laboratorium: zadanie 6 termin: 16 maja 2019 r.

ZAAWANSOWANE TECHNOLOGIE JAVY

APLIKACJE KLIENT–SERWER NA TCP

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

TCP jest protokołem działającym w trybie klient—serwer. Serwer oczekuje na nawiązanie połączenia na określonym porcie. Klient inicjalizuje połączenie do serwera.

W przeciwieństwie do UDP, TCP gwarantuje wyższym warstwom komunikacyjnym dostarczenie wszystkich pakietów w całości, z zachowaniem kolejności i bez duplikatów. Zapewnia to wiarygodne połączenie kosztem większego narzutu w postaci nagłówka i większej liczby przesyłanych pakietów. Chociaż protokół definiuje pakiet TCP, z punktu widzenia wyższej warstwy oprogramowania dane płynące połączeniem TCP należy traktować jako ciąg oktetów. W szczególności, jednemu wywołaniu funkcji interfejsu programowania aplikacji *send()* nie musi odpowiadać wysłanie jednego pakietu. Dane z jednego wywołania mogą zostać podzielone na kilka pakietów lub odwrotnie — dane z kilku wywołań mogą zostać połączone i wysłane jako jeden pakiet. Również funkcje odbierające dane *receive()* w praktyce odbierają nie konkretne pakiety, ale zawartość bufora stosu TCP/IP, wypełnianego sukcesywnie danymi z przychodzących pakietów.

* * *

Zadanie 1.

Napisz konsolowy program serwera o nazwie `Date`, który po nawiązaniu połączenia z klientem po protokole TCP na porcie 20191 odpowie komunikatem o bieżącej dacie i godzinie na serwerze. Po udzieleniu odpowiedzi połączenie ma być zamykane a serwer ma oczekiwać na następnego klienta.

Program serwera ma działać w nieskończoność (procesu serwera można oczywiście zabić zewnętrznie).

Działanie serwera przetestuj używając programu *telnet*.

Zadanie 2.

Napisz konsolowy program serwera o nazwie `Echo`, który po nawiązaniu połączenia z klientem po protokole TCP na porcie 20192 będzie odpowiadał takimi samymi komunikatami jakie od niego otrzyma. Dialog taki ma być kontynuowany, aż do momentu wysłania przez klienta linii zawierającej pojedynczą kropkę.

Program serwera ma działać przez pewien określony czas (na przykład przez 2 minuty), a potem ma zwolnić zajmowany port i się wyłączyć.

Działanie serwera przetestuj używając programu *telnet*.

Zadanie 3.

Napisz konsolowy program serwera o nazwie `Totient`, który po nawiązaniu połączenia z klientem po protokole TCP na porcie 20193 będzie mu odpowiadał uporządkowanym ciągiem liczb naturalnych, które są nie większe i względnie pierwsze z zadaną liczbą naturalną (odpowiedź serwera ma oczywisty związek z funkcją Φ Eulera). Jeśli przesłane przez klienta dane nie zawierają liczby naturalnej, to serwer ma odpowiedzieć komunikatem o błędzie. Dialog taki ma być kontynuowany, aż do momentu wysłania przez klienta linii zawierającej pojedynczą kropkę.

Program serwera ma działać do momentu, aż oprator wpisze polecenie `quit` (stwórz osobny wątek na serwerze, czytający linia po linii polecenia od operatora).

Działanie serwera przetestuj używając programu *telnet*.

Zadanie 4.

Zaprogramuj aplikację sieciową do gry w liczby. Gra ma przebiegać według następującego schematu: serwer po nawiązaniu połączenia z klientem losuje liczbę naturalną z zakresu 1000...9999, a klient stara się ją odgadnąć wysyłają swoje typowania. Serwer odpowiada czy podana liczba jest mniejsza, większa czy równa od propozycji przesłanej przez klienta. Jeśli klient odgadnie wylosowaną przez serwer liczbę przed 13 pytaniem, to zostaje zwycięzcą, w przeciwnym przypadku przegrywa.

Napisz w technologii *Swing* albo *FX* okienkowy program serwera gry w zgadywanie liczb o nazwie `WhatTheNumber`. Po nawiązaniu połączenia po protokole TCP na porcie 20194 serwer gra z klientem według opisanych powyżej reguł. Aplikacja powinna mieć możliwość uruchomienia i zatrzymania serwera. Serwer nie kolejkuje połączeń (obsługuje co najwyżej jednego klienta w danym momencie). Przetestuj wstępnie jego działanie za pomocą programu *telnet*.

Napisz także w technologii *Swing* albo *FX* okienkowy program klienta o nazwie `SelectNumber`, która po nawiązaniu połączenia z serwerem umożliwi granie według opisanych powyżej reguł. Klient powinien mieć możliwość poddania się przed rozstrzygnięciem gry.

Wskazówka.

Program *telnet* uruchamiamy z dwoma paramterami: nazwą hosta i numerem portu.

```
> telnet localhost 20190
```

Program *telnet* działa w ten sposób, że po wysłaniu danych do serwera oczekuje na jego odpowiedź i wypisuje ją na standardowe wyjście. Zaraz po nawiązaniu połączenia *telnet* oczekuje na pierwszą porcję danych (serwer powinien pierwszy napisać coś do klienta).

Program *telnet* jest standardowym programem narzędziowym pod Linuxem. Po Windowsem należy włączyć odpowiednią funkcję, aby *telnet* zadziałał:

(i) z menu kontekstowego przycisku otwierającego główne menu Windowsów (standardowo w lewym dolnym rogu) wybierz opcję *Aplikacje i funkcje*;

(ii) na samym dole okna *Aplikacje i funkcje* w części *Powiązane ustawienia* wybierz opcję *Programy i funkcje*;

(iii) w lewym panelu okna *Programy i funkcje* wybierz opcję *Włącz lub wyłącz funkce systemu Windows*;

(iv) w oknie *Funkcje systemu Windows* znajdź i zaznacz opcję *Klient Telnet*.

Uwaga.

Ogólny schemat pracy serwera:

```
// = zajęcie lokalnego portu przez serwer =
final int port = 20190;
ServerSocket serwer = null;
try {
    serwer = new ServerSocket(port);
    System.err.println("= serwer zajął port " + port + " =");
}
catch (IOException ex) {
    System.err.println("= nie można zająć portu " + port + " =");
    return;
}
    // - akceptowanie przychodzących połączeń w pętli -
    boolean warunek = true;
    while (warunek)
    {
        Socket polaczenie = null;
        try {
            polaczenie = serwer.accept();
            System.err.println("- nawiazano połączenie (" + polaczenie + ") -");
            BufferedReader we = new BufferedReader(
                new InputStreamReader(polaczenie.getInputStream()));
            PrintWriter wy = new PrintWriter(
                new OutputStreamWriter(polaczenie.getOutputStream()),true);
            // ... dzialanie serwera poprzez gniazdo robocze
            // ... wymiana informacji z klientem poprzez strumienie
            // ... byc moze zmiana warunku w petli
            System.err.println("- zamknieto połączenie (" + polaczenie + ") -");
        }
        catch (IOException ex) { System.err.println("- błąd gniazda roboczego -"); }
        finally
        {
            try { if (polaczenie != null) polaczenie.close(); }
            catch (IOException ex) { }
        }
    }
try { serwer.close(); }
catch (IOException ex) { }
System.err.println("= serwer zwolnił port " + port + " =");
```