ネットワークプログラミング
ネットワークプログラミング演習
21001教室
第4回
2013/10/7

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%2F2013netpro

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

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@gmail.com>

2限副手紹介

- 鉄谷研究室M2
  - 多田　真之　tada先輩 2限
  - 新井　駿　arashun先輩　2限

成績

- 出席
- 毎回課題
- 中間試験
- 最終試験＋最終課題

- ★演習は演習最終発表会を加味

講義内容

第1回	Java理解度チェック
第2回	TCP/IPの復習
第3回	TCPサーバ
第4回	TCPクライアント/サーバ通信　チャットプログラム
第5回	UDP通信
第6回	**中間試験（持ち込み不可　紙提出）**
第7回	スレッド基礎 サーバのスレッド化 マルチスレッド
第8回	デザインパターン ファクトリメソッド　シングルトン
第9回	ノンブロッキングI/O　　最終課題に向けた目標設定
第10回	マルチスレッド　スレッドプール　Twitter4J
第11回	Twitter4J, Webクライアント
第12回	WEBサーバ,プロジェクト設計
第13回	プロジェクト実装
第14回	予備
第15回	**学力考査（持ち込み可　プログラミング提出）**

授業予定日日程

- http://www.soe.dendai.ac.jp/kyomu/portal/2013_schedule_t.pdf
- (2)9/16 敬老の日【授業実施日】
- (3)9/23 秋分の日【授業実施日】
- (4)9/30
- (5)10/7
- **(6)10/14 体育の日【授業実施日】 中間試験**
- (7)10/21
- (8)10/28
- ~~休み 11/4 文化の日~~
- (9)11/11
- (10)11/18
- (11)11/25
- (12)12/2
- (13)12/9
- (14)12/16
- ~~休み 岩井出張休講 12/23~~
- ~~休み1/13 成人の日~~
- **(15)1/20 学力考査**
- **※授業予定日に休講が有る場合は連絡します。**

概要

- クライアント／サーバモデル、TCP/IPネットワークのアプリケーションプログラミングインタフェースの基本および、ネットワークアプリケーションを効率的に動作させるためのマルチスレッドプログラミングを講義する。この基本の後、チャット等の対話型アプリケーション、Twitter4J等のアプリケーション開発の実例を講義する。

ゴール

- 通信ネットワークを利用したアプリケーションソフトウェアを、TCP/IP を意識したレベルで作成できる力を養成することを目標とする。

参考書

- 購入の必要はありません。
  - TCP/IPソケットプログラミング JAVA編
  - –ISBN4-274-06520-0
  - –オーム社
  - •[改訂第２版]JAVA言語プログラミングレッスン上
  - –ISBN4-7973-3211-5
  - –SoftBankCreative
  - •[改訂第２版]JAVA言語プログラミングレッスン下
  - –ISBN4-7973-3212-3
  - –SoftBankCreative

