インターネットプログラミング
教室2406
水曜日7限19:50-21:20

第10回
2013/11/27

岩井将行

授業資料

- http://www.cps.im.dendai.ac.jp

- http://www.cps.im.dendai.ac.jp/index.php?Classes%2F2013InternetProg

講師紹介

- http://cps.im.dendai.ac.jp/index.php?Members%2Fiwai

- 岩井研究室
- http://cps.im.dendai.ac.jp

- 岩井研究室の研究分野
- http://cps.im.dendai.ac.jp/index.php?Research%2FTopics

- 連絡先　1号館11F 11107b
- 　　iwaiあっと im.dendai
- 研究室内線2844

講師

- 慶應義塾大学　卒
- 元東京電機大学　戸辺義人先生
  OSOITEProject参加
- 東京大学生産技術研究所　助教
- 2013.4月より未来科学部情報メディア学科
  准教授

- 詳しくはFacebook　岩井将行
- Twitter @masaiwai

TA・SA・副手

- 加藤 佳祐
- 鉄谷研究室
- 東京電機大学 未来科学部 情報メディア学科
- E-mail： Keisuke Kato
- <case.unl[atmark--].-gmail.com>

ソケット通信

プログラム同士の通信は

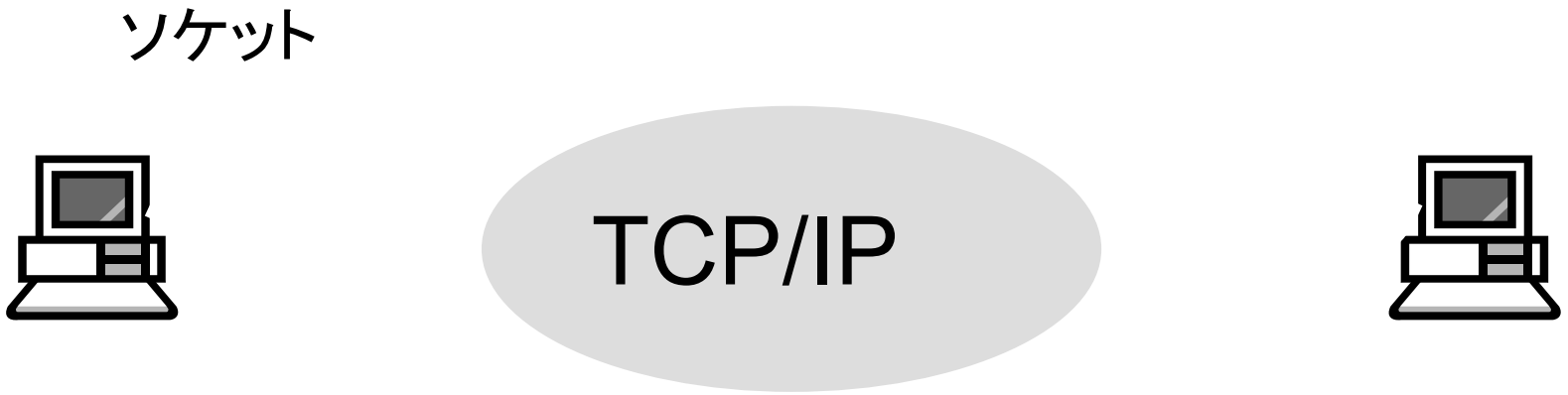
