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4回
2013/10/2

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

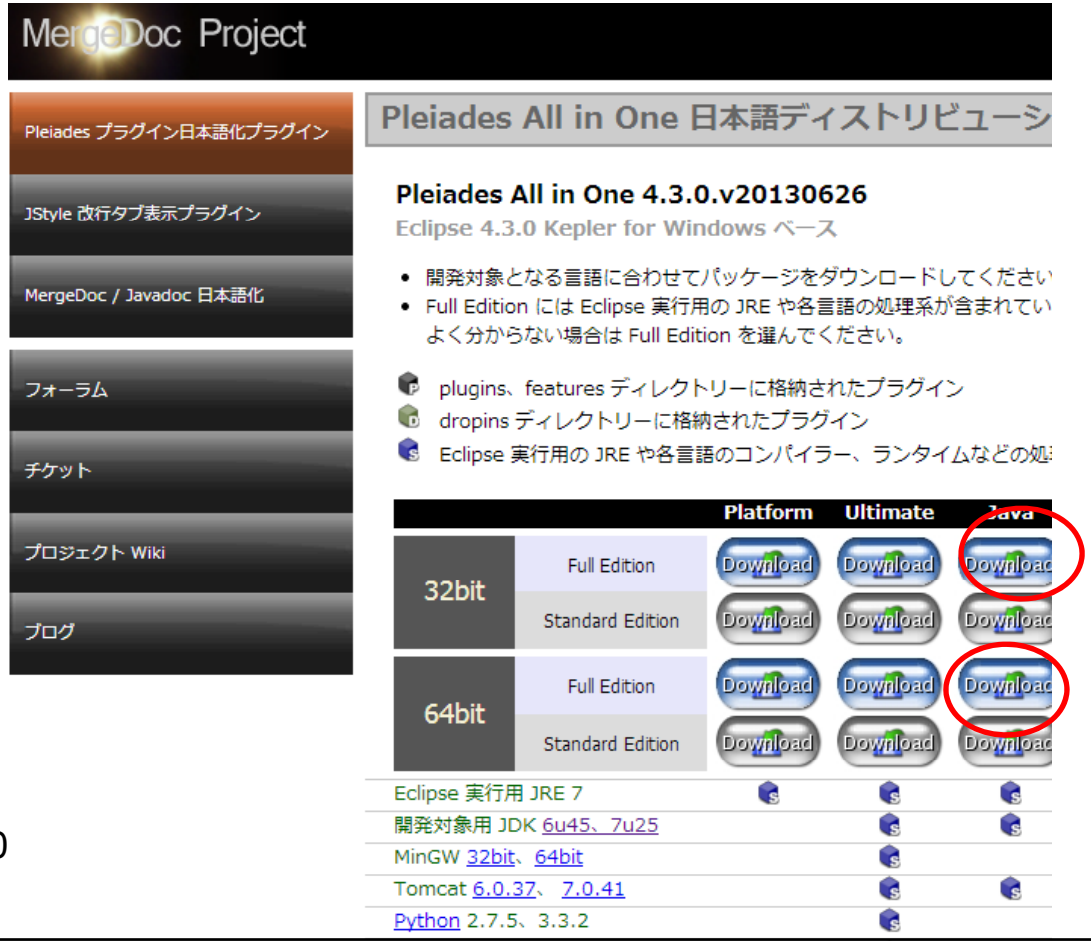
- 詳しくはFacebook　岩井将行
- Twitter @masaiwai

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>

日本語版eclipse

- http://mergedoc.sourceforge.jp/



または、かつ

または ||

条件式1 || 条件式2

かつ &&

条件式1 && 条件式2

例

((p > q) && (r < s)) || ((p < q) && (r < s))

for文

```
for ( 初期化; 条件式; 次の一歩 ) {
        繰り返す処理
}
```

```
for (int i = 0; i < 3; i++) {
   System.out.println(i);
}
```

