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
for while 二重loop

第3回
2013/9/25

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

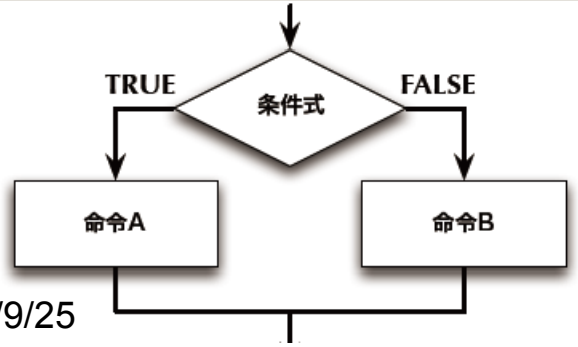
- 詳しくはFacebook　岩井将行
- Twitter @masaiwai

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@gmail.com>

条件分岐 (if文)

```
if(rain==1) {
   goByBus();
}
else {
   goByBicycle();
}
```



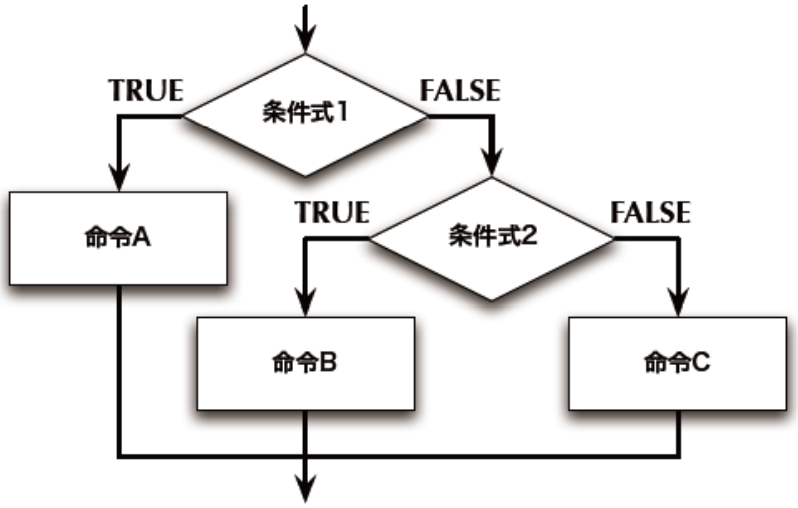

```
if(条件式) {
   条件式が成り立つ
   場合実行する処理
}
else {
   条件式が成り立たない
   場合実行する処理
}
```

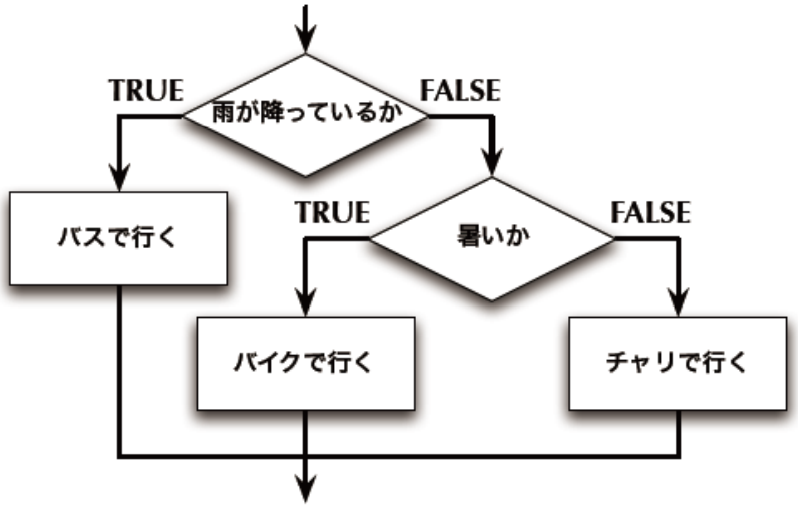
変数の比較

左辺 >= 右辺	≧	左辺の値が右辺の値以上
左辺 > 右辺	>	左辺の値が右辺の値より大きい
左辺 <= 右辺	≦	左辺の値が右辺の値以下
左辺 < 右辺	<	左辺の値が右辺の値より小さい
左辺 == 右辺	=	左辺の値と右辺の値が等しい
左辺 != 右辺	≠	左辺の値と右辺の値が等しくない