- ソケットを使ってデータの送受信
- ソケットを使った通信

ソケット通信

ソケット

意味:「接続の端点」

コンピュータとTCP/IPを
つなぐ出入り口



ソケット通信

- ソケットを使って通信を行うには
  2つのプログラムが必要

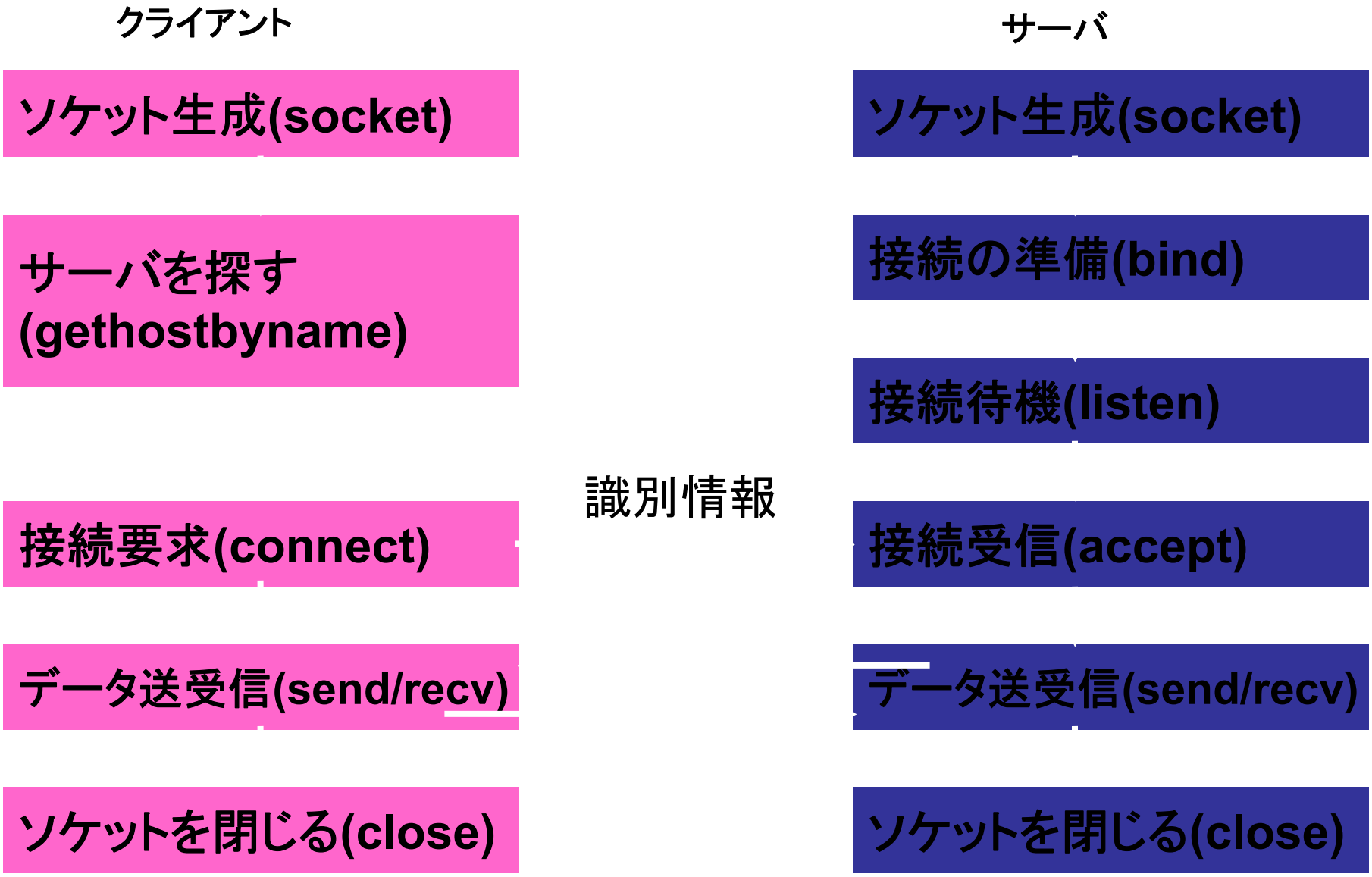
クライアントプログラム

ソケットを用意して
サーバに接続要求を行う

サーバプログラム

ソケットを用意して接続要求を待つ

ソケット通信の全体の流れ

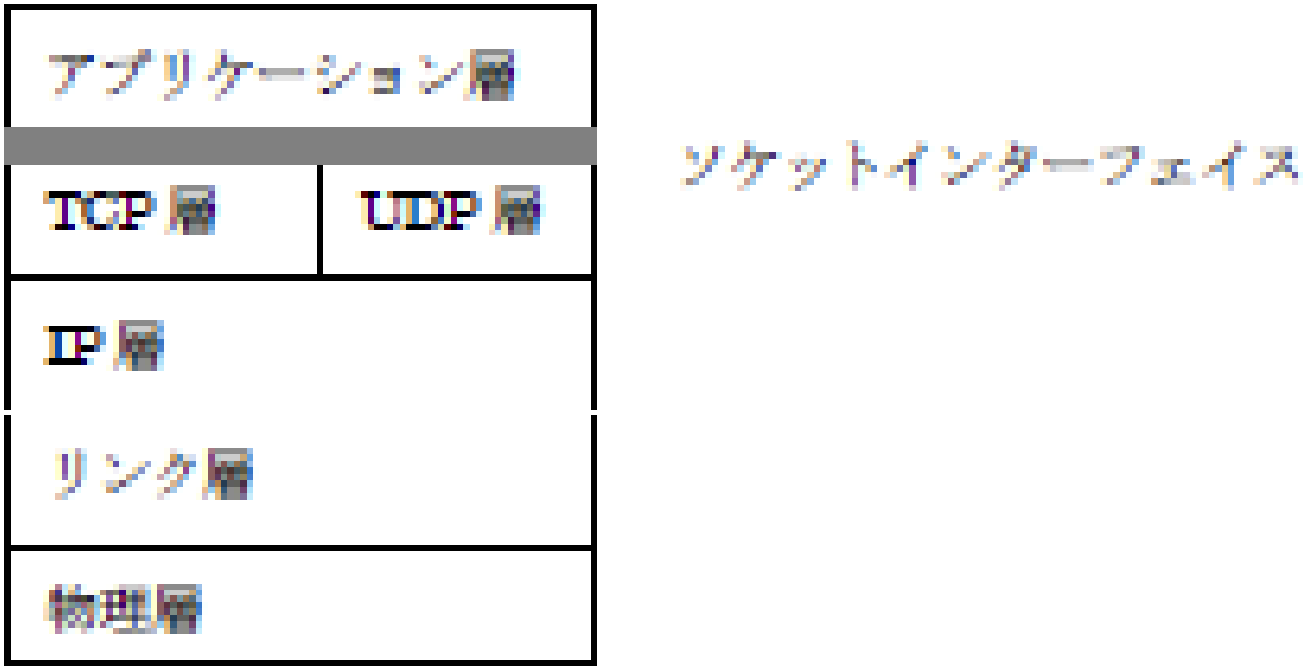


動かしてみよう。

- SwingAnimationBasic
- SwingAnimationFaceObj

- Swing　より美しいグラフィックスを提供 java2D
- ちらつき防止
- http://www.java2s.com/Tutorial/Java/0240__Swing/Catalog0240__Swing.htm

Socket interface



UDP通信

- UDP通信では接続という概念はない。
- ユーザは毎回データを送信し、また毎回自分のソケット、及び相手のソケットとアドレスを指定することになる。TCPでは最初に相手のソケットとの間の接続を行い、双方のソケット間の接続が確立されたら、互いの通信が可能になる。
- TCPではそのような制限はない。
- TCPでは接続が確立されたら2つのソケットはストリームのように振る舞う。TCPでは受信したデータの信頼性と順序が保障される。

