Java独習　第3版

12.1 インターネットアドレス
12.2 サーバーソケットとソケット

2006年7月5日(水)　　南　慶典

12.1 インターネットアドレス

インターネットアドレス

インターネットアドレス

- 32ビットの長さを持つ
- インターネットに接続されたマシンを識別するのに使う。

インターネットアドレスは、ピリオドで区切られたトークンの並びで表現されることもある

「www.mycompany.com」

インターネットアドレスをドットストリング表記からドット10進表記へ変換するのがDNS（Domain Name System）の役割。

- ドットストリング表記（「www.obsorne.com」など）
- ドット10進表記(「200.200.200.200」など)

InetAddressクラス

- java.net パッケージのInetAddress クラスは、インターネットアドレスをカプセル化している

InetAddressクラスに定義されている主なインスタンスメソッド

メソッド	説明
byte[] getAddress()	アドレス情報を含むバイトの配列を返す。データはネットワークバイトオーダー（最初の要素が上位バイト）で格納される
String getHostAddress()	アドレス情報を表す文字列を返す
String getHostName()	ホスト名を表す文字列を返す

InetAddressのメソッド ①

getByName()メソッド

```
static InetAddress getByName(String hostName) throws
UnknownHostException
```

DNSによって提供される情報を使って、名前からアドレスへの変換を実行する

getAllByName()メソッド

```
static InetAddress getAllByName(String hostName) throws
UnknownHostException
```

ホストが複数のアドレスもっている場合使用する。InetAddressオブジェクトの配列を取得できる。

InetAddressのメソッド ②

２つのメソッドのhostNameには、インターネットのホストの名前を指定。

- ドットストリング表記（「www.obsorne.com」など）
- ドット10進表記(「200.200.200.200」など)

getLocalHost()メソッド

```
static InetAddress getLocalHost throws
UnknownHostException
```

このメソッドからは、ローカルホストの情報をカプセル化したInetAddressオブジェクトが返される。

プログラム例

```
import java.net.*;

class InetAddressDemo {
    public static void main(String[] args) {
        try{
            // アドレスを取得する
            InetAddress ias[] =  InetAddress.getAllByName(args[0]);

            for (int i=0; i<ias.length;i++){
                System.out.println(ias[i].getHostName());
                System.out.println(ias[i].getHostAddress());
                byte bytes[] = ias[i].getAddress();

                for(int j=0;j<bytes.length;j++){
                    if(j>0)
                        System.out.print(".");
                    if(bytes[j] >=0)
```

```
                    System.out.print(bytes[j]);
                else
                    System.out.print(bytes[j]+256);
            }
            System.out.println("");
        }
    }
    catch(Exception e){
        e.printStackTrace();
    }
  }
}
```

実行結果

```
>java InetAddressDemo localhost
localhost
127.0.0.1
127.0.0.1
```

12.2 サーバーソケットとソケット

ソケット

- ソケットとは、2つのマシン間の双方向通信経路の一端
- 2つのアプリケーションが、信頼性のある順次データ交換を行うためのメカニズムを提供。
  ⇒ これはソケットがTCP(Transmission Control Protocol)と
  IP（Internet Protocol）を使用することによって実現されている。
- ServerSocket クラスと Socket クラスはクライアント/サーバーアプリケーションを作成するのに使用する。

ServerSocket コンストラクタ

ServerSocketコンストラクタ

ServerSocket(int port) throws IOException

portは、クライアントからの要求を監視するためのソフトウェアポート。ほかの形式のコンストラクタは、送られてくる要求の待ち行列を制限したり、特定のアドレスをバインド（監視）したりするものである。

accept()メソッド

Socket accept() throws IOException

- クライアントから送られてくる要求を監視する。このメソッドは要求が到着するまで待機する。
- accept()メソッドからは、クライアントとの通信に使用するSocketオブジェクトが返される。

close()メソッド

void close() throws IOException

- サーバーソケットをクローズする。

Socket

クライアントとサーバーのデータ交換には、Socketクラスを使う。このクラスのコンストラクタの１つを次に示す。

Socketクラス

```
Socket(String hostName, int port) throws UnknownHostException, IOException
```

- hostNameはサーバーホストの名前で、ドットストリング表記でもドット10進表記でも構わない
- portは、そのサーバーのソフトウェアポートで、このソケットの接続先
- ソケットの作成後は、通信に使う入力ストリームと出力ストリームを取得しなければならない

Socket

getInputStream()メソッド、getOutputStream()メソッド

```
InputStream getInputStream() throws IOException
OutputStream getOutputStream() throws IOException
```

InputStreamオブジェクトとOutputStreamオブジェクトは通常、それぞれDateInputStreamオブジェクトとDataoutputStreamオブジェクトを作成するのに使用する。

close()メソッド

```
void close() throws IOException
```

- サーバーソケットをクローズする。

例12.2　サーバーアプリケーション

次の例は、簡単なクライアント/サーバーアプリケーションであり、クライアントはサーバーに接続して、乱数を取得し、それを表示する。

サーバーソフトウェア

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

public class ServerSocketDemo {
    public static void main(String[] args) {
        try{
            // ポートを取得する
            int port = Integer.parseInt(args[0]);

            // 乱数ジェネレータを作成する
            Random random = new Random();

            // サーバーソケットを作成する
            ServerSocket ss = new ServerSocket(port);
```

```
            // 無限ループを作成する
            while(true){

                // クライアントからの要求を受け取る
                Socket s = ss.accept();

                //結果をクライアントに書き込む
                OutputStream os = s.getOutputStream();
                DataOutputStream dos = new DataOutputStream(os);
                dos.writeInt(random.nextInt());

                // ソケットをクローズする
                s.close();
            }
        }
        catch(Exception e){
            System.out.println("Exception :"+e);
        }
    }
}
```

例12.2　クライアントアプリケーション

クライアントソフトウェア

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

class SocketDemo {
    public static void main(String[] args) {
        try{
            // サーバーとポートを取得する
            String server = args[0];
            int port = Integer.parseInt(args[1]);

            // ソケットを作成する
            Socket s = new Socket(server, port);

            // サーバーから乱数を読み取る
            InputStream is = s.getInputStream();
            DataInputStream dis = new DataInputStream(is);
            int i = dis.readInt();
```

```
                // 結果を表示する
                System.out.println(i);

                // ソケットをクローズする
                s.close();
            }
            catch(Exception e){
                System.out.println("Exception: " + e);
            }
        }
}
```

>java ServerSocketDemo 4231

- 4231は送られてくる要求の到着先となるソフトウェアポート。
- 4231でなくても構わないが、1024より下の番号は避ける。

>java SocketDemo 127.0.0.1 4231

- 127.0.0.1 はローカルマシンを表す。
- ２番目の引数はサーバーアプリケーションの時と同じポート番号を指定しなければならない。

SocketDemoアプリケーションは、乱数を表示すると終了

実行結果

34889301

練習問題

問題1
クライアントはサーバーに接続して文字列を取得しそれを表示させる、簡単なクライアント/サーバーアプリケーションを作成しなさい。ただしサーバープログラムではコマンドライン引数としてポート番号を、クライアントプログラムではサーバー名とポート番号を指定してください。

問題2
無限ループを実行するサーバー作成し、クライアントから送られてくるdouble型の値を読み取りなさい。読み取った値を2乗し、その結果をクライアントに書き込みなさい。また、double型の値をサーバーアプリケーションに送るクライアントアプリケーションを作成しなさい。double型の値は、クライアントアプリケーションのコマンドライン引数として指定するようにしなさい。