により、

0

1

2

が表示される。

前回の演習

```
   *
  ***
 *****
*******
```

表示。
System.out.print("　"); 　"の中はスペース
を用いる。

改行指定

- System.out.print(“¥n”);
- System.out.println();

変数の有効範囲(スコープ)

```
for (int i = 0; i < 3; i++) {
    System.out.println(i);
}
System.out.println("i = " + i):
```

← iは有効範囲外なので
コンパイルエラー。

解決策

```
int i;
for (i = 0; i < 3; i++) {
    System.out.println(i);
}
System.out.println("i = " + i):
```

while 文

```
while (条件式) {
   繰り返す処理
}
```

null (ナル、ヌル)

```
while (line != null) {
```

null　入力の終わりに達したときの特殊なオブジェクト

LoopTest問題

1 から 100 までの整数を足し合わせる

(1) その1:　for文を使う

(2) その2:　while文を使う

LoopTest.java

Method

- Calc3MethodTest.java

定義する場所

クラス

```
返り値  メソッド1 {
  XXXXXXXXXX
}
```

```
返り値  メソッド2 {
  XXXXXXXXXX
}
```

```
返り値  メソッド3 {
  XXXXXXXXXX
}
```

```
返り値  メソッド4 {
  XXXXXXXXXX
}
```

いくつでも
作ってよい。

public ?

クラス

```
public 返り値  メソッド1 {
  XXXXXXXXXX
}
```

```
public  返り値  メソッド2 {
  XXXXXXXXXX
}
```

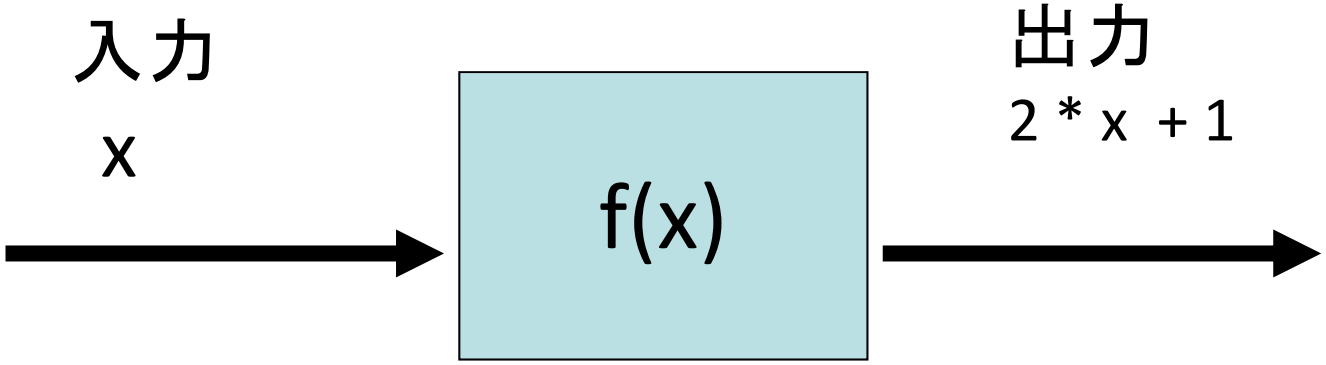
```
返り値  メソッド3 {
  XXXXXXXXXX
}
```

```
返り値  メソッド4 {
  XXXXXXXXXX
}
```