DatagramPacket

- SocketやServerSocketでは、データのやりとりは入出力ストリームに抽象化されていたため、パケットという概念が見えてこなかった。
- DatagramPacketとDatagramSocketの場合はUDPパケットはUDPパケットとして抽象化されており、パケットにデータも送信先アドレスも含める必要がある。
- SocketやServerSocketとは別物

UDPServer

```
int serverPort = 5000;

DatagramSocket socket = new
    DatagramSocket(serverPort);

DatagramPacket receivePacket = new
    DatagramPacket(new byte[DMAX], DMAX);

socket.receive(receivePacket);



socket.close();
```

Udp Client

```
String servhostname="localhost";

InetAddress serverAddress =
    InetAddress.getByName(servhostname);
String message="hello UDP from yourname";
byte[] bytesToSend = message.getBytes();
int serverPort = 5000;
DatagramSocket socket = new DatagramSocket();
DatagramPacket sendPacket = new
    DatagramPacket(bytesToSend, bytesToSend.length,
    serverAddress, serverPort);
socket.send(sendPacket);
socket.close();
```

今週の課題[UDPの送受信をやってみよう]

- 堤出フォルダは第9回
- DendaiUDPClient1 DendaiUDPServ1を改造して、「送信したメッセージを真逆にして返してくる」
- 送信時はDendaiUDPServer5000番ポート
  返信時にはDendaiUDPClientの5001番ポートを指定すること