持ち物

- Laptop pc
  –Eclipse環境が整っていること
- PDF アクロバットリーダ
- LANでネットワークに接続できること。
- Macでもよい

日本語版eclipse

- http://mergedoc.sourceforge.jp/

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

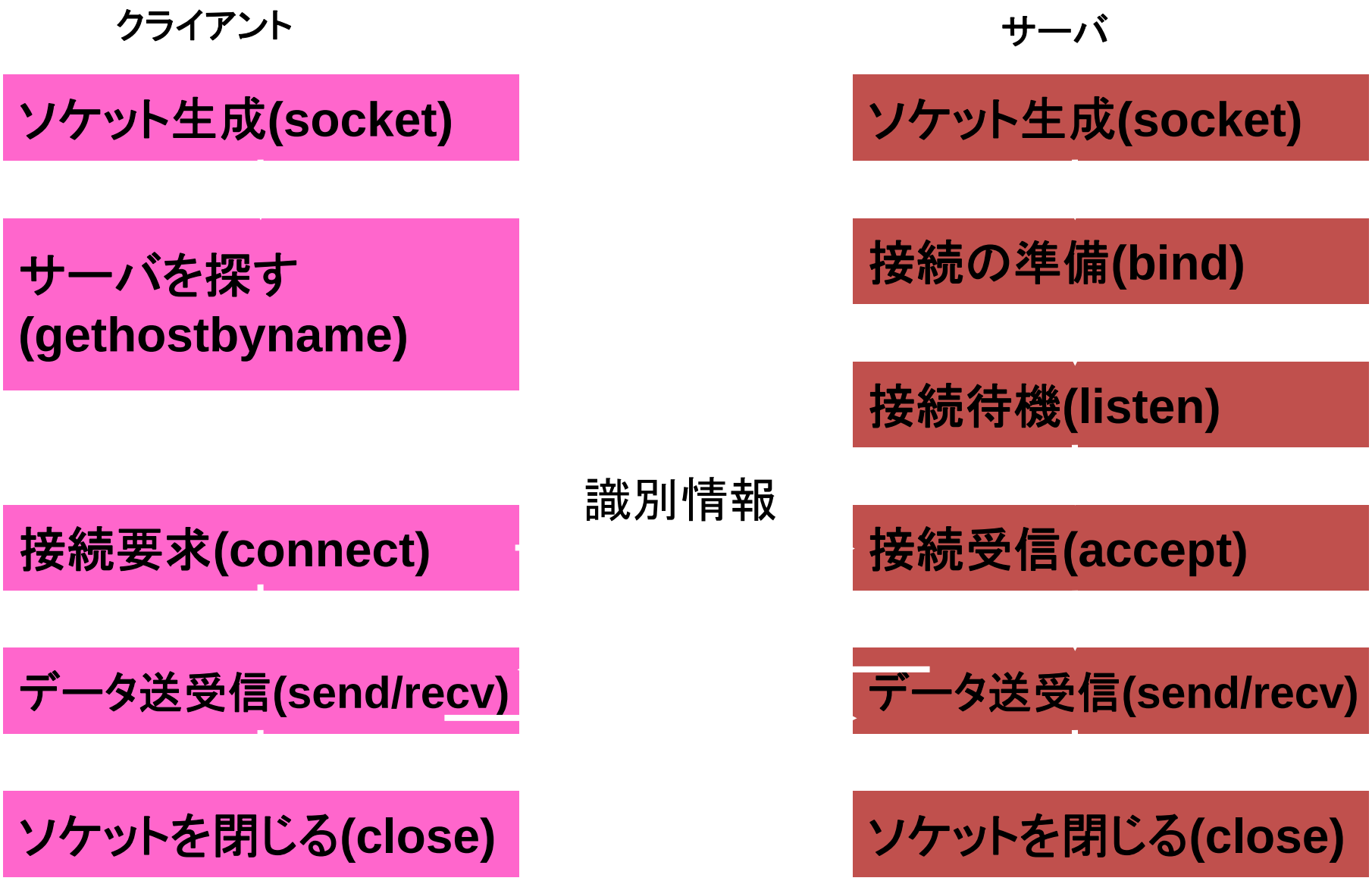
クライアントプログラム

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



識別情報

- 正しくデータを受け渡しするために
  通信する相手を識別する

IPアドレス

コンピュータを識別

コンピュータのアドレス

ポート番号

プログラムを識別

プログラムの識別番号

クライアントプログラム（Java）

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

public class Client{
   public static void main( String[] args ){

      try{
         //ソケットを作成
         String host="localhost";
         Socket socket = new Socket( host, 10000 );

         //入力ストリームを作成
         DataInputStream is =  new DataInputStream (
                  new BufferedInputStream(
                     socket.getInputStream()));
```

```
         //サーバ側から送信された文字列を受信
         byte[] buff = new byte[1024];
         int a = is.read(buff);
         System.out.write(buff, 0, a);

         //ストリーム，ソケットをクローズ
         is.close();
         socket.close();

      }catch(Exception e){
                              System.out.println(e.getMessage());
         e.printStackTrace();
      }
   }
}
```

サーバプログラム（Java）

