8. Gyakorlat

legendi@inf.elte.hu

2010. március 30.

Kiegészítés

ZH Feladatok:

- Interfész használatára példa:

```
public interface GroupedInterface
    extends Interface1, Interface2, Interface3 { ... }
```

Forrás: `http://java.sun.com/docs/books/tutorial/java/IandI/interfaceDef.html` vagy pl.:

```
class A extends B implements I { ... }
```

- Absztrakt osztályra példa

```
abstract class Point {
    int x = 1, y = 1;
    void move(int dx, int dy) {
        x += dx;
        y += dy;
        alert();
    }
    abstract void alert();
}
```

Forrás: `http://java.sun.com/docs/books/jls/second_edition/html/classes.doc.html#34944`

Networking

Elméletben OSI modell ("All People Seem To Need Data Processing" v. "Please Do Not Throw Salami Pizza Away"), gyakorlatban TCP/IP:

- Alkalmazás (SMTP, Telnet, FTP, etc.),
- Transzport (TCP vagy UDP),

- Hálózati (IPv4, IPv6),
- Adatkapcsolati, Fizikai rétegek (kommunikáció)

Szolgáltatási pont definiálásához három dolog kell:

- Internet cím (IPv4 vagy DNS által feloldva)
- Protokoll azonosító (TCP v. UDP)
- Port (lehet TCP, UDP is, a címtartományok függetlenek (!), 16 bites egész érték 0-65535)

Java esetén a `java.net.*` csomag használható.

Összeköttetés-alapú kapcsolat

Kliens-szerver modell, a szerver általában:

1. Lefoglal egy TCP portot
2. Várakozik egy kliens kapcsolódására
3. Ha kliens jelzi a kapcsolódási szándékát, felveszi vele a kapcsolatot, és kiszolgálja
4. Folytatja a 2. ponttól

A kliens működése általában:

1. Lefoglal egy TCP portot, ezen keresztül kommunikál a szerverrel
2. Kapcsolódik a másik végponton a szerverhez, azon a porton, amelyet az kozzétett
3. Lezajlik a kommunikáció
4. A kliens bontja a kapcsolatot a szerverrel

Megjegyzések

- a kliens lokális portját nem kell ismernünk
- a kapcsolat kiépítése után az full-duplex (kétirányú)
- a szerver szinte mindig többszálú (ld. következő óra!)

Összeköttetés-mentes kapcsolat

Majd előadáson ;]

Címek kezelése

Hasznunkra válik az `InetAddress` osztály:

- getByName(String) név -> cím
- getByAddress(byte[]) IPv4/6 -> cím
- getAllByName() név -> címek (attól függ, hogy van beheggesztve a NS)
- getLocalHost() -> saját cím

Részletesen: `http://java.sun.com/javase/6/docs/api/java/net/InetAddress.html`

Példa

Szerveralkalmazás

```
package gyak8;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class SimpleServer {
    public static void main(final String[] args) throws IOException {
        final int port = Integer.parseInt(args[0]);
        final ServerSocket server = new ServerSocket(port);

        Socket client = null;

        while (true) {
            client = server.accept();
            final BufferedReader br = new BufferedReader(
                new InputStreamReader(client.getInputStream()));
            final PrintWriter pw = new PrintWriter(client.getOutputStream());

            final String line = br.readLine();
            System.out.println("Got message: " + line);

            final String ret = new StringBuilder(line).reverse().toString();
```

```
            System.out.println("Sending reply: " + ret);

            pw.println( ret );
            pw.flush();

            client.close();

            if (line.equals("exit")) {
                break;
            }
        }

        server.close();
    }
}
```

Használat

```
> java SimpleServer 5000
```

Kliens alkalmazás

```
package gyak8;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;

public class SimpleClient {
    public static void main(String[] args)
            throws UnknownHostException, IOException {
        String host = args[0];
        int port = Integer.parseInt(args[1]);
        String value = args[2];

        Socket socket = new Socket(host, port);

        BufferedReader br = new BufferedReader(
            new InputStreamReader(socket.getInputStream()));
        PrintWriter pw = new PrintWriter(socket.getOutputStream());
```

```
        System.out.println("Message: " + value);
        pw.println(value);
        pw.flush();

        System.out.println( "Response:" + br.readLine() );
        socket.close();
    }
}
```

Használat:

```
> java SimpleClient localhost 5000 mentegetnem
> java SimpleClient localhost 5000 ideseggesedi
> java SimpleClient localhost 5000 exit
```

Sauce: `http://hu.wikipedia.org/wiki/Magyar_nyelvû_palindromok_listája`

Internet és a WWW-objektumok elérése

Socket helyett [Http]URLConnection használatával.

