1 Telnet Echo-Server

Schreibt eine Klasse `TCPEchoServer`, der Text-Daten, die über eine TCP-Verbindung eintreffen, wieder an den Sender zurückschickt. Der Port, auf dem die Anwendung lauscht, sollte beim Programmaufruf angegeben werden können.
Seht zunächst zu, daß Ihr die erste eintreffende Verbindung behandelt. Wenn Ihr das soweit geschafft habt, denkt mal darüber nach (und setzt es evtl. um), wie man mit mehreren Verbindungen gleichzeitig umgehen könnte.

Ihr könnt euren Server mit dem Kommando `telnet` testen (`telnet <host> <port>`

2 Client/Server-Anwendung in UDP

Schreibt zwei Klassen `UDPServer` und `UDPClient`. Der Client soll Byte-Arrays zufälliger Größe (z.B. 1-40 Byte) an den Server senden. Der Server soll die Daten entgegennehmen, den Inhalt bearbeiten (z.B. jedes Byte durch 2 teilen) und wieder an den Client zurücksenden.
Der Client soll natürlich sowohl die gesendeten, als auch die empfangenen Zahlen ausgeben, um das Ergebnis sichtbar zu machen.

Beim Server muß eingestellt werden könne, auf welcher Portnummer er lauschen soll, während man beim Client den Namen des Servers und dessen Portnummer einstellen können muß - die lokale Portnummer des Clients muß nicht festgelegt werden.
Zur Verfeinerung könnte man darauf achten, daß der Client auch nur solche empfangenen Daten akzeptiert, die vom Server kommen (sonst könnte man ihn durcheinander bringen).

3 Erweiterter Telnet-Server

Basierend auf der Aufgabe 1, bringt eurem Server bei, die empfangenen Sätze als Kommandos aufzufassen. So könnte z.B. das Wort `exit` dafür sorgen, daß der Server die Verbindung schließt. Das Kommando `get date` liefert die aktuelle Datum und `get time` die aktuelel Zeit.