Innerクラスの整理
動かしてみよう

- ClassA

ソケット通信

プログラム同士の通信は

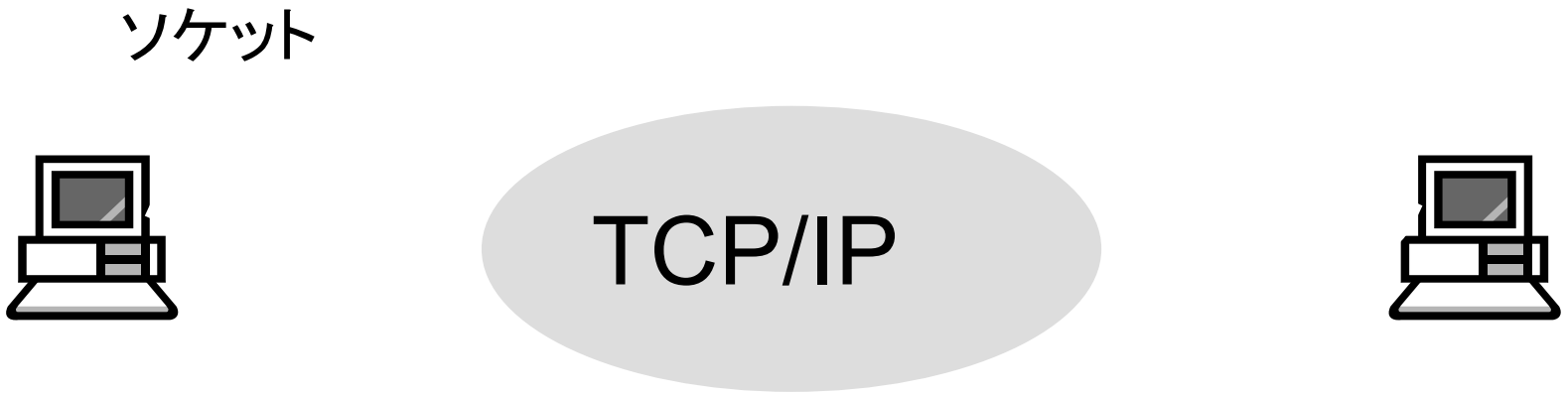
- ソケットを使ってデータの送受信
- ソケットを使った通信

ソケット通信

ソケット

意味:「接続の端点」

コンピュータとTCP/IPを
つなぐ出入り口



ソケット通信

- ソケットを使って通信を行うには
  2つのプログラムが必要

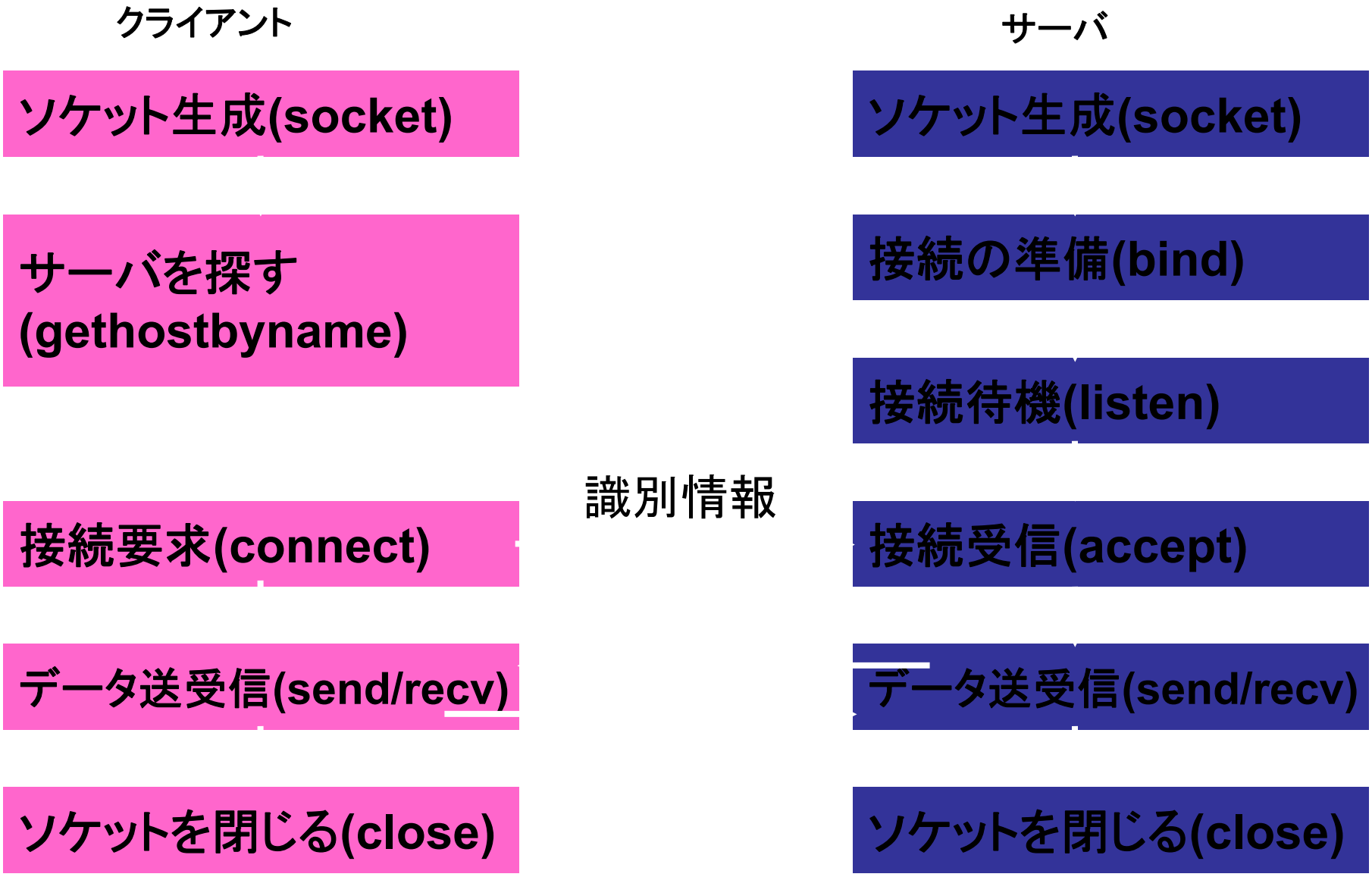
クライアントプログラム

ソケットを用意して
サーバに接続要求を行う

サーバプログラム

ソケットを用意して接続要求を待つ

ソケット通信の全体の流れ

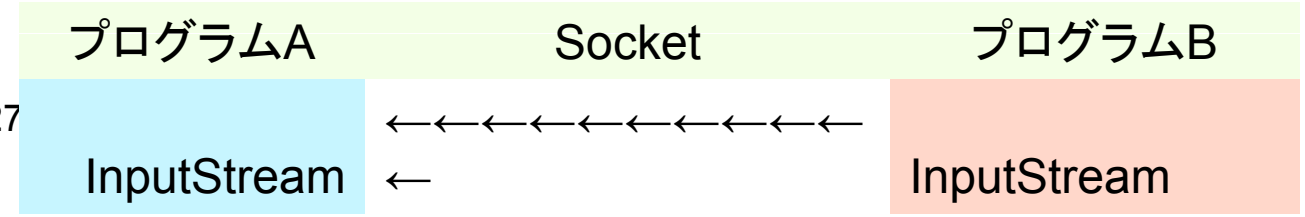


Sokect通信

- 双方がデータのやりとりを行う場合、双方を結んで情報を運ぶための「線」が必要となります。Javaにおいてはこの「線」のことを「***Socket（ソケット）***」という概念で表します。「Socket」は逆方向の情報の流れ（Stream）をもつ二本の通信線を一本に束ねたものだと考えられます。

InputStream, OutptStream

- 仮にAとBという二つのプログラムが通信を行うとすれば、一方の通信線がAにとっての「InputStream」でありかつBにとっての「OutptStream」、もう一方の通信線はAにとっての「OutputSteam」かつBにとっての「InputStream」となる。
- この2本の通信線を束ねる「Socket」を介して情報のやりとりが行われる。