関数

- 関数

f(x) = 2 * x + 1

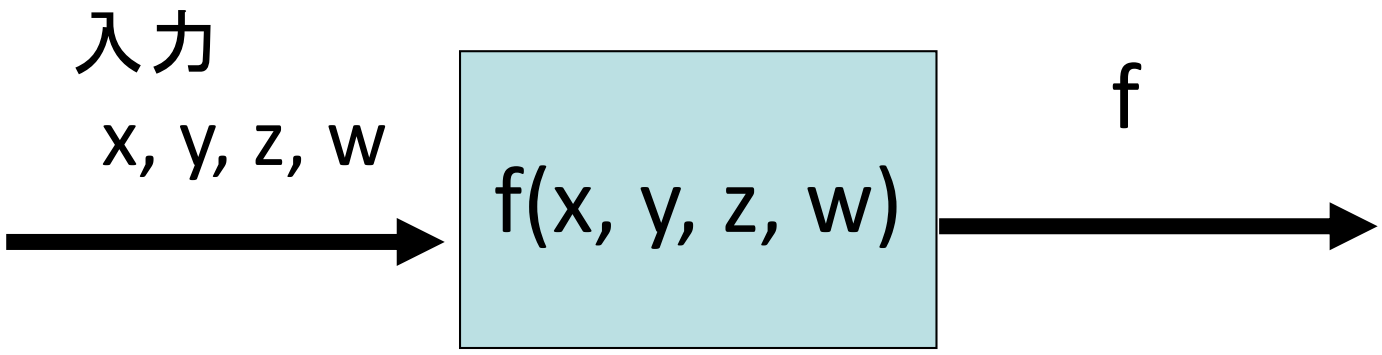


関数の定義部分

しかし、関数の入力はたくさん あってよい。

- 関数

$$f(x, y, z, w) = 2 * x + y - z$$



関数の定義部分

メソッドで引数がたくさんあるとき

```
int   calcComplex(int x, int y, int z,
                  float w) {
      if ( x > y ) {
         return z;
      } else {
         return (int)w;
}
```

メソッド分け

- 合成関数

  f(x) = 2 * x + 1

  h(x) = 3 * (2 * x + 1) + 5

のとき、h(x) = (g ○ f) (x)

```
int  h(int x) {
  return 3 * (2 * x + 1) + 5;
}
```

```
int  h(int x) {
  return 3 * g(x) + 5;
}

int  g(int x) {
  return 2 * x + 1;
}
```

Javaプログラミングも同じ。メソッドとして
独立させた方がよいかどうか、よく考える。

メソッドの形式

公開するか否か

クラスメソッドとする

戻り値の型

```
public static int メソッド名(引数宣言) {

    メソッドの中身

    return (戻り値);
}
```

void

関数によっては、返り値がいらないものもある。そのときには、返り値なし (void) を指定する。
前回作成した、drawBar に返り値は必要なかった。

引数がない場合もある。

型

int　　整数
float　浮動小数点数（実数)
char　文字型

等

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(XXXXXXXXXXX);

クラスとインスタンス

クラスは、物の設計図。
　中に変数やメソッドが定義される。

オブジェクトは、クラス定義に基づく実際の物。プログラム上は、変数。

例：Automobile というクラスを定義する。
　Automobileクラスで、volkesWagen, audi, volvo というオブジェクトを作る。

メソッド

車というクラスを定義する。メソッドとして、

　　乗る、止める

を用意する。

ポルシェというオブジェクトを車というクラスで生成すると、

　ポルシェ. 乗る、

　ポルシェ. 止める

というメソッドが使える。

インスタンスとメソッドの練習

- 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。

Graphicsクラス

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。

定数の宣言

C++/C では、#define 文を使用した。
(例)

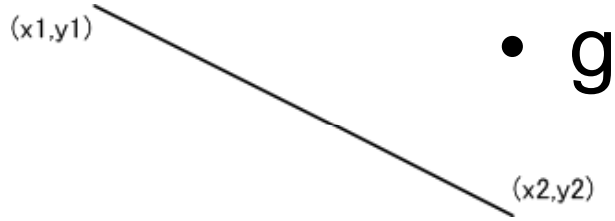
```
#define   WIDTH     80
```

Javaでは、final static で修飾する。
(例)