```
//Server.java

import java.net.*;
import java.lang.*;
import java.io.*;

   public class Server{
      public static void main( String[] args ){

         try{
            //ソケットを作成
            ServerSocket svSocket = new ServerSocket(10000);
            //クライアントからのコネクション要求受付
            Socket cliSocket = svSocket.accept();

            //出力ストリームを作成
            DataOutputStream os = new DataOutputStream(
                  new BufferedOutputStream(
                     cliSocket.getOutputStream()));

            //文字列を送信
            String s = new String("Hello World!!¥n");
            byte[] b = s.getBytes();
            os.write(b, 0, s.length());
```

```
//ストリーム，ソケットをクローズ
            os.close();
            cliSocket.close();
            svSocket.close();

         }catch( Exception e ){
               System.out.println(e.getMessage());
            e.printStackTrace();
         }
      }
}
```

サーバプログラム（Java）

ソケット作成，コネクション要求受付待機

```
ServerSocket svSocket =
  newServerSocket(10000);
Socket cliSocket = svSocket.accept();
```

出力ストリーム作成

```
DataOutputStream os =
  new DataOutputStream(
    new BufferedOutputStream(
      cliSocket.getOutputStream());
```

クライアントプログラム（Java）

ソケットを作成

```
Socket socket = new Socket(
            “hoge.com”, 10000 );
```

入力ストリームを作成

```
DataInputStream is =
  new DataInputStream (
    new BufferedInputStream(
      socket.getInputStream()));
```

クライアントプログラム（Java）

サーバ側から送信された文字列を受信

```
n = is.read(buff);
System.out.write(buff, 0, n);
```

ストリーム，ソケットをクローズ

```
is.close();
socket.close();
```

サーバプログラム（Java）

ソケット作成，コネクション要求受付待機

```
ServerSocket svSocket =
  newServerSocket(10000);
Socket cliSocket = svSocket.accept();
```

出力ストリーム作成

```
DataOutputStream os =
  new DataOutputStream(
    new BufferedOutputStream(
      cliSocket.getOutputStream());
```

サーバプログラム（Java）

文字列を送信

```
String s = new String("Hello World!!¥n");
byte[] b = s.getBytes();
os.write(b, 0, s.length());
```

ストリーム，ソケットをクローズ

```
os.close();
cliSocket.close();
svSocket.close();
```

まずは動かす。

- ChatServer
- ChatClient

InetAddress

- http://docs.oracle.com/javase/jp/6/api/java/net/InetAddress.html
- **IPアドレスを扱うクラス**
- **java.net**
  クラス InetAddress
- java.lang.Object **java.net.InetAddress すべての実装されたインタフェース:**Serializable**直系の既知のサブクラス:**Inet4Address, Inet6Address

InetAddress

- String**getHostName**()
  この IP アドレスに対応するホスト名を取得します。
- static InetAddress**getByAddress**(String host, byte[] addr)
  指定されたホスト名および IP アドレスに基づいて InetAddress を作成します。

- static InetAddress**getLocalHost**()
  ローカルホストを返します。

InetAddress

- static InetAddress**getByName**(String host)
  指定されたホスト名を持つホストの IP アドレスを取得します。

- boolean**isReachable**(int timeout)
  そのアドレスに到達可能かどうかをテストします

- String**toString**()
  この IP アドレスを String に変換します。

メソッドの引数

```
戻り値　メソッド名(型　変数名1,
                  型　変数名2,
                  型　変数名3,
                  型　変数名4
                    ..............)　{

    メソッドの本体


}
```

Javaのメソッドの引数

戻り値　メソッド名(型　変数名1,
型　変数名2,
型　変数名3,
型　変数名4
.....) {

メソッドの本体

}

クラス（既に定められたものでも、
自分で定めたものでも) の名前でもよい

メソッド呼び出し

本来は、