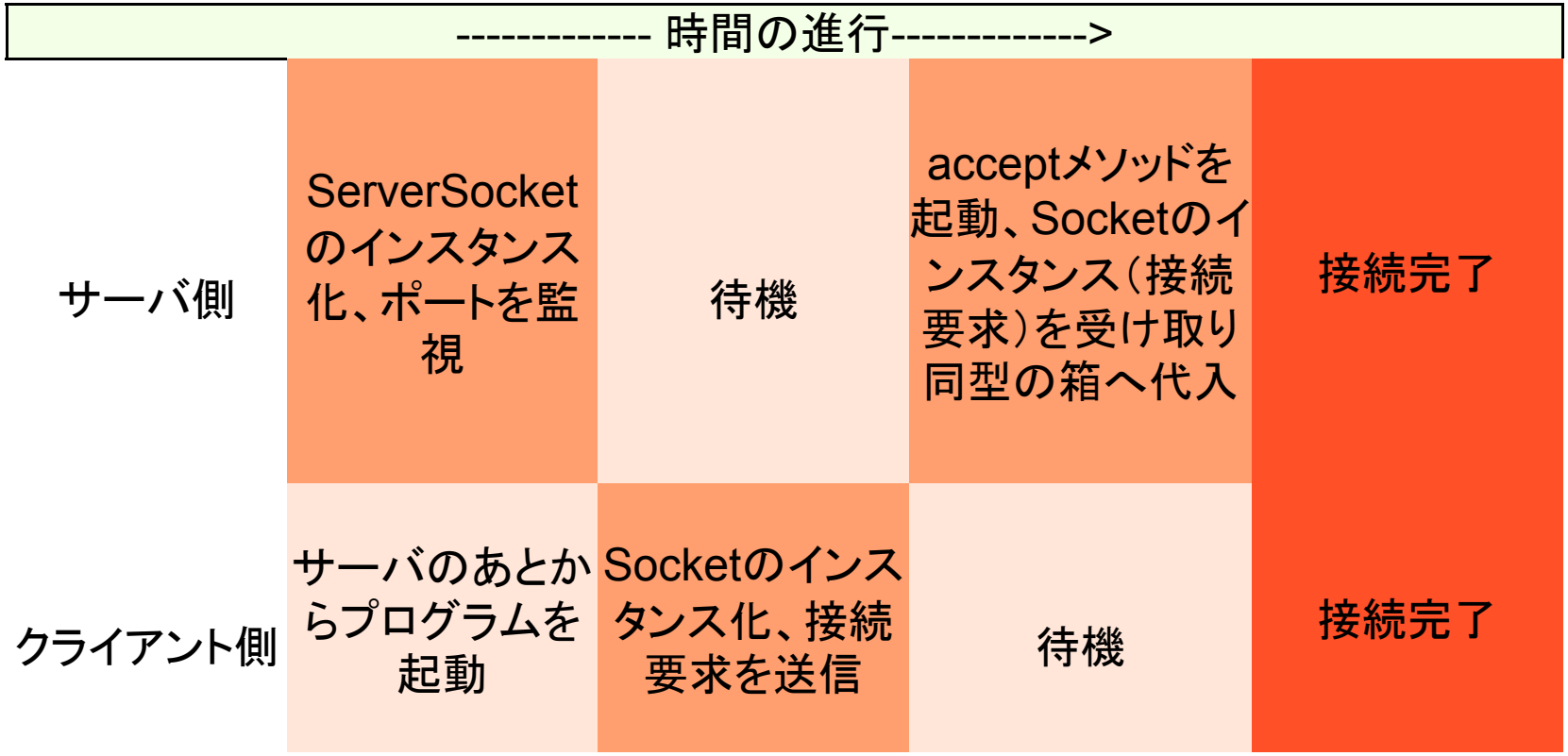
Socketとport番号

- ***クライアント側からサーバ側に向かって「Socket」が送り込まれて、初めて通信が開通***

- クライアントが「Socket」を送り込む際には、必ず「どのサーバコンピュータ」の「何番の通信口」に向かって送り込むのかを明らかにしなければなりません。
- その際の「どのサーバコンピュータ」のことを一般に***「サーバ名」***、「何番の通信口」のことを***「ポート番号」***、と呼びます。サーバには複数の「通信口＝ポート」があり、番号で管理しています。サーバの側は、プログラムによって決まった番号のポートを監視しています。
- クライアントはサーバが監視するポート番号に「Socket」を送らなければなりません。

ServerSocketクラス、Socketクラス

- **ServerSocketクラス**　サーバの側に用いられます。インスタンス化の際に引数として「ポート番号（整数型）」を要求します。インスタンス化が済んだ時点でポートの監視が始まります。そのとき、もしもクライアントから接続要求（「Socketクラス（後ほど説明）」のインスタンスが送られてくる）があれば、「acceptメソッド」で受け取ります。この時点で通信の準備が完了します。
- **Socketクラス**　クライアントの側に用いられます。インスタンス化の際に引数として「サーバ名（文字列型）、ポート番号（整数型）」の順に二つの引数を要求します。インスタンス化が行われた時点で接続要求がサーバに送られます。


---------- 時間の進行---------->
サーバ側
ServerSocketのインスタンス化、ポートを監視
待機
acceptメソッドを起動、Socketのインスタンス（接続要求）を受け取り同型の箱へ代入
接続完了
クライアント側
サーバのあとからプログラムを起動
Socketのインスタンス化、接続要求を送信
待機
接続完了

動かしてみよう1

- DendaiTCPServ1
- DendaiTCPClient1

Objectのやりとり

- OutputStreamへのデータの書き込み　→　データの送信
- InputStreamからのデータの読み込み　→　データの受信

- **OutputStream/InputStreamの取得**　OutputStream/InputStreamの取得を行うためには、Socketクラスのメソッドである「***getOutputStream/getInputStream」メソッド***を利用します。これらのメソッドが起動されると、返値として、取得したOutputStream/InputStreamのインスタンスが返されます。
- このインスタンスを引数にして、Streamを用いてデータの通信を行うクラスをインスタンス化することで、OutputStream/InputStreamが取得されデータ送受信の準備が完了します。
- 通信の際にやりとりするデータの型が「任意のクラスのインスタンス」の場合、データ通信は「***ObjectOutputSream/ObjectInputSream クラス***」を用いて行います。従ってgetOutputStream/getInputStreamメソッドの返値であるOutputStream/InputStreamインスタンスを引数として、これらのクラスのインスタンス化を行うことで、OutputStream/InputStreamが取得されデータ送受信の準備が完了します。

ObjectOutputStream、ObjectInputStream

- **データ送信の準備**（但しObjectOutputStreamのインスタンス名をoos、Socketのインスタンス名をsocketとする）
- ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
- **データ受信の準備**（但しObjectInputStreamのインスタンス名をois、Socketのインスタンス名をsocketとする）
- ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());

OutputStreamへのデータの書き込み（送信）

- OutputStreamへのデータの書き込みは、「ObjectOutputStream」の「writeObject」メソッドを利用して行います

- **データの書き込み（送信）**
- oos.writeObject(送信したいインスタンス名);
- oos.flush();