多重分岐

- 3つ以上分岐するには？
  - 雨だったらバス、雨でなくても暑かったらバイク、それ以外ならチャリ





多重分岐 (else if)

```
if(rain==1) {
  goByBus();
}
else if(hot==1) {
  goByBike();
}
else {
  goByBicycle();
}
```

```
if(条件式1) {
 条件式1が成り立つ場合
 実行する処理
}
else if (条件式2) {
 条件式1が成り立たないが
 条件式2が成り立つ場合
 実行する処理
}
else {
 条件式1も条件式2も
 成り立たない場合実行する処理
}
```

if文

```
if (条件式) {
    条件が成り立つときの処理
}
```

条件式は boolean型

その値は、trueかfalse

比較演算子

例　if (flag == true)　flagはboolean型変数

```
if ( p == 50)
if ( p != 50)
if (p >= 50)
if ( p < 50)
```

if … else

```
if (条件) {
        条件が成り立つときの処理
    }
else {
        条件が成り立たないときの処理
}
```

booleanの演算(かつ)

かつ　&&

条件式1 &&　条件式2

true&&true -> true

true&&false->false

false&&true->false

false&&false->false

booleanの演算(または)

または ||

条件式1 || 条件式2

true||true -> true

true || false->true

false || true-> true

false || false->false

条件式

```
if(          )    //iが3なら
if(          )    //iが10以上なら
if(          )    //iが10未満なら
if(          )    //iが偶数なら
if(          )    //iが奇数なら
if(                ) // iが6以上12以下
if(                ) // iが6未満12より大
```

問題１

- 宿題を改良して、平均が80点以上なら「よくできました」 ８０点以下なら「がんばりましょう」と表示するプログラムを作ろう

if … elseの連鎖

```
if (条件式1) {
        条件1が成り立つときの処理
    }
else  if  (条件式2) {
        条件1が成り立たず、
        条件2が成り立つときの処理
}
else  {
       条件1も、条件2も成り立たないときの処理
}
```

問題２

- 平均が１００点以上なら「点数が間違っています」、平均が80点以上なら「よくできました」８０点以下なら「がんばりましょう」 と表示するプログラムを作ろう

- else if を使う

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))

問題３

- 平均が１００点以上または ０点未満なら「点数が間違っています」、平均が80点以上なら「よくできました」 ８０点以下なら「がんばりましょう」 と表示するプログラムを作ろう

- || をつかう

問題3.5

１～６を入力する
１がでたら、「一の目です」
２がでたら、「二の目です」
３がでたら、「三の目です」
４以上がでたら、「それ以外の目です」
と表示する

Switch 文を使う

問題４

さいころプログラムを作ろう

１がでたら、「一の目です」

２がでたら、「二の目です」

３がでたら、「三の目です」

４以上がでたら、「それ以外の目です」

と表示する

- Math.random()　をつかう
  - 0.0以上1.0未満の値の中からランダムな値を算出します。

問題５

- 引数に入れた2つの数字を加算するプログラム
  - Calc2
- ２つの引数以外を入れているとエラーメッセージがでるようなプログラムをつくる

```
public class Hikisu2 {
    public static void main(String[] args) {
        System.out.println(“1番目の引数は” + args[0] + "です。");
        System.out.println(“1番目の引数は” + args[1] + "です。");
        }
    }
}
```

