

基本文法の復習

2018/4/10(火) プログラミングIV 第一回

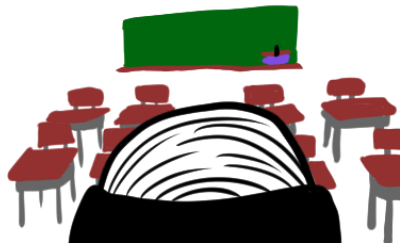
福井大学 工学部 電気電子情報工学科

長谷川達人

本講義の概要

講義名称 プログラミングIV

担当教員 長谷川, 福間先生



講義(2限目)

Web上でいつもの(質問や感想)
提出で出席とする

どちらも
計算機室



演習(3限目)

提示された課題をWeb上に
提出で出席とする

本講義の概要

配布資料について



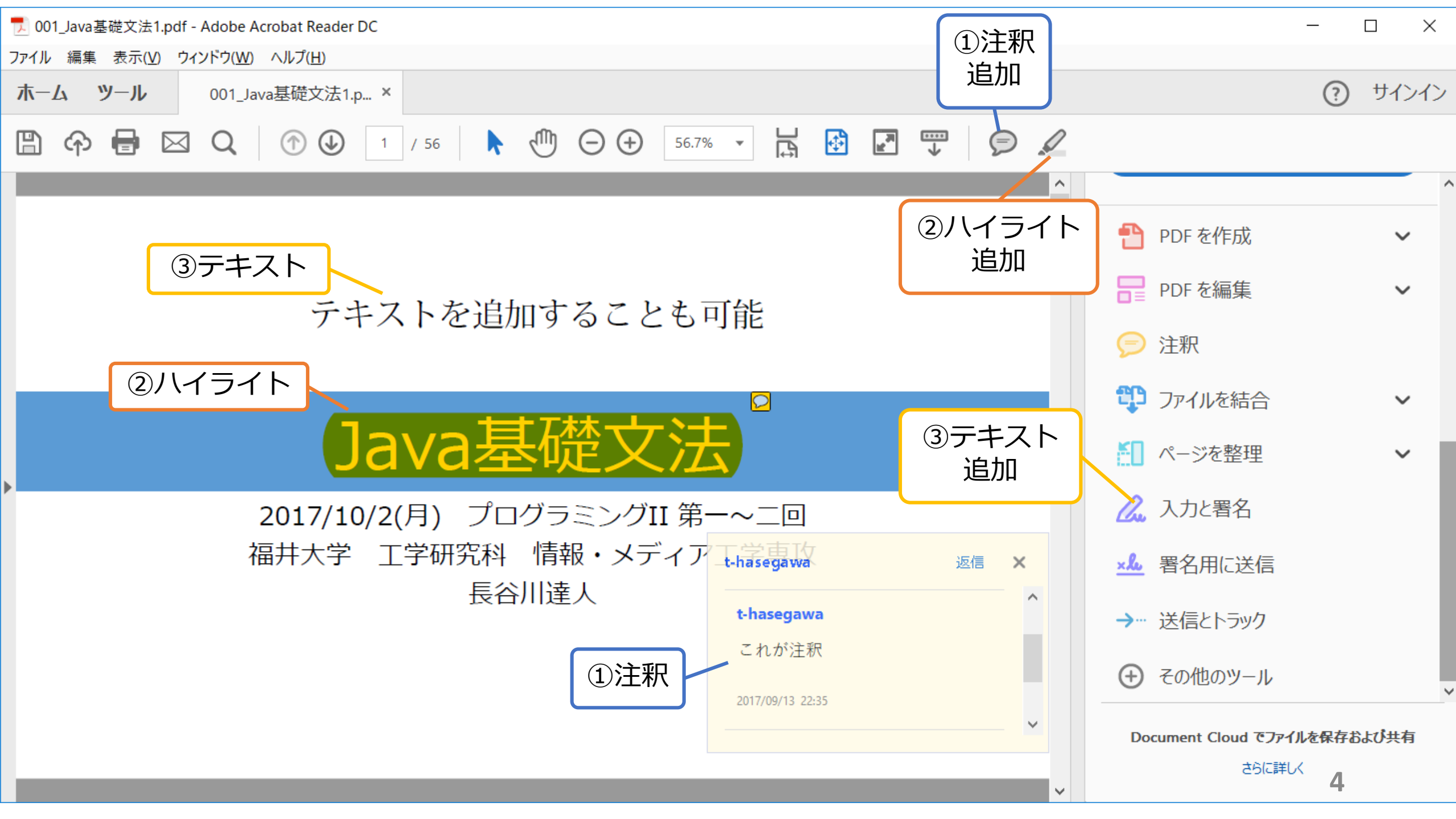
毎週の資料は学内のみWebで公開する（必要な人は**事前印刷**）．
印刷する場合は1Pに4枚で両面印刷を推奨する（用紙の節約）．
<http://hsgw-nas.fuis.u-fukui.ac.jp/lecture.html>



