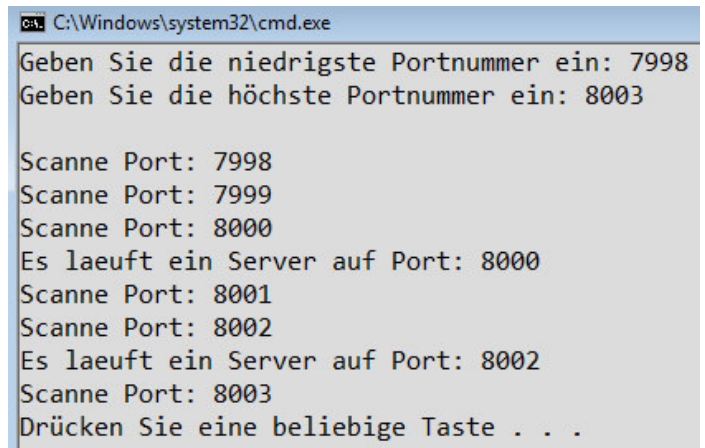


1. Aufgabe

Ein Programm soll die Ports des eigenen Rechners (URL: `localhost`) scannen. Alle Portnummern, welche bereits durch einen Server belegt sind, müssen auf der Konsole ausgegeben werden. Nach dem Start des Programms legt der Anwender den Bereich der Portnummern fest, welcher gescannt werden soll.

Der abgebildete Screenshot bezieht sich auf diese Situation:

- Die Ports im Bereich 7998 bis 8003 sollen gescannt werden.
- Auf den Ports 8000 und 8002 laufen Server, auf den anderen nicht.



```
C:\Windows\system32\cmd.exe
Geben Sie die niedrigste Portnummer ein: 7998
Geben Sie die höchste Portnummer ein: 8003

Scanne Port: 7998
Scanne Port: 7999
Scanne Port: 8000
Es laeuft ein Server auf Port: 8000
Scanne Port: 8001
Scanne Port: 8002
Es laeuft ein Server auf Port: 8002
Scanne Port: 8003
Drücken Sie eine beliebige Taste . . .
```

Das Programm soll aus einer Klasse `PortScanner` und einer Startklasse bestehen, welche die `main`-Methode enthält. Die Klasse `PortScanner` soll eine Methode `scannen` enthalten, welcher beim Aufruf eine Portnummer übergeben wird. Sollte auf dieser Nummer ein Server laufen, so gibt sie `true` und ansonsten `false` zurück. In der `main`-Methode wird dann die Methode `scannen` entsprechend oft aufgerufen, um Portnummern zu prüfen. Alle Ein- und Ausgaben müssen innerhalb der `main`-Methode realisiert werden.

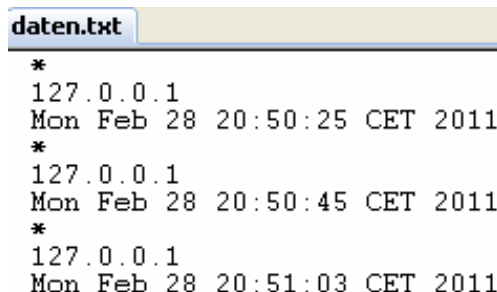
Erstellen und testen Sie das Programm.

Hinweis:

Zur Realisierung der Methode `scannen` kann dieses Prinzip genutzt werden: Läuft z.B. ein Server auf dem Port 8000, so kann ein Socket für diesen Port erzeugt werden. Lässt sich kein Socket für diesen Port erzeugen, so bedeutet dies, dass auf diesem Port kein Server läuft.

2. Aufgabe = Zusatzaufgabe

Ein Server soll die Verbindungsdaten von Clients speichern. Wenn sich ein Client mit dem Server verbindet, so speichert der Server sofort die IP-Adresse in der Dotted-Quad-Notation (Dotted-Decimal-Notation) sowie Datum und Uhrzeit des Verbindungsaufbaus in einer Textdatei mit diesem Format:



```
daten.txt
*
127.0.0.1
Mon Feb 28 20:50:25 CET 2011
*
127.0.0.1
Mon Feb 28 20:50:45 CET 2011
*
127.0.0.1
Mon Feb 28 20:51:03 CET 2011
```

Anschließend findet der Datenaustausch zwischen Client und Server statt, welcher hier allerdings keine Rolle spielen soll. Es würde auch genügen, den Server mit `Thread.sleep(1000)` für eine Sekunde anzuhalten und danach die Verbindung zum Client sauber zu trennen. Nach der Trennung wartet der Server auf den nächsten Client. Als Client kann der Telnet-Client benutzt werden.