```
g.drawString(XXXXXXXXXXXXX);
```

のように、

オブジェクト.メソッド名(引数...);

と書く。

メソッド呼び出し(2)

しかし、自分で定義したクラスの中の
メソッドを呼び出すときは、

　　オブジェクト.

なしに、

　　メソッド名(引数...);

でよい。

例:

　　drawBar(XXXXXXXXXX);

methodとクラス

- Heikin.java と Kamoku.java

- Heikin と Kamoku クラスを作る
  - public class Heikin
  - class Kamoku
- Heikin クラス
  - Kamokuクラスのインスタンスとして、englishとmath を作る
  - english の name に "英語" を設定する
  - english の score に 80 を設定する
  - math　も english と同様に (name→数学, score→70）
  - 英語と数学のscore を読み出して、平均値を表示する
- Kamoku クラス
  - String name
  - setScore というメソッドを定義する。score に値を設定する。
  - getScore というメソッドを定義する。scoreを返す。

定数の宣言

C++/C では、#define 文を使用した。
(例)

```
#define   WIDTH     80
```

Javaでは、final static で修飾する。
(例)

```
public final static  int  WIDTH  = 80;
public final static  String  school = "dendai";
```

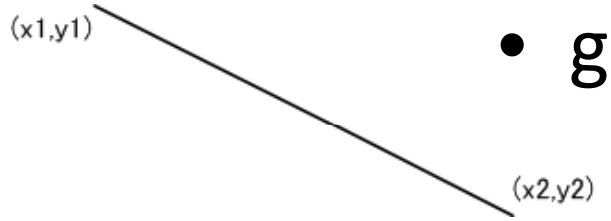
Graphics

Graphics というクラスには、
　drawString, drawCircle 等の
メソッドが定義されている。

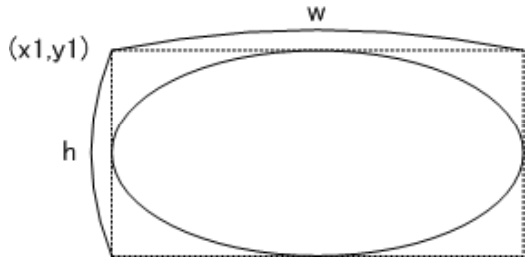
Graphics クラスである g という
オブジェクトに対して、
　g.drawString、
　g.drawCircle
という形でメソッドを呼び出せる。

Graphicsのmethod

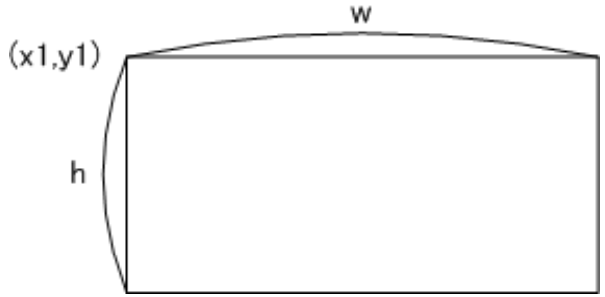
- g.drawLine(x1,y1,x2,y2);



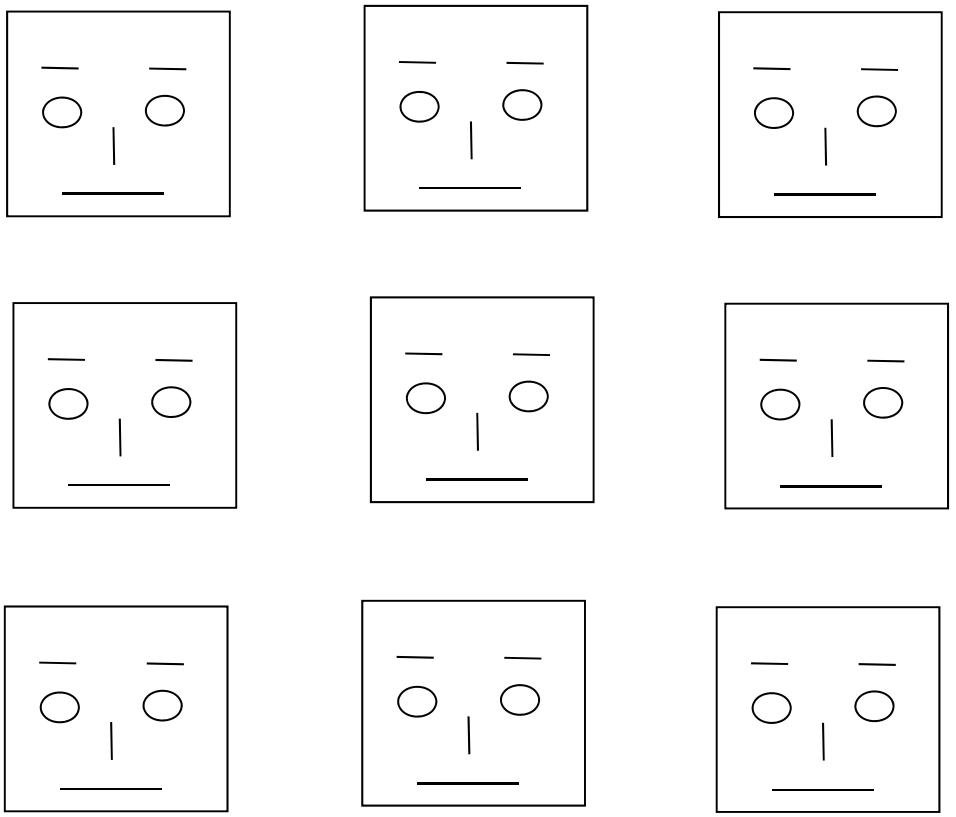
- g.drawOval(x,y,w,h);



- g.drawRect(x,y,w,h);



前回課題faceを沢山つくってみよう
Face.javaを改造してFaces.java



Face ヒント1

