第A章　Javaによるネットワークプログラミング（発展編）

ネットワークプログラムでは、標準入力とソケットからの入力など、複数の入力を待ち受ける必要がある場合が必然的に多くなる。このような場合は、ある入力を待ち受けるためにブロック（ストップ）してしまって、他の入力があるのにそれに反応できない、という状況は避けなくてはならない。

A.1　スレッドを用いた複数の入出力への対処

これに対処する方法としてひとつ考えられるのが、入力があるかどうかいちいち調べる方法（＿＿＿＿＿＿＿＿＿＿＿）である。（Java の場合、InputStream クラスの ＿＿＿＿＿＿＿＿というメソッドを用いる）

```
while (true) {
   if (input1.available() > 0) {
     ...  // input1 に対する処理
   } else if (input2.available() > 0) {
     ...  // input2 に対する処理
   } ...
}
```

しかし、この方法は、＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ので、望ましくない。

Java では次の例のようにスレッドを使って複数の入力を待ち受けることができる。

例題 A.1.1 スレッドを使った例

ファイル *TCPThread.java*

```
import java.net.*;
import java.io.*;

public class TCPThread {
  public static void main(String[] argv) {
    try {
      Socket rwSocket   = new Socket(argv[0], Integer.parseInt(argv[1]));
      InputStream  instrm = rwSocket.getInputStream();
      OutputStream outstr = rwSocket.getOutputStream();

      Thread input_thread  = new Thread(new StreamConnector(System.in, outstr));
      Thread output_thread = new Thread(new StreamConnector(instrm, System.out));
      input_thread.start(); output_thread.start();
    } catch (Exception e) {
      e.printStackTrace(); System.exit(1);
    }
  }
}
```

このクラスでは *StreamConnector* という補助的なクラスを定義している。*StreamConnector* の ____ メソッドがスレッドで実行される。独立したスレッドの中で入力を待つので、標準入力を待っている状態でも、ソケットからの入力に対応することができる。

ファイル *TCPThread.java*（続き）

```
class StreamConnector implements Runnable {
  InputStream src  = null;
  OutputStream dist = null;
  // コンストラクタ 入出力ストリームを受け取る
  public StreamConnector(InputStream in, OutputStream out){
    src = in;
    dist = out;
  }

  // 処理の本体: ストリームの読み書きを無限に繰り返す
  public void run(){
    byte[] buff = new byte[1024];
    while (true) {
      try {
        int n = src.read(buff);
        if (n > 0)
          dist.write(buff, 0, n);
      }
      catch(Exception e){
        e.printStackTrace(System.err);  System.exit(1);
      }
    }
  }
}
```

.....
.....

使用法は、

`java TCPThread サーバホスト名 ポート番号`

である。

問 A.1.2 複数の *HTTP* サーバに同時接続してファイルをダウンロードし、さらにユーザから新規接続の要求も受け取るプログラムを作成せよ。

A.2　サーバのプログラミング

HTTP サーバや Telenet サーバなどのサーバは多数のクライアントからの接続を受け付けなければならないので、クライアント側とは異なる形でソケットを利用する。Java では _____ というクラスを用いる。

例題 A.2.1 コネクション型（*TCP*）（サーバ側）

ファイル Pphttpd.java

```
import java.io.*;
import java.net.*;

public class Pphttpd{
  public static void main(String args[]){
    try {
      // サーバ用ソケットの作成
      ServerSocket servsock = new ServerSocket(Integer.parseInt(args[0]));
      while(true){
        Socket sock = servsock.accept(); // 接続要求の受付
        // 以下の処理は、時間がかかる場合は、
        // 本来はすぐに接続要求の受付に戻れるように、スレッドで行なうべきである。

        // 接続先の表示
        System.out.println("Request from "
                           + (sock.getInetAddress()).getHostName());
        // 効率を考慮してバッファを利用する。
        // （1 文字ずつではなく、まとめて読めるようにする。）
        BufferedReader in = new BufferedReader(
                  new InputStreamReader(sock.getInputStream()));
        // println メソッドが使えるように PrintStream クラスを用いる
        PrintStream out = new PrintStream(sock.getOutputStream());
        // とりあえず改行を 2 つ読み飛ばす
        int i ;
        for(i=0; i<2; i++) {
          in.readLine();
        }
        out.println("<html>");
        out.println("<head><title>Test</title></head>");
        out.println("<body>Hello!</body>");
        out.println("</html>") ;
        // 接続終了
        sock.close() ;
      }
    } catch (Exception e){
      e.printStackTrace();  System.exit(1) ;
    }
  }
}
```

.....

.....

ServerSocket のコンストラクタの引数はポート番号である。

クライアントからの接続要求の受け付けは、_____メソッドで行なう。

```
sock = servsock.accept();
```

このメソッドはクライアントからの接続要求があるまで待ち、新しい *Socket* クラスのインスタンスを返す。クライアントとの通信は、この新しい *Socket* を通じて行なう。*ServerSocket* の方は、次のクライアントからの接続要求のために再び利用する。

このプログラムを例えば、

```
java Pphttpd 8080
```

というように *8080* 番のポートで起動して、*Netscape* などで *URL* を *http://XXX.XXX.XXX.XXX:8080/*（*XXX.XXX.XXX.XXX* の部分は、*Pphttpd* を起動したマシンのホスト名か *IP* アドレス）と指定する。すると *"Hello!"* という内容だけの *Web* ページがあるかのように表示される。

WindowsXP の場合、マシンの *IP* アドレスは *ipconfig* コマンドで調べることができる。また、*IP* アドレス *127.0.0.1* は必ず自分自身を指すので、*Pphttpd* とクライアントを同じマシンで実行する時は、*127.0.0.1* を使うこともできる。

問 A.2.2 接続要求を受け付けると、別の *Web* サーバに要求をそのまま中継して、サーバから受信したデータをそのままクライアントに送るプログラム（超簡易 *proxy* サーバ）を書け。

問 A.2.3 アクセスカウンタ付 *Web* ページを配信する（偽）*HTTP* サーバプログラムを書け。

問 A.2.4 時計付 *Web* ページを配信する（偽）*HTTP* サーバプログラムを書け。

問 A.2.5 （難）オセロや麻雀などのネットワーク対戦型ゲームのサーバとクライアントを作成せよ。

キーワード　ビジーウェイト、スレッド、ServerSocket クラス、accept メソッド、PrintStream クラス、