InputStreamからのデータの読み込み（受信）

- InputStreamからのデータの読み込みも、ファイルからの読み込みと同様「ObjectInputStream」の「readObject」メソッドを利用します。

- **データの読み込み（受信）**（但し、ObjectInputStreamのインスタンス名をois、読み込むデータをVector型のインスタンスとする）
- Vector vec = (Vector)ois.readObject();

通信終了の合図

- **通信終了の合図**（但し、ObjectOutputStreamのインスタンス名をoosとする）
- oos.close();

- socket.close();

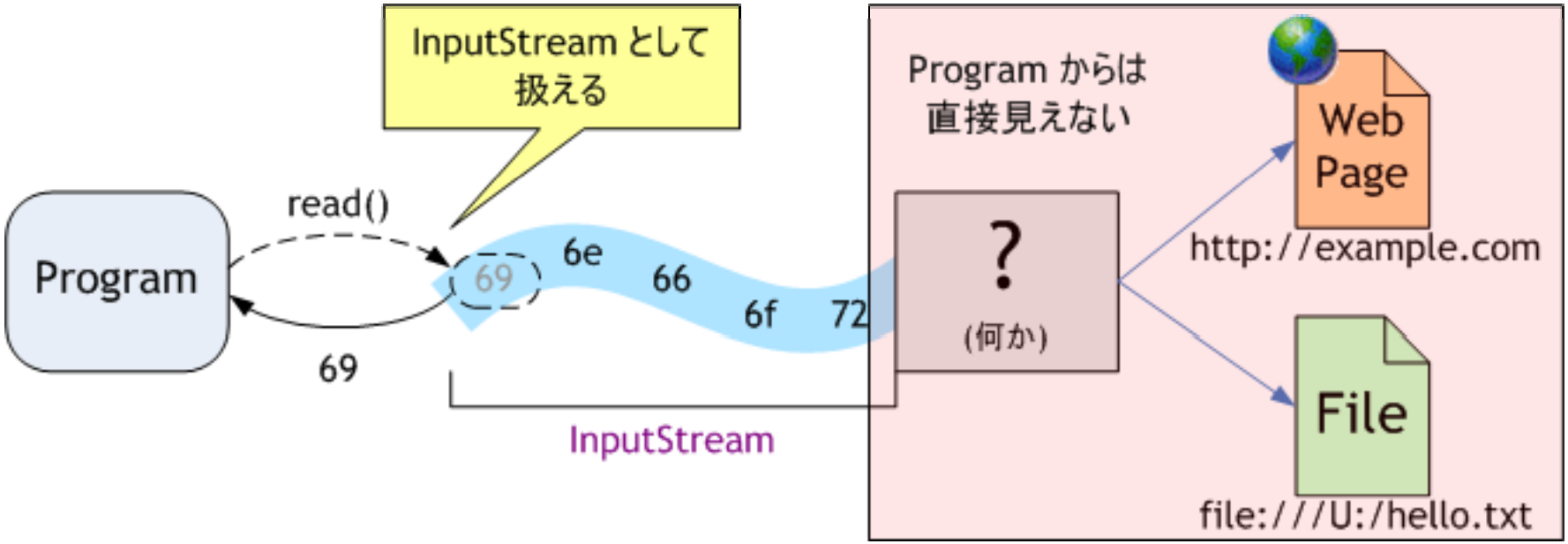