Hasznos segédeszközök

ping, netstat, netcat, telnet, tcpdump, etc.

Feladat

Készítsünk egy egyszerű alkalmazást, amely egy megadott HTML oldalt megjelenít a konzolon. Ennek szabványos kommunikációja így zajlik:

```
> telnet index.hu 80
Trying 217.20.130.97...
Connected to index.hu.
Escape character is '^]'.
GET / HTTP/1.1
Host: index.hu


```

Megjegyzés A végén két újsor karakter van!

Feladat

Készítsünk egy közös chat alkalmazást! Ti írjátok a kliens programot, amely csatlakozik a pandora.inf.elte.hu szerver 5000-es portjára.

A program parancssori argumentuma a felhasználó választott nickneve (ez csak alfanumerikus karaktereket tartalmazhat).

Indulás után építsen ki egy standard TCP kapcsolatot, majd küldje el a szervernek a "NICK nick" üzenetet.

Ezután a felhasználó megkapja a jelenleg chatelő nickneveket (; karakterrel elválasztva, egyetlen sorban).

Ha új felhasználó lép a chatszobába, minden más felhasználó kap egy "JOIN nick" üzenetet.

Ha valamely felhasználó bontja a kapcsolatot, minden más felhasználó kap egy "QUIT nick" üzenetet.

Ha a szervert valamely okból leállítják, minden felhasználó kap egy "DISCONNECT uzenet" üzenetet.

Ha valaki üzenetet küld, akkor minden más felhasználó kap egy "MSG uzenet" üzenetet.

A szerver minden üzenetre visszajelez: "OK uzenet", vagy "ERR hibauzenet" formában (pl. az adott nicknév ár foglalt, etc.).

A fenti programhoz készíts egy tetszőleges vezérlő felületet (lehet konzolos, lehet grafikus).

+/- Feladat

A feladatokat a `legendi@inf.elte.hu` címre küldjétek, **következő szombat éjfélig** (2010. április 13., a szünetre való tekintettel)! A subject a következőképp nézzen ki:

```
csop<csoportszám>_gyak<gyakorlat száma>_<EHA-kód>_<a megoldott feladatok száma>
```

Például:

```
csop1_gyak7_LERIAAT_1
csop1_gyak7_LERIAAT_12
```

Mellékelni csak a Java forrásfájlokat mellékeljétek (semmiképp ne teljes Eclipse/NetBeans projecteket), esetleg rarolva, zipelve, és egy levelet küldjetek (több levél esetén az utolsó mellékleteit értékelem)! További fontos kritériumok:

- A konvenciókra figyeljetek plz!
- Ékezetes karaktereket **ne** használjatok! Főleg azonosítók esetében ne! A Java ugyan ezt megengedi, ugyanakkor a különböző környezetekbe való konvertáláskor ($latin2 \leftrightarrow UTF-8 \leftrightarrow Cp1250$) összetörnek a karakterek! Az ilyen forrásokat fordítani, következésképp értékelni sem tudom.

1. Feladat

Készítsetek egy minimális port scanner alkalmazást! A programnak 1 parancssori argumentuma legyen, a tesztelendő host címe. Ezután a program az [1..1024] intervallumban (ezek az ún. *well known* portok) tesztelje végig, hogy hol fogad kapcsolatot az adott gép (próbáljon meg egy TCP socket kapcsolatot létrehozni). Amennyiben ez sikeres, jelezzük a felhasználónak, majd írjuk ki, hogy milyen szolgáltatás fut valószínűleg a gépen. Localhoston teszteljétek. A host azonosításához használjátok az `InetAddress` osztályt!

Egy lista, amely tartalmazza a szolgáltatások neveit:
`http://www.iana.org/assignments/port-numbers`

2. Feladat

Válassz ki egy tetszőleges alkalmazási protokollt (HTTP, SMTP, FTP, IRC, etc.). Készíts egy minimális kliens alkalmazást hozzá, amely egyszerű funkciók ellátására képes. Nézz utána, hogy működik az adott protokoll, majd próbáld meg telnet segítségével használni. Ha sikerül, írj a kisebb funkciókra egy programot. **Tipp:** Google → FTP over telnet