今年度開始の科目なので準備が直前になる可能性が高い．
あらかじめ，ご了承下さい．



Adobe Acrobat Reader DCの**注釈，ハイライト，入力と署名**
機能を使うと，PDFにメモが可能である（用紙の節約）．



①注釈
追加

②ハイライト
追加

③テキスト

②ハイライト

③テキスト
追加

①注釈

テキストを追加することも可能

Java基礎文法

2017/10/2(月) プログラミングII 第一～二回
福井大学 工学研究科 情報・メディア工学専攻
長谷川達人

t-hasegawa

返信

×

t-hasegawa

これが注釈

2017/09/13 22:35

PDFを作成

PDFを編集

注釈

ファイルを結合

ページを整理

入力と署名

署名用に送信

送信とトラック

その他のツール

Document Cloud でファイルを保存および共有

[さらに詳しく](#)

本講義の概要

前半		後半	
第1回	基本文法の復習	第9回	標準ライブラリ
第2回	クラス～カプセル化の復習	第10回	ファイル入出力
第3回	抽象クラス, インタフェース	第11回	デバッグ, インポート, 高速化
第4回	ポリモーフィズム	第12回	オブジェクト指向
第5回	GUI : Canvas	第13回	自由開発演習 1
第6回	GUI : Layout, イベントリスナ	第14回	自由開発演習 2
第7回	スレッド, 例外処理	第15回	自由開発演習発表会
第8回	ジェネリクス, コレクション	第16回	期末試験

何を開発するか, 少しずつ考えておくこと

本講義の概要

成績評価と参考図書

成績評価：全体

各回の課題・自由開発演習の評価・期末試験の成績
を総合評価し60%以上で合格とする。

参考図書

このような注釈を講義中で表示するが、注釈は公開PDFに
含まれないものもある。必要に応じてメモするとよい。

中山 清喬 他 (2014) 『スッキリわかるJava入門 第2版』
三谷 純 (2017) 『Java第2版実践編アプリケーション作りの基本』
※必携ではない (JavaではWeb上でPDF資料を公開する)

本講義の概要

理解には段階がある



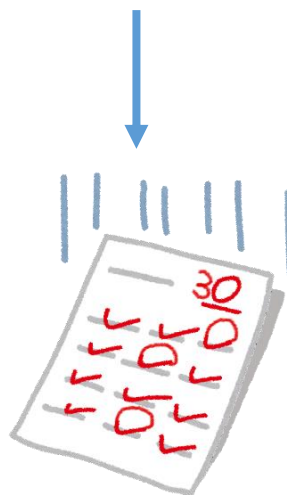
何もしていない



ここまでではできている人が多い



ちゃんと理解した



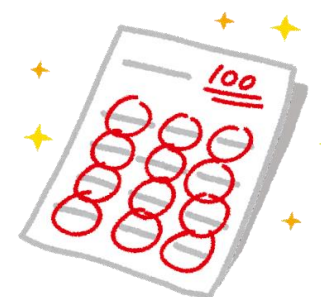
このギャップに戸惑うことが多い



ちゃんと理解した



何度も練習



本日の目標

概要

プログラミングIIで学習した前半部分を復習し、
Javaの感覚以前の基礎的なプログラミング感覚を取り戻す。

目標

Javaの超基本文法に関するプログラミングができる。

第一～二回目は復習なので、基本的なところは省略しながら
要点だけ説明し、練習時間を増やせればと思う。



なるほど

本日の提出課題

講義パート

できればわからないこと等のコメントはその場でしてくれるとありがたい。

課題 1

本日の授業を聞いて、
初めて知ったと思う内容を2点簡潔に述べよ。

課題 2

本日の授業を聞いて、
質問事項または**気になった点**を1点以上簡潔に述べよ。

課題 3

感想（あれば）

Java基礎文法

最もシンプルなJavaプログラム

ファイル「Sample1.java」の中身を以下とする。

Sample1.java	クラス名 : ファイル名と同じにする必要がある.
<pre>class Sample1{ public static void main(String[] args){ System.out.println("Hello world!"); } }</pre>	mainメソッド : 最初に実行されるメソッド. 書き方は 暗記する .
出力	コンソール出力メソッド : Cでいうprintf()である.
Hello world!	