```
void   drawFace(Graphics g,
                int  xStart,
                int  yStart)  {
```

　左隅の座標を (xStart, yStart) として、一つの顔を描くメソッドを記述する。

```
}
```

Faceヒント2

paint メソッドの中には、

```
   for(int i = 0; i < 3; i++) {
      for(int j=0; j < 3; j++) {
         drawFace(g, 20 + 80 *i,
                              20 + 80 *j);
      }
 }
```

80 という数字は仮。顔の大きさを考えて計算する。

しかし、数字決め打ちは避けたい

```
  for(int i = 0; i < 3; i++) {
     for(int j=0; j < 3; j++) {
        drawFace(g, 20 + step *i,
                          20 + step *j);
     }
}
```

眉毛や鼻の形を自由に変えたい

```
void   drawFace(Graphics g, int  xStart,  int  yStart)  {
      drawFrame(g, xStart, yStart);
      drawEyeBrow(g, xStart, yStart);
      drawEye(g, xStart, yStart);
      drawNose(g, xStart, yStart);
      drawMouth(g, xStart, yStart);
}

void drawFrame(Graphics g, int xStart, int yStart) {
   記述
}
void drawEyeBrow(Graphics g, int xStart, int yStart) {
   記述
}
void drawEye(Graphics g, int xStart, int yStart) {
   記述
}
void drawNose(Graphics g, int xStart, int yStart) {
   記述
}
void drawMouth(Graphics g, int xStart, int yStart) {
   記述
}
```

さらに内部で
メソッドに分ける。

次週以降

MovingBall
Thread,Runnable
RandSwitch
配列

配列の宣言

- 型　配列名[] = new 型[n];
  int tensu[] = new int[100];
  型[]　配列名 = new 型[n];
  int[] tensu = new int[100];

0 ～ n-1 の n個の配列ができる。

配列の添字は0から始まる
配列 xData の大きさが3のとき、使えるのは、xData[0]、xData[1] 、xData[2]。xData[3]は使えない。

配列の宣言

・int mathScore[] = new int[5];
と宣言すると、

```
  mathScore[3] =  82;
for(int i = 0; i < 5; i++) {
    mathScore[i] = 10;
}
```

のような代入等が可能となる。

配列の長さ

- 配列名.length
- 例えば、xData の長さは、xData.length で求められる。

配列の初期化

配列の型[]　配列 = {要素, 要素, 要素};

例

```
int[]   xData = {90, 85, 65};
```