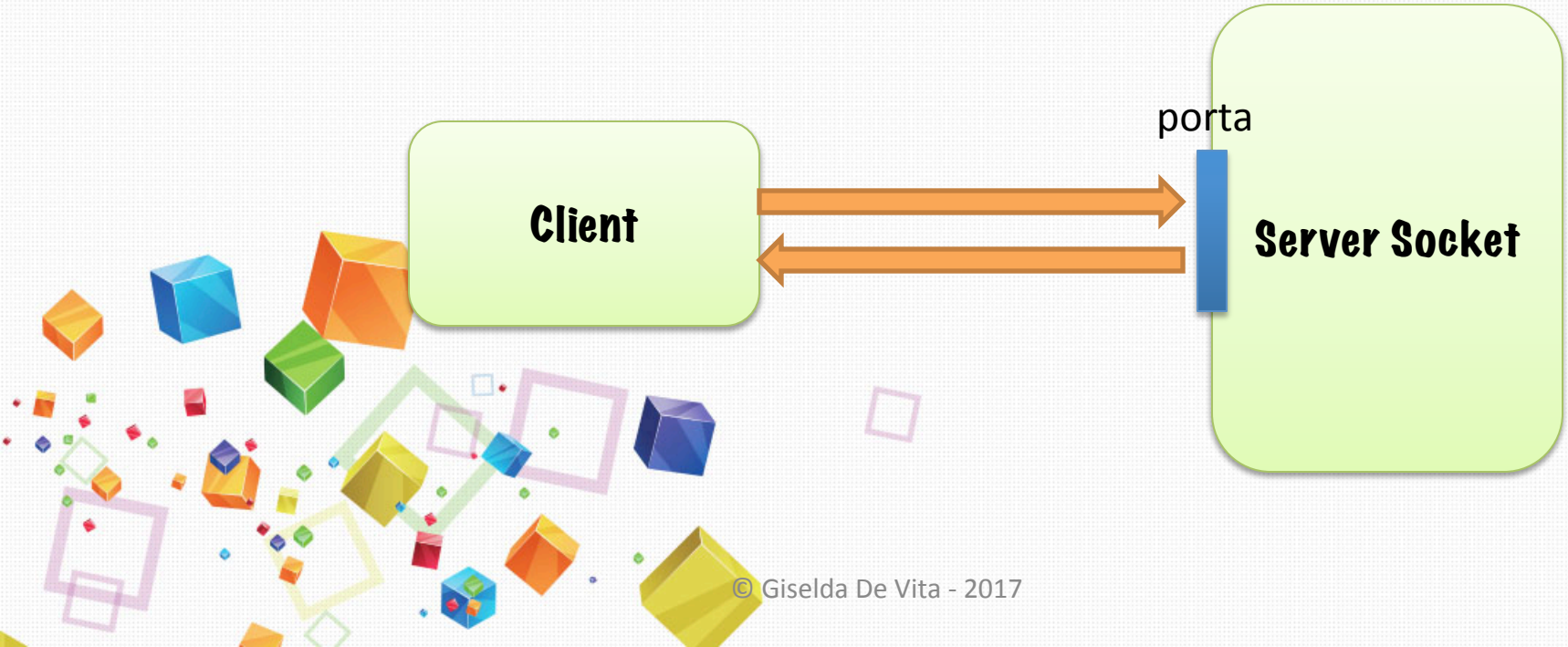


Client

Server Socket

Un socket

Un **SOCKET** viene caratterizzato, a parte dall'ip address del server, da un numero di porta che resta aperta e in ascolto delle connessione da parte di eventuali client.



Un socket client

Client

Il client deve conoscere l'ip address e la porta del server. Crea un **SOCKET** indicando proprio ip e porta nel costruttore:

```
Socket clientSocket = new Socket("127.0.0.1", 3000);
```

Dall'oggetto **SOCKET** si istanziano input e output con cui si comunicherà con il server:

```
DataOutputStream messDaSpedire = new
DataOutputStream(clientSocket.getOutputStream());

BufferedReader ricevutoDalServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
```

Un socket client

Client

Il client spedisce sull'outputStream e legge sull'Input Stream:

```
messDaSpedire.writeBytes(messaggioDaInviare + '\n');

String messaggioDalServer = ricevutoDalServer.readLine();
```

```
import java.io.*;
import java.net.*;

class TestClient {
    String host = "localhost"; //nome o ip address
    int porta = 1373; //porta

 public TestClient() throws Exception{
    System.out.println("Parte il CLIENT...");
    String messaggioDaInviare = "chiamo il SERVER " +host +
     "sulla porta " + porta + " creando un socket!";

    Socket clientSocket = new Socket(host, porta);
    DataOutputStream messDaSpedire = new DataOutputStream(clientSocket.getOutputStream());
    BufferedReader ricevutoDalServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

    messDaSpedire.writeBytes(messaggioDaInviare + '\n');
    String messaggioDalServer = ricevutoDalServer.readLine();

    System.out.println("Il SERVER risponde: " + messaggioDalServer);

    clientSocket.close();
 }

 public static void main() throws Exception {
     new TestClient();
    }

}
```

Un server socket

Il server, invece, istanzia su una determinata porta, indicata nel costruttore, un oggetto **ServerSocket**.

```
ServerSocket serverSocket = new ServerSocket(3000);
```

Un server socket

Il un ciclo while(true) istanzia un socket che deriva dall'accept sul serversocket, ovvero il socket si istanzia quando arriva la richiesta di un client.

```
Socket connectionSocket = serverSocket.accept();
```

Un server socket

Il un ciclo while(true) istanzia un socket che deriva dall'accept sul serversocket, ovvero il socket si istanzia quando arriva la richiesta di un client.

```
Socket connectionSocket = serverSocket.accept();
```

Come per il client, sul server si istanziano input ed output per comunicare.

```java
import java.io.*;
import java.net.*;

public class TestServer
{
    /**
     * Costruttore degli oggetti di classe  TCPServer
     */
    public TestServer()  throws Exception
    {
        String messaggioClient;
        String risposta;
        ServerSocket serverSocket = new ServerSocket(1373);
        System.out.println("Parte il SERVER...");
        while (true) {

          Socket connectionSocket = serverSocket.accept();
          BufferedReader ricevutoDalClient =
            new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
          DataOutputStream daSpedireAlClient = new DataOutputStream(connectionSocket.getOutputStream());
          SocketAddress ipClient = connectionSocket.getRemoteSocketAddress();
          messaggioClient = ricevutoDalClient.readLine();
          System.out.println("Ricevuto da: " + ipClient + "   " + messaggioClient);
          risposta = "Ho ricevuto " + messaggioClient + '\n';
          daSpedireAlClient.writeBytes(risposta);
         }
    }

public static void main() throws Exception {
      new TestServer();
  }
}
```

Esercitazione

Modificare i messaggi inviati dal client e la risposta del server

Esercitazione

Creare una interfaccia grafica per il client in modo da poter indicare ip address e porta e messaggio da inviare.