Java基礎文法

最もシンプルなJavaプログラム

Javaプログラム記述時には次の点に気を付ける。

Sample1.java

```
class Sample1{  
    public static void main(String[] args){  
        System.out.println("Hello world!");  
    }  
}
```

括弧の位置：大括弧の開始は行末とする（改行後ではない）。

インデント：大括弧の中身はインデントを一つ上げる（タブキー）。

出力

Hello world!

Java基礎文法

変数宣言と型

JavaにもC言語同様に様々な変数型がある。
宣言，初期化方法はC言語と基本的には同じである。

```
int i = 0;  
long l = 0L;  
float f = 0.0f;  
double y = 0.0d;  
boolean b = false;  
char c = 'A';  
String s = "文字列";
```

0Lはlong型の0であることを明示している。
fはfloat, dはdoubleである。
大文字小文字は問わない。

boolean型 : true or falseの二値のみを取り扱う型

String型 : 文字列を取り扱う型

Java基礎文法

演算子

JavaにもC言語同様に様々な演算子がある。
宣言，初期化方法はC言語と基本的には同じである。

```
int i = 0;
i = 10 + 4; // + 加算 10+4=14
i = 10 - 4; // - 減算 10-4=6
i = 10 * 4; // * 積算 10*4=40
i = 10 / 4; // / 除算 10/4=2 (int型なので切り捨て)
i = 10 % 4; // % 余り 10%4=2 (商2, 余り2なので)
```

```
String s = "str";
s = s + "ing";
System.out.println(s);
```

文字列型変数と何かを「+」で
結ぶと文字列として結合される。

出力は文字列型変数を引数に指定するだけで良い。この場合
s = "str"+"ing"なので「string」と出力される。

Java基礎文法

代入演算子

JavaにもC言語同様に様々な代入演算子がある。
インクリメント, デクリメント演算子もある。

```
int i = 10;  
i += 4; // + 加算 10+4=14  
i -= 4; // - 減算 14-4=10  
i *= 4; // * 積算 10*4=40  
i /= 4; // / 除算 40/4=10  
i %= 4; // % 余り 10%4=2  
String s = "str";  
s += "ing"; // + 結合 str+ing=string
```

$i += 4;$ と $i = i + 4;$ は同義である。

```
i++; // ++ インクリメント 1加算される  
i--; // -- デクリメント 1減算される
```

$i++;$ と $i = i + 1;$ は同義である。

Java基礎文法

練習問題 1 : 久々のJavaプログラミング

以下を参考に，int型変数aとdouble型変数bの加算結果を表示するプログラムを完成させ動作結果を確認せよ。

```
class Renshu1{
```

1. mainメソッドを記述する.
2. int型変数aを定義し，初期値10を代入する.
3. double型変数bを定義し，初期値2.5を代入する.
4. コンソールに「12.5」のように表示させる.

```
}
```

ただし2.5単体でもdoubleと判断できるような表記とする.

Java基礎文法

型変換

なお、大小関係は、
byte < short < int < long < float < double < String

Javaの型変換のタイミングは3種類ある.

- 代入時の自動型変換

- 小さい型→大きい型に代入したとき、自動的に大きい型になる.

```
short s = 10;  
int i = s;  
long l = i;
```

小さい型→大きい型
なので自動的に型変換

```
long l = 10L;  
int i = l;  
short s = i;
```

大きい型→小さい型
なのでエラー

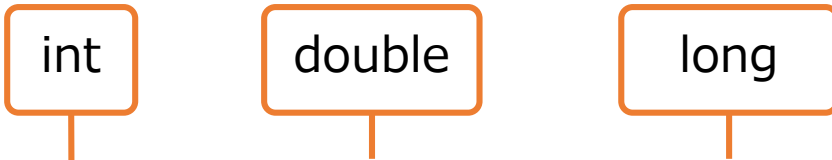
- 演算時の自動型変換 : 次頁以降で説明
- 強制的な型変換 : 次頁以降で説明