四則演算

- 四則演算と変数
  - 代入
    - i = 10 / 2 + 1;
    - i = i * i;
  - 比較
    - i / 2 > 5

足す (+)	+
引く (−)	-
掛ける (×)	*
割る (÷)	/
剰余 (…)	%

- 奇数かどうかを求めるには
  - 2で割った余りが1かどうか?
    - j % 2 == 1

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。

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

問題

- While 文を使って表示する

0
1
2
…
1000

問題

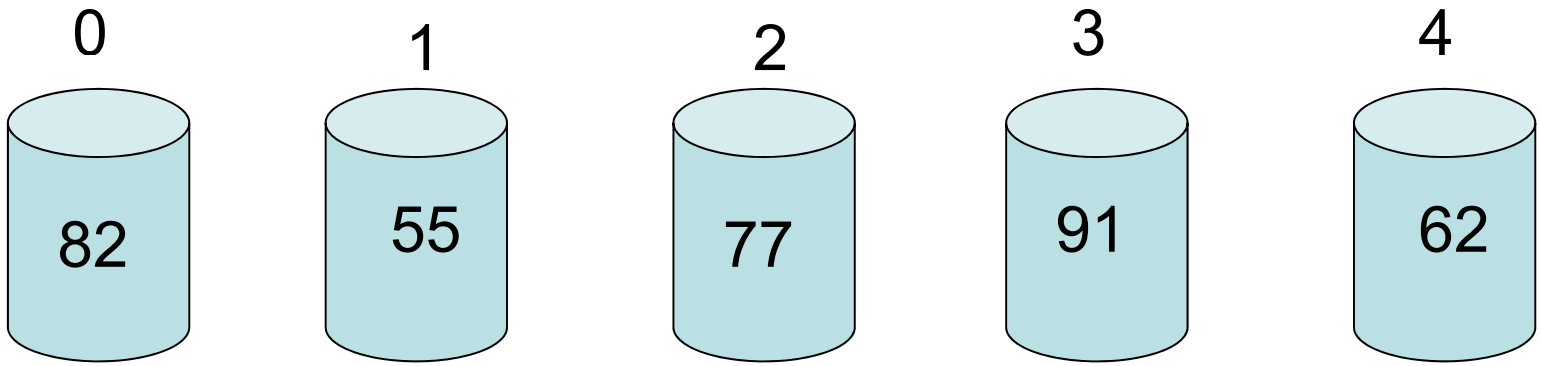
1 から 100 までの整数を足し合わせる

(1) その1:　for文を使う

(2) その2:　while文を使う

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};

問題

- 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

先週の答え合わせ

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

先週の答え合わせ２

- 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

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。

Quick Summary (3)

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
}
```

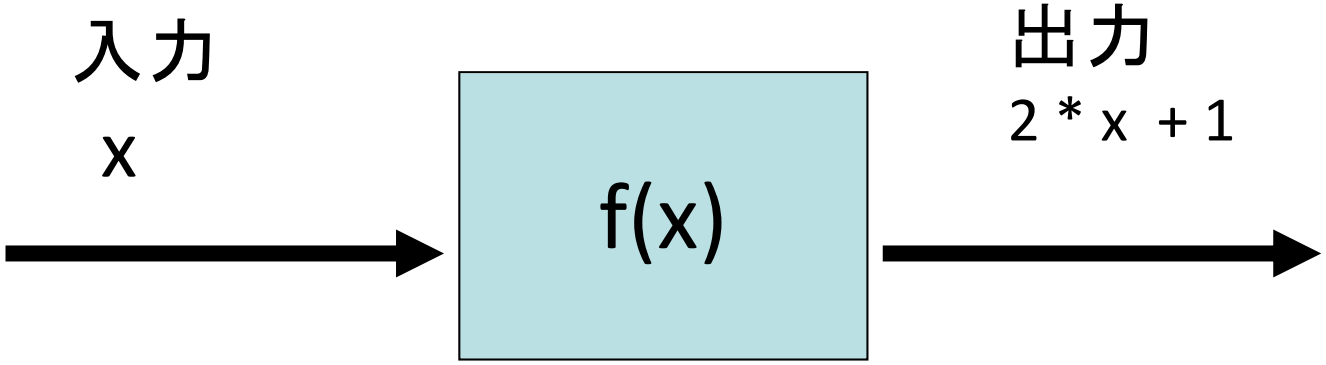
```
返り値  メソッド3 {
  XXXXXXXXXX
}
```