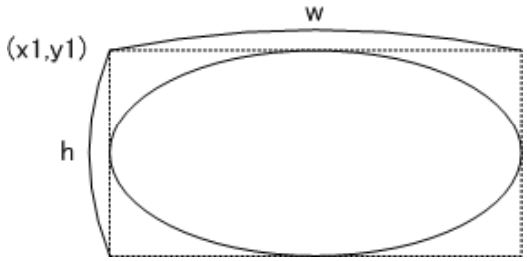
```
public final static  int  WIDTH  = 80;
public final static  String  school = "dendai";
```

Graphics method

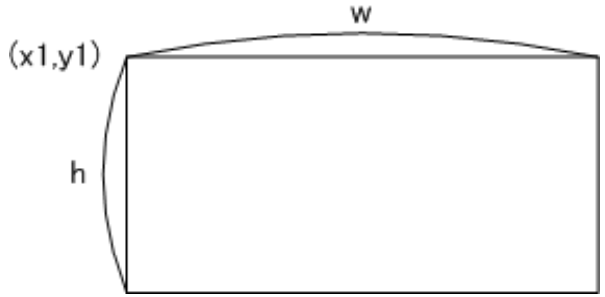
- g.drawLine(x1,y1,x2,y2);



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

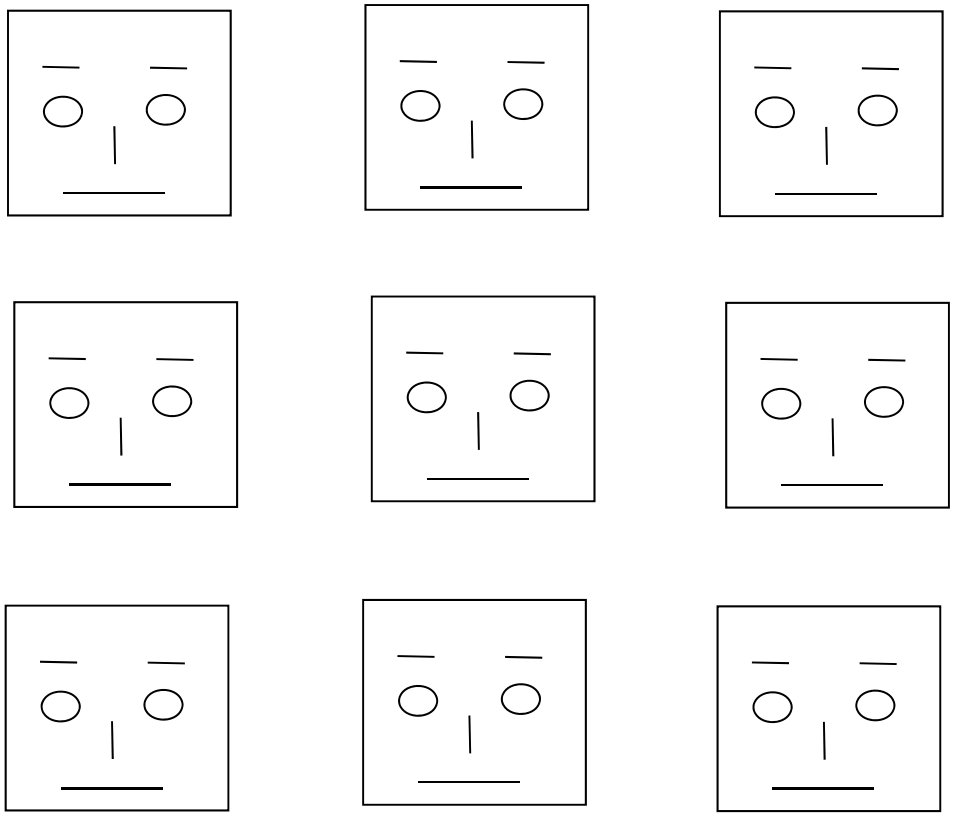


- g.drawRect(x,y,w,h);