Java基礎文法

演算時の自動型変換

演算時には、**各演算で**大きい方の型に**統一されてから**計算されている。

`double d = 10 + 1.5d + 10L;`



`10.0d + 1.5d = 11.5d`



`11.5d + 10.0d = 21.5d`



Java基礎文法

強制的型変換（キャスト）

キャスト演算子を明示的に記述することで、大きい型→小さい型でも、強制的に型変換を行うことができる。

```
long l = 10000000000L;  
int i = (int)l;  
short s = (short)i;  
System.out.println(l + "," + i + "," + s);
```

(int) や (short) を
キャスト演算子と呼ぶ。



10000000000,10000000000,-13824

ただし、情報落ちすることがある。
この場合、long→intは大丈夫だが、int→shortで情報が落ちた。

Java基礎文法

強制的型変換（キャスト）

文字列→数値のキャストも可能である.

```
String str = "12345";  
int i = Integer.parseInt(str);  
i += 100;  
System.out.println("" + i);
```

String→intのキャスト

int→Stringの
演算時の自動型変換

12445

strが数字の場合は成立するが、
文字が含まれている場合エラーが発生する.

Java基礎文法

練習問題 2 : 久々のJavaプログラミング

先程のプログラムを顧客に提出したところわかりにくいとクレームが入った. 4の部分を下のように改造してほしい.

```
class Renshu1{
```

1. mainメソッドを記述する.
2. int型変数aを定義し, 初期値10を代入する.
3. double型変数bを定義し, 初期値2.5を代入する.
4. コンソールに「**a+b=12.5**」のように表示させる.

```
}
```

Java基礎文法

条件分岐 (if)

```
if( [ 条件式1 ] ){  
    // [ 条件式1 ] が true 時の処理  
} else if( [ 条件式2 ] ){  
    // [ 条件式1 ] が false かつ [ 条件式2 ] が true 時の処理  
} else{  
    // [ 条件式1 ] と [ 条件式2 ] が false だったときの処理  
}
```

if文等で条件式を書くときに使う演算子を関係演算子と呼ぶ。
関係演算の結果は**boolean型**となる。

```
int i=100;  
boolean flag1 = (i == 100);
```

i == 100
↓
true

変数flag1
True

Java基礎文法

条件分岐 (if)

したがってJavaでは、if文等で指定する条件式はboolean型の結果をもとに条件の認否を決定している。

Java

```
int day = 10;  
if(day % 2 == 0){}
```

判定

```
if(true){}
```

trueなので実行する

C言語

```
int day = 10;  
if(day % 2 == 0){}
```

判定

```
if(1){}
```

0でないので実行する

Java基礎文法

条件分岐 (if)

String型の一致確認は「==」ではなく「.equals()」を使う。

```
String str1 = "str";  
if(str1.equals("str")){}
```

==でも正しく動く（こともある）が、システムの安全性を考えると非推奨である。

ANDは「&&」，ORは「||」，NOTは「!」を用いてboolean型変数を「0:false, 1:true」と見たときに論理演算ができる。

```
boolean flagA = true && false;    // 1 and 0なので0になる  
boolean flagO = true || false;    // 1 or 0なので1になる  
boolean flagN = !true;             // not 1  なので0になる
```

応用すると， 2つ以上の条件をif文に記述できる。

```
int i = 10, j = 20;  
if(i == 10 && j == 20){}    // 1 and 1なので1(true)になる
```

Java基礎文法

繰り返し (for & while)

初期化 継続条件 ステップ

```
for(int i = 0; i < 10; i++){  
    System.out.println(i);  
}
```

ソースコードの途中やforの中でも変数宣言ができる。

スコープを限定した変数として使えて便利！
(この中でだけ「i」は有効)



forもwhileも同じ動作が実現できる

初期化 継続条件 ステップ

```
int i = 0;  
while(i < 10){  
    System.out.println(i);  
    i++;  
}
```



Java基礎文法

continueとbreak

処理をスキップするcontinueとループを終了するbreakがある。

```
int i=0;
while(true){
    i++;
    if(i<10)    continue;
    System.out.println("実行中:" + i);
    if(i>100)   break;
}
```

continue : ここまで実行して
ループに戻るという動作をする。
(これ以降の処理をスキップする)

break : ここまで実行して
ループを抜けるという動作をする。

実行中:10
実行中:11
...(略:101まで)

Java基礎文法

練習問題 3 : ifとforとrandom

Math.random()は0以上1未満の乱数を生成するメソッドである.
これとif文for文を用いて, グーかチョキかパーをランダムで10回
出力せよ.