```
返り値  メソッド4 {
  XXXXXXXXXX
}
```

関数

- 関数

f(x) = 2 * x + 1

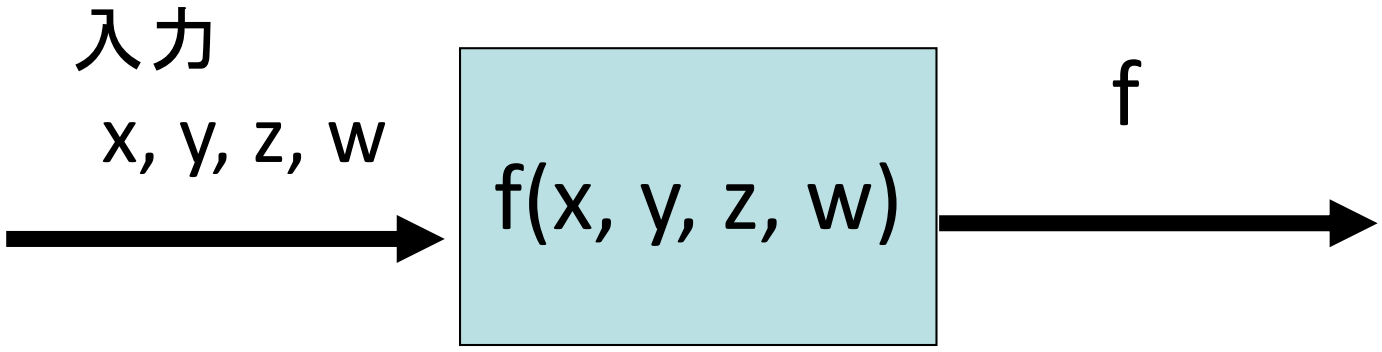


関数の定義部分

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
+ ∟w」



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);

課題1

- 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

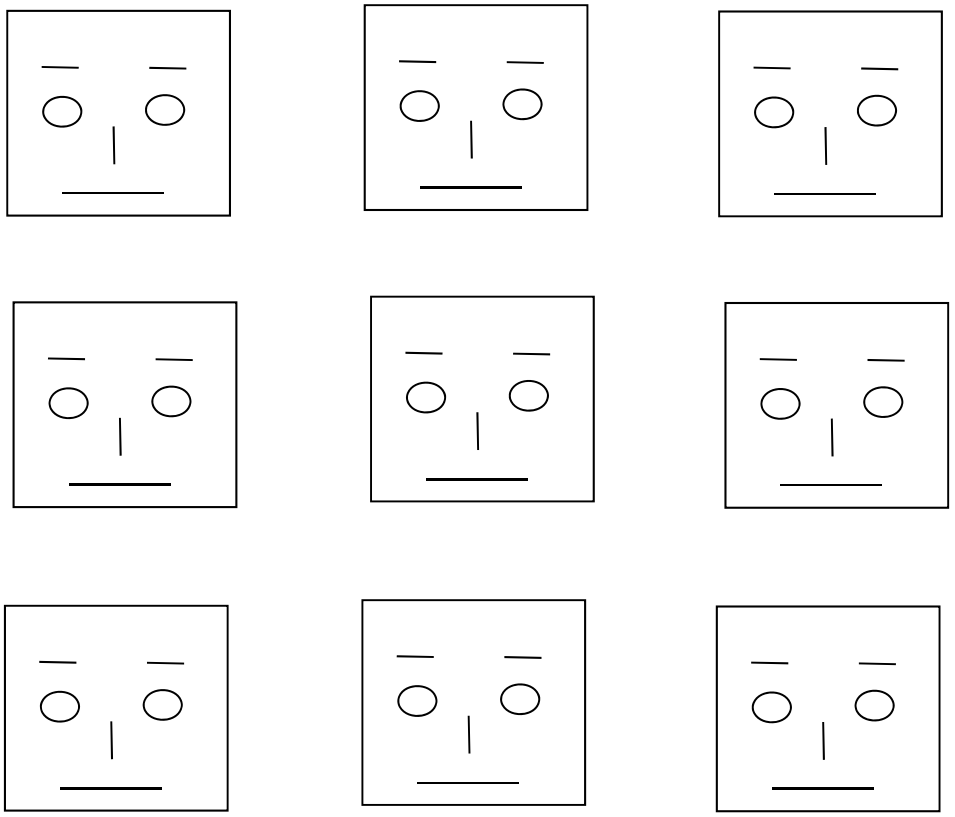
C++/C では、#define 文を使用した。
(例)

```
#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";
```

課題2



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。

```
}
```

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。