課題Faces.java

- FaceSample1.javaを改良して3x3の顔を作成せよ



ヒント1

```
void   makeFace(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。


}
```

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++) {
          makeFace(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++) {
     makeFace(g, 20 + step *i, 20 + step *j);
  }
}
```

ヒント3:眉毛や鼻の形を自由に変えたい

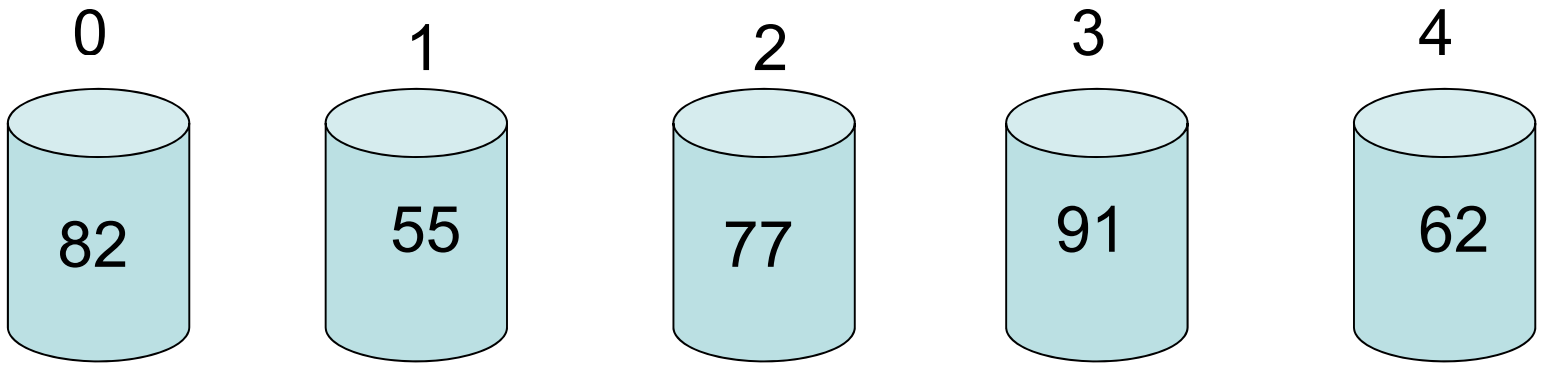
```
void   makeFace(Graphics g, int  xStart,  int  yStart)  {
      makeFrame(g, xStart, yStart);
      makeEyes(g, xStart, yStart);
      makeNose(g, xStart, yStart);
      makeMouth(g, xStart, yStart);
}

void makeFrame(Graphics g, int xStart, int yStart) {
   記述
}
void makeEyeBrow(Graphics g, int xStart, int yStart) {
   記述
}
void makeEye(Graphics g, int xStart, int yStart) {
   記述
}
void makeNose(Graphics g, int xStart, int yStart) {
   記述
}
void makeMouth(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

配列

int xData;
という宣言をすると、xDataという名前の整数型の変数が使える。
５人の学生がいて、各学生のコンピュータ基礎の点数を保存したいときにはどうしたらよいか？
int mathScore;
では 1つの変数。→ 5人分.



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};

intの配列

- mathScore という大きさ5のintの配列をつくり、82, 55, 77, 91, 62 の初期値を入れる
- 全てを　＋５する
- 最高点と平均点を表示する
  - for文を使う
  - TwoArray.java

Quick Summary

クラスは、物の設計図。
　中に変数やメソッドが定義される。

オブジェクトは、クラス定義に基づく実際の物。プログラム上は、変数。

例：Automobile というクラスを定義する。
　Automobileクラスで、volkesWagen, audi, volvo というオブジェクトを作る。

Quick Summary (2)

車というクラスを定義する。メソッドとして、

　　乗る、止める

を用意する。

ポルシェというオブジェクトを車というクラスで生成すると、

　ポルシェ. 乗る、

　ポルシェ. 止める

というメソッドが使える。