動かしてみよう

- DendaiTCPServ2Age
- DendaiTCPClient2Age

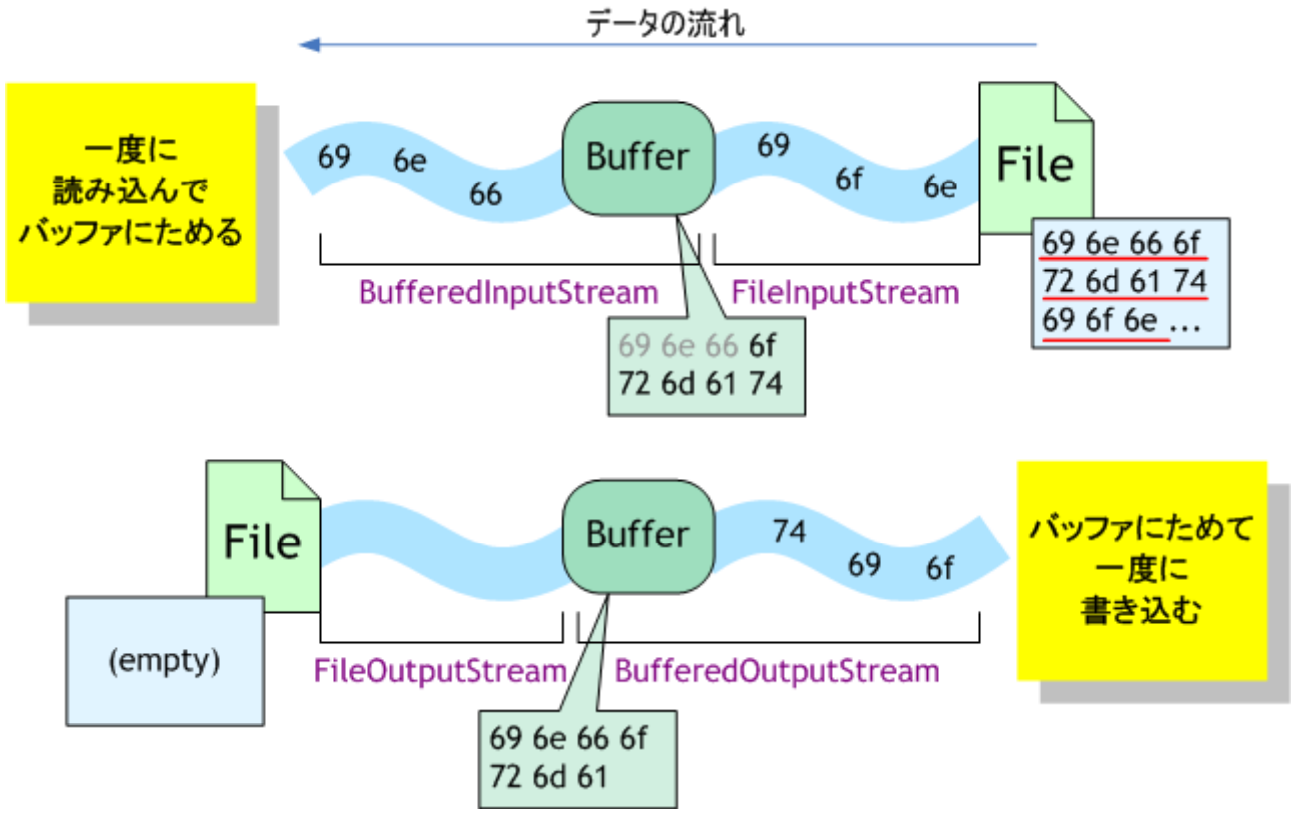
Serializable

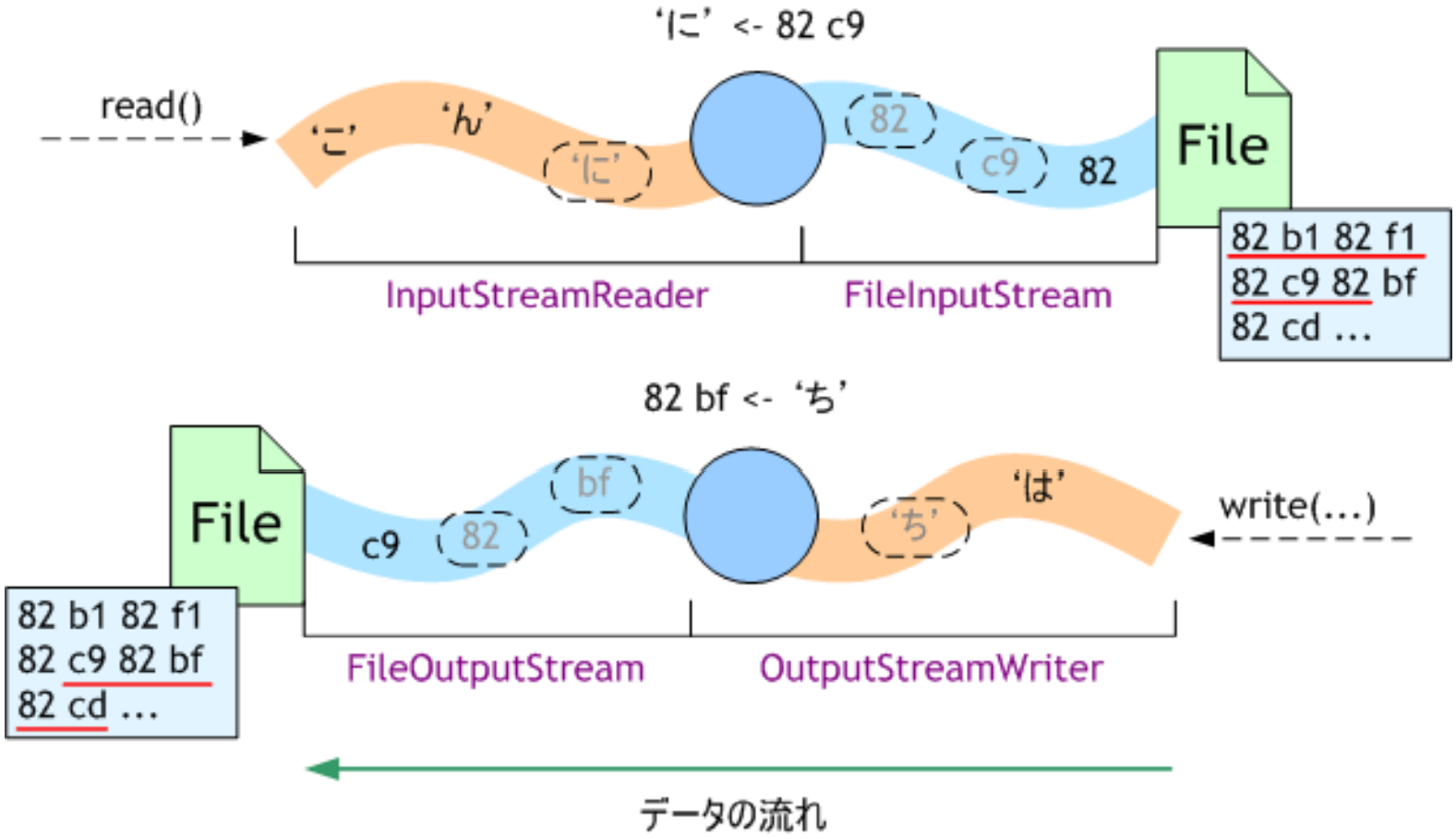
- erializableを実装したら、
- **serialVersionUID**という定数を定義
- class Test implements Serializable {
  private static final long
  serialVersionUID =
  8531245739641223373L; }
- Objectの送受信が可能になる。

