Codice del client

Dopo aver scritto e descritto il codice del server, passiamo al codice del client.

```
namespace SocketClient
{
    class Program
    {
        static void Main(string[] args)
        {
            EseguiClient();
            Console.WriteLine("Premi un tasto per finire");
            Console.ReadKey();
        }

        static void EseguiClient()
        {
            try
            {
                IPAddress ipAddr = IPAddress.Parse("127.0.0.1");
                IPEndPoint remoteEndPoint = new IPEndPoint(ipAddr, 3306);

                Socket sender = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

                try
                {
                    sender.Connect(remoteEndPoint);
                    while (true)
                    {
                        Console.WriteLine("Inserisci un messaggio");
                        string messaggio = Console.ReadLine();
                        byte[] messageSent = Encoding.ASCII.GetBytes(messaggio);
                        if (messaggio.IndexOf("@") > -1)
                            break;

                        sender.Send(messageSent);
                        byte[] messageReceived = new byte[1024];

                        int byteRecv = sender.Receive(messageReceived);
                        Console.WriteLine("Messaggio dal Server -> {0} da {1}",
                                                    Encoding.ASCII.GetString(messageReceived, 0, byteRecv),
                                                    sender.RemoteEndPoint.ToString());
                    }
                }
                catch (ArgumentNullException ane)
                {
                    Console.WriteLine("ArgumentNullException : {0}", ane.ToString());
                }

                catch (SocketException se)
                {
                    Console.WriteLine("SocketException : {0}", se.ToString());
                }

                catch (Exception e)
                {
                    Console.WriteLine("Unexpected exception : {0}", e.ToString());
                }
                sender.Shutdown(SocketShutdown.Both);
                sender.Close();
            }

            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }

        }
    }
}
```

Spiegazione:

Nell'entry point (**Main**) compare la chiamata al metodo che gestisce tutta la logica, `EseguiClient();`, poi a seguire due istruzioni che invitano l'utente a chiudere il programma.
Nel metodo queste sono le istruzioni:

`try` `{`	Tutte le istruzioni del metodo, sono state inserite in un blocco **try…catch** per la cattura di eventuali eccezioni
`IPAddress ipAddr = IPAddress.Parse("127.0.0.1");`	Creo l'oggetto **ipAddr** di tipo **IPAddress** che rappresenta l'indirizzo IP remoto del software server precedente convertendo una stringa "**127.0.0.1**" in un indirizzo IP valido (***Parse***)
`IPEndPoint remoteEndPoint = new IPEndPoint(ipAddr, 3306);`	Creo un oggetto di tipo **IPEndPoint** che rappresenta l'IP remoto e la porta di comunicazione. L'IP remoto è lo stesso del client perché stiamo eseguendo entrambi i software in locale, sullo stesso PC.

`Socket sender = new Socket(`
`AddressFamily.InterNetwork,`
`SocketType.Stream,`
`ProtocolType.Tcp`
`);`
	Creo l' oggetto di tipo Socket al cui costruttore passo 3 parametri di tipo enumeratore:
	4. AddressFamily.InterNetwork
	5. SocketType.Stream
	6. ProtocolType.Tcp
	Il primo rappresenta la tipologia di indirizzi che riguardano le reti e nella fattispecie quelli relativi all'IPV4.
	Il secondo indica che vogliamo usufruire di un flusso sulla rete.
	Il terzo indica il tipo di protocollo utilizzato.

`sender.Connect(remoteEndPoint);`	Stabilisce la connessione con il server

`while (true)`
`{`
`Console.WriteLine("Inserisci un messaggio");`
`string messaggio = Console.ReadLine();`
`byte[] messageSent = Encoding.ASCII.GetBytes(messaggio);`
`if (messaggio.IndexOf("@") > -1)`
`break;`
`sender.Send(messageSent);`
`byte[] messageReceived = new byte[1024];`
`int byteRecv = sender.Receive(messageReceived);`
`Console.WriteLine("Messaggio dal Server -> {0} da {1}",`
`Encoding.ASCII.GetString(messageReceived, 0,`
`byteRecv),`
`sender.RemoteEndPoint.ToString());`
`}`
	Ciclo infinito che consente la trasmissione di messaggi fino a quando non compare il carattere '@' di fine trasmissione.

`catch (ArgumentNullException ane)`
`{`
`Console.WriteLine("ArgumentNullException : {0}", ane.ToString());`
`}`
`catch (SocketException se)`
`{`
`Console.WriteLine("SocketException : {0}", se.ToString());`
`}`
`catch (Exception e)`
`{`
`Console.WriteLine("Unexpected exception : {0}", e.ToString());`
`}`
`sender.Shutdown(SocketShutdown.Both);`
`sender.Close();`
`}`
`catch (Exception e)`
`{`
`Console.WriteLine(e.ToString());`
`}`
	Tutte le clausole catch relative ad eventuali eccezioni che dovessero essere catturate

Prova di esecuzione: