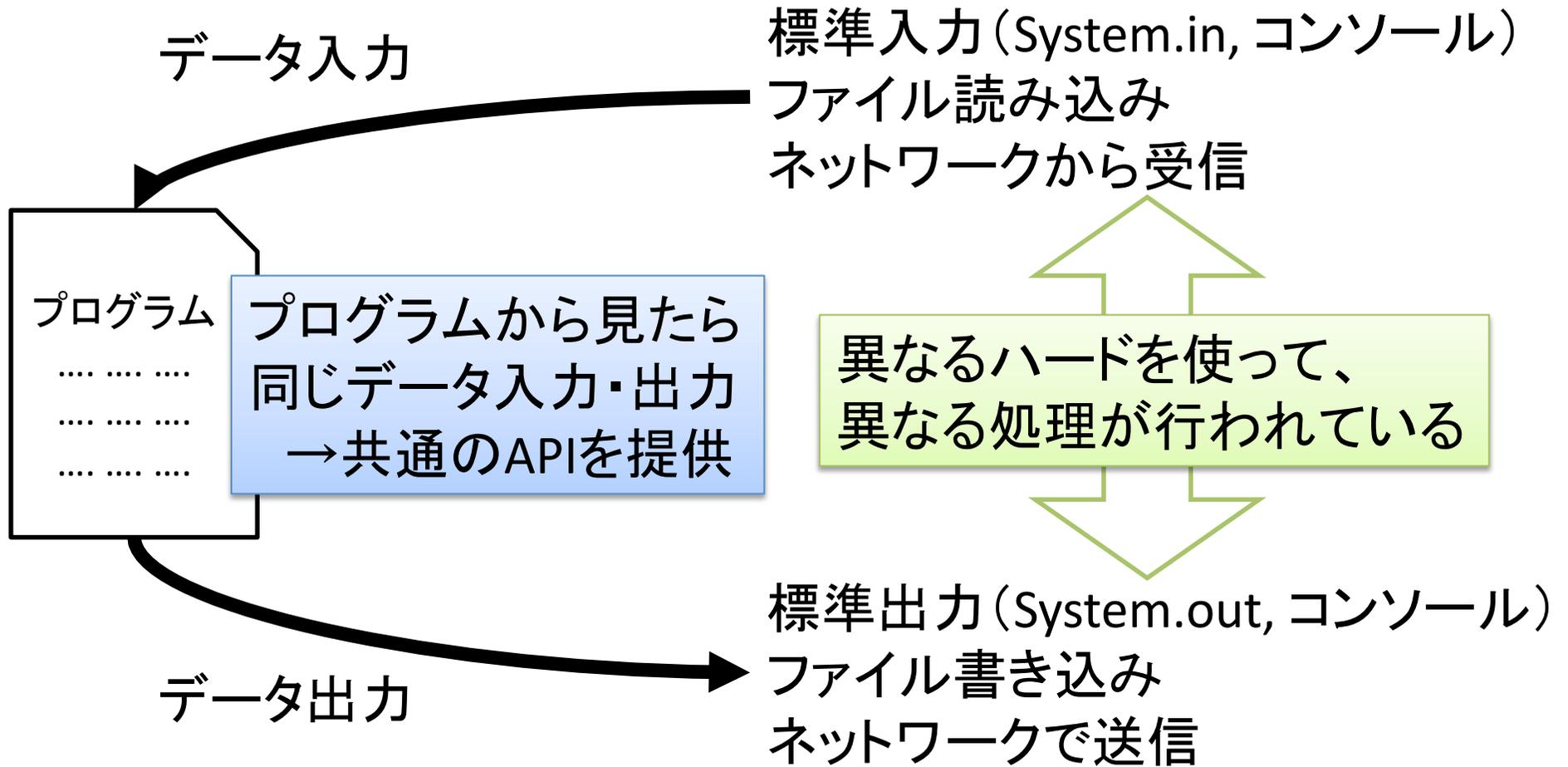


# プログラミング基礎

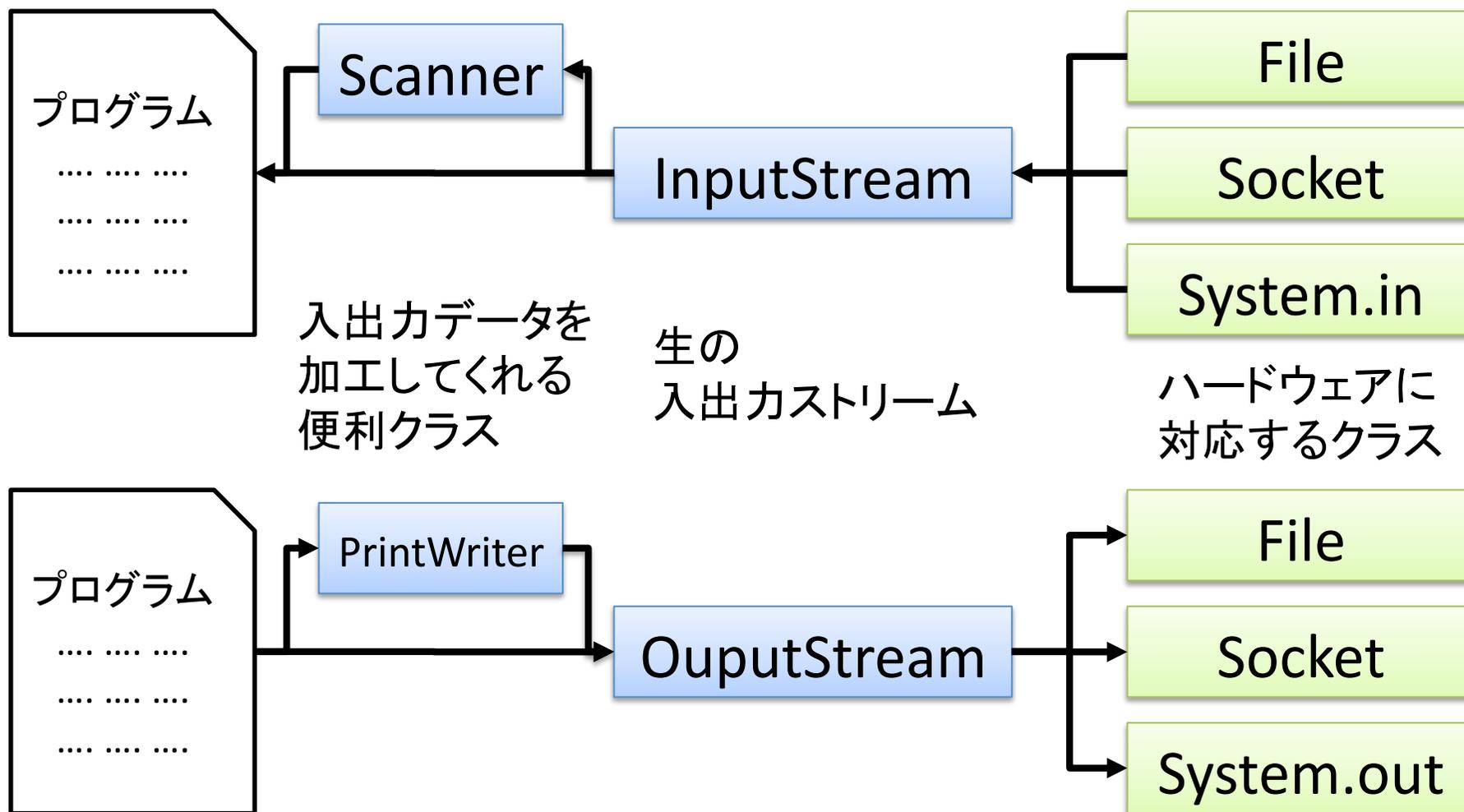
入出カストリーム

例外処理

# 入出力ストリーム (IO stream)



# Javaの入出力API



# 例外処理 (exception handling)

- 何らかの異常が発生したときの対応方法を記述すること
- 入出力には例外が付き物
  - そんな名前のファイルはありません！
  - 相手のコンピュータに接続できません！

# ファイル入出力： ファイルをコピーするプログラム

- File オブジェクトから FileInputStream を作る

```
public class FileCopy {  
  
    public static void main(String[] args) {  
        File f1 = new File("test.txt"); // ここから  
        File f2 = new File("copy.txt"); // ここにコピーする  
  
        FileInputStream is = new FileInputStream(f1);  
    }  
}
```

処理されない例外の型 FileNotFoundException

2 個のクイック・フィックスが使用可能です:

- スロー宣言の追加
- try/catch で囲む

# try ... catch 文

- FileInputStream を作るときには例外が発生する可能性がある

```
try {  
    FileInputStream is = new FileInputStream(f1);  
} catch (FileNotFoundException e) {  
    // TODO 自動生成された catch ブロック  
    e.printStackTrace();  
}
```

## 【書式】

```
try { 例外が起きるかもしれない処理 }  
catch(例外クラスの変数){ 例外時の処理 }
```

# 同様に FileOutputStream も作成

```
try {
    FileInputStream is = new FileInputStream(f1);
    Scanner scanner = new Scanner(is);
    FileOutputStream os = new FileOutputStream(f2);
    PrintWriter writer = new PrintWriter(os);

    while(scanner.hasNext()){
        writer.println(scanner.nextLine());
    }
    writer.close();
} catch (FileNotFoundException e) {
```

InputStream や OutputStream には文字が書き込めない  
→ Scanner や PrintWriter を

# 読み込みと書き込み

```
while(scanner.hasNext()){  
    writer.println(scanner.nextLine());  
}  
writer.close();
```

- **注意: 実際に書き込まれるのは flush したとき**
  - flush 関数 = たまっているデータを送る
  - close 関数 = flush してからストリームを閉じる
  - new PrintWriter(os, true) と書いて、自動的に flush してくれる PrintWriter を作っても良い

# 練習1

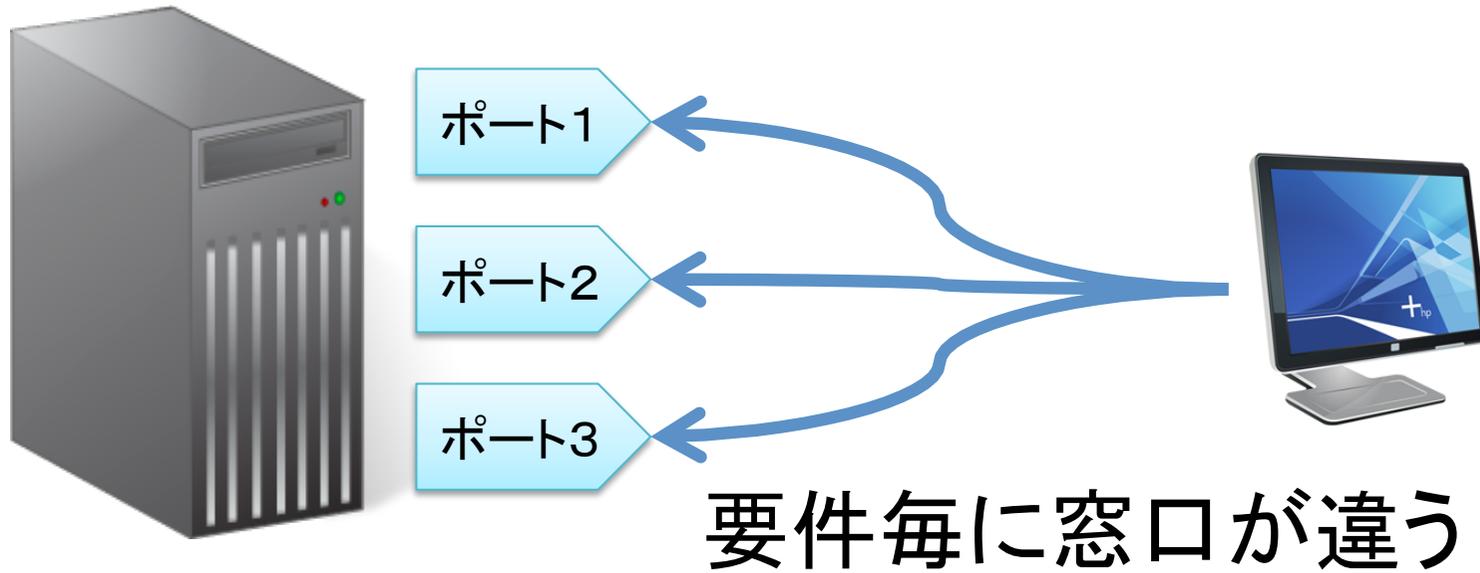
1. コピー元のファイル test.txt を作成
1. FileCopy を書いて実行
  1. コピー元のファイルが存在する場合
  2. 存在しないファイル名 test2.txt を指定した場合
2. catch の中身を変更してもう1度実行

```
} catch (FileNotFoundException e) {  
    // e.printStackTrace();  
    System.out.println("コピー元のファイルが存在しませんでした");  
}
```

# 通信

- プログラムの書き方はファイルを扱う場合とほとんど同じ
- File の代わりに Socket からストリームを得る
  - Socket の場合は通信相手を指定
  - 「アドレス」と「ポート」
- 発生する例外が異なる

# アドレスとポート



アドレス = コンピュータの住所  
ポート = 担当窓口

# アドレスとポート(つづき)

- IPアドレス
  - 4つの数字の組 (例 113.152.70.115)
- ポート
  - 0～65535の数字
  - 0～1023までは使い方が決まっている
    - メール: 25, 110, 143 等
    - ウェブ: 80, 443 等
    - 自分のプログラムではたとえば 10000 などを使う

# データを送信するプログラム

```
public class Sender {  
  
    public static void main(String[] args) {  
        try {  
            Socket socket = new Socket("localhost", 10000);  
            OutputStream os = socket.getOutputStream();  
            PrintWriter writer = new PrintWriter(os);  
            writer.println("Hello");  
            writer.close();  
        } catch (UnknownHostException e) {  
            // TODO 自動生成された catch ブロック  
            e.printStackTrace();  
        } catch (IOException e) {  
            // TODO 自動生成された catch ブロック  
            e.printStackTrace();  
        }  
    }  
}
```

# データを受信するプログラム

```
public class Receiver {  
    public static void main(String[] args) {  
        try {  
            ServerSocket serverSocket = new ServerSocket(10000);  
            System.out.println("接続を待っています");  
            Socket socket = serverSocket.accept();  
            System.out.println("接続されました");  
            InputStream is = socket.getInputStream();  
            Scanner scanner = new Scanner(is);  
            System.out.println("受信:" + scanner.nextLine());  
        } catch (IOException e) {  
            // TODO 自動生成された catch ブロック  
            e.printStackTrace();  
        }  
    }  
}
```

# 練習2

1. Sender と Receiver を作って実行してみよう
  - 先に Receiver を実行して接続を待機している状態にしてから Sender を実行
2. Receiver を実行していない状態で Sender を実行したらどうなるか試してみよう
3. “localhost” の部分を変えて、隣の人プログラムに接続してみよう
  - アドレスの調べ方は次のスライドを参照

# IPアドレスの調べ方

- 左上の🍏 > システム環境設定

The image shows a screenshot of the macOS System Preferences window, specifically the Network pane. The window is titled "システム環境設定" (System Preferences) and "ネットワーク" (Network). The left sidebar shows various system settings categories: パーソナル (Personal), ハードウェア (Hardware), インターネットとワイヤレス (Internet & Wireless), システム (System), and その他 (Other). The main pane shows the Network environment set to "自動" (Automatic). Underneath, there are three network interfaces: Ethernet (connected), AirMac (connected), and FireWire (not connected). The Ethernet interface is selected, and its status is "接続" (Connected). The status message says "Ethernet は現在使用中で、IP アドレス 133.30.244.171 が設定されています。" (Ethernet is currently in use, and IP address 133.30.244.171 is configured). The IPv4 configuration is set to "DHCP サーバを使用" (Use DHCP server). The IP address is 133.30.244.171, the subnet mask is 255.255.255.0, the router is 133.30.244.254, the DNS server is 133.30.244.1, and the search domain is cla.kobe-u.ac.jp. At the bottom, there are buttons for "アシスタント..." (Assistant...), "元に戻す" (Reset), and "適用" (Apply).

# 課題6

- 以下のような2つのクラスを作成
  - Client
    - 何か文字列を1回送る
    - 受け取ったデータを標準出力に表示する
  - Server
    - 受け取った文字列に何か加工をして送り返す  
(前に「受信:」と付ける、1回受信したら3回返す etc.)
- 2つのjavaファイルを圧縮したzipを提出

# 課題6 (option)

- 以下のような2つのクラスを作成
  - FileClient
    - ファイル名を送信する
    - 相手から送られてきたデータを System.out に出力する
  - FileServer
    - ファイル名を受信する
    - ファイルが存在→ファイルの内容を相手に送信
    - ファイルが存在しない→ “Not found” と相手に送信
- 2つのjavaファイルを圧縮した zip を提出