IO Stream

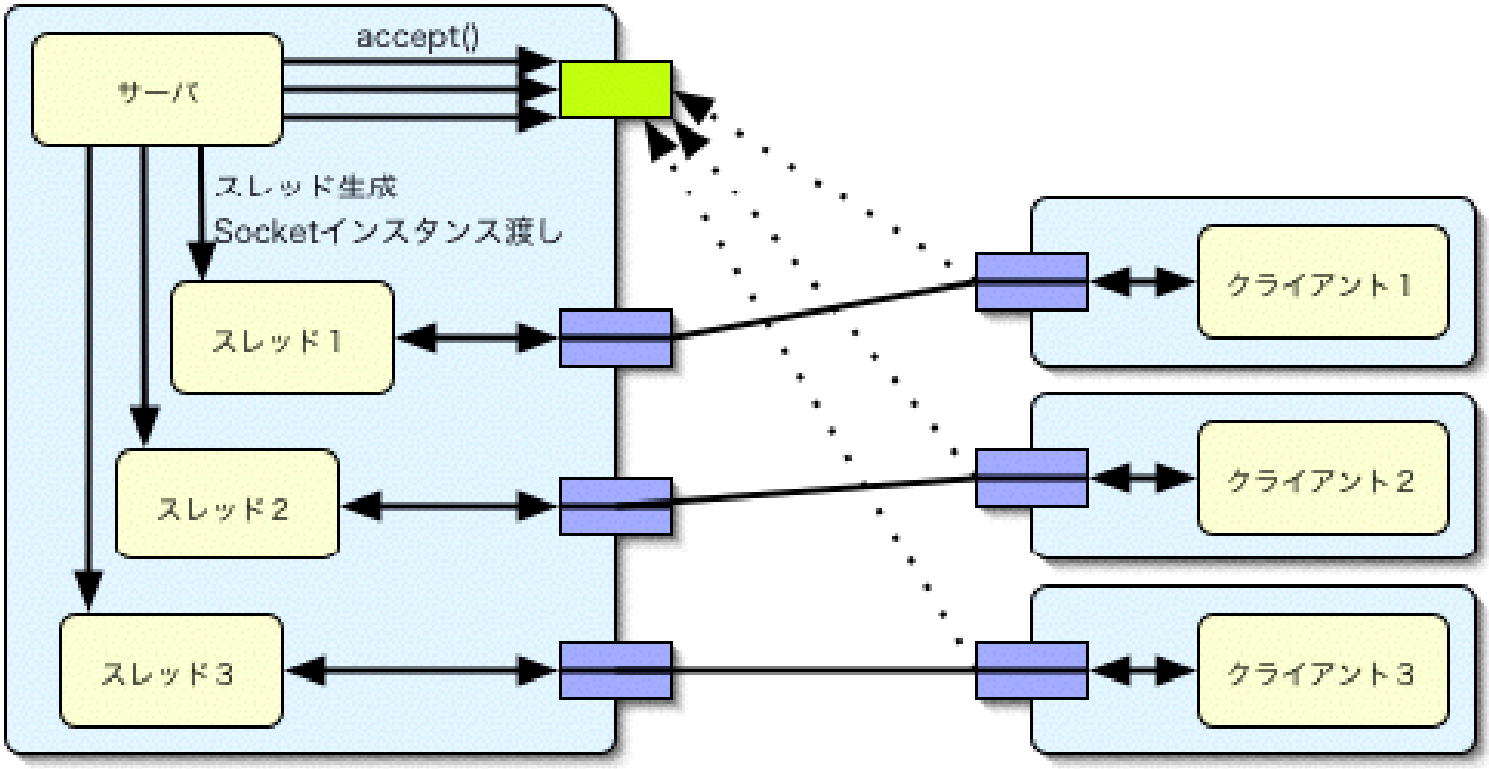

InputStream として
扱える
Program からは
直接見えない
Program
read()
69
6e
66
6f
72
69
InputStream
?
(何か)
Web
Page
http://example.com
File
file:///U:/hello.txt

BufferdI/OSream




‘に’ <- 82 c9
read()
‘こ’
‘ん’
‘に’
82
c9
82
File
82 b1 82 f1
82 c9 82 bf
82 cd …
InputStreamReader
FileInputStream
82 bf <- ‘ち’
File
c9
82
bf
‘ち’
‘は’
write(…)
82 b1 82 f1
82 c9 82 bf
82 cd …
FileOutputStream
OutputStreamWriter
データの流れ

MultiThread



動かしてみよう

- CountTenRunnable

try-catch

```
try {
    例外のスロー
} catch (Exception e ) {
     例外のキャッチ
}
```

例外を投げる

```
throw new Exception();


Public void method1 (int x) throws Exception
   {
     throw new Exception();
}
```

例外を投げる。

```
// NumberFormatException を捕捉するための try-catch
        try {
            n = Integer.parseInt(input);
        }
        catch (NumberFormatException e) {
            // 数値の形式が不正である場合は、入力自体が不正
            throw new IllegalInputException("不正な入力 " + input);
        }

        if (n < 0) {
            // 負の値が入力された場合は、不正な入力
            throw new IllegalInputException("不正な入力 " + input);
        }

        System.out.println("入力された正の値は" + n);
```

例外クラス

```
public class IllegalInputException extends
   Exception {
    public IllegalInputException(String
   message) {
        // 親クラスのコンストラクタにメッセージを
   渡す
        super(message);
    }
}
```

スタックトレース

- e.printStackTrace();　を使ってみる

うごかしてみよう。

- MultiServerSample.java
- MultiClientSample.java

今日の課題

- Basic MultiServerSample、MultiClientSample を変更して送信した10進数整数を2進数か16進数に変換すするプログラムを書け。

- AdvanceどのようにしたらDosアタックからサーバ全体が落ちることを防ぐことができるかコードで示せ。