Java基礎文法

その他 (switch, do-while)

Switch文

```
switch([変数]){  
  case 1:  
    // [変数]が1の時の処理  
    break;  
  case 2:  
    // [変数]が2の時の処理  
    break;  
  default:  
    // [変数]が1,2以外の時の処理  
    break;  
}
```

Do-while文

```
do{  
    // [継続条件]を満たす時の処理  
} while([継続条件]);
```

- Switchは1変数の一致で条件分岐をしたい場合に使用できる条件分岐構文である.
- Do-whileは1度以上はループ内の処理を実行したい場合に使用する繰り返し構文である.
 - ✓ forやwhileは最初に継続条件を確認するため、継続条件が最初から満たされない場合1度もループ内の処理が実行されない.

Java基礎文法

配列と拡張for (for-each)

配列の宣言はこの書き方を暗記すること。
変数型名[] 配列名 = **new** **変数型名**[要素数]

要素数を変数で（動的に）定義可能である。
C言語ではmallocやcallocを使う必要があった。

要素の参照方法はC言語と同じである。
自動的に0で初期化してくれている。

配列名.lengthで配列の長さが取得可能である。
.length()ではないことに注意されたい。

{ }を用いることでカンマ区切りで初期化しながら配列を生成することができる。

array[0]から一要素ずつstrに格納し、
ループ中の処理を実行する。

```
int i=10;
int[] array = new int[i];
System.out.println("array[0]:" + array[0]);
System.out.println("length:" + array.length);

String[] array = {"a", "i", "u", "e", "o"};
for(String str : array){
    System.out.print(str + ",");
}
```

Java基礎文法

コマンドライン引数

メインメソッドの引数(`String[] args`)は暗記するとしてきたが、実はコマンドラインから実行した際に送られてくる引数である。すなわち、Javaプログラムの外から送られてくる引数である。

```
public class Sample1{  
    public static void main(String[] args){  
    }  
}
```

このこと

メソッドの使い方

```
public static void main(String[] args){  
    double res = 0.0d;  
    // 呼び出し方法  
    // [メソッド名]([引数], ...);  
    res = methodA(5, 1.2d);  
}
```

重要！

```
// 宣言方法  
// public static [戻り値] [メソッド名]([引数], ...){  
public static double methodA(int num, double val){  
    return num*val;  
}
```

重要！

public static は各々意味はあるが、現段階ではおまじないとしておく。
プロトタイプ宣言不要 + 引数戻り値に配列が利用可能な点でC言語と異なる。

メソッドの使い方

変数のスコープ（有効範囲）

変数のスコープは宣言されてから
同じブロックが終了するまでの
間に限定される。

ブロック：括弧の開始 "{" から、
括弧の終了 "}" までで囲まれた
領域のこと

int型変数iが二回登場して
いるが、お互いの処理が
影響し合っていない。

```
public class Sample1{  
    public static void main(String[] args){  
        int i = 10;  
        methodA();  
        i *= 2;  
        System.out.print(i+",");  
    }  
    public static void methodA(){  
        int i = 10;  
        i *= -2;  
        System.out.print(i+",");  
    }  
}
```

[出力] -20,20,

メソッドの使い方

オーバーロード

- 配列引数を用いて複数変数を表示するメソッドを実装したが、毎回配列を宣言して呼び出すのは少々面倒である。普通にint型引数を使って実装はできないものだろうか。

```
public static void main(String[] args){  
    printInts(100);  
    printInts(100,200);  
}
```

1変数でも2変数
でも実行できる。

```
public static void printInts(int num){  
    System.out.println(""+num);  
}
```

オーバーロード：同じ名称のメソッド
を多重定義することができる。その際、
引数の**数か型を変える必要**がある。

```
public static void printInts(int num1, int num2){  
    System.out.println(num1 + ", " + num2);  
}
```

100 100, 200	[出力]
-----------------	------

メソッドの使い方

可変長引数

- オーバーロードでは、1引数の場合、2引数の場合、... をすべて事前に定義する必要があり面倒であった。もっと良い方法はないだろうか。

```
public static void main(String[] args){
```

```
    printInts(100);
```

```
    printInts(100,200);
```

```
}
```

```
public static void printInts(int... nums){
```

```
    for(int n : nums){
```

```
        System.out.print(n + ", ");
```

```
    }
```

```
    System.out.println("");
```

```
}
```

可変長引数：いくつでも繰り返してよい引数を「...」で定義することができる。使用時は通常の配列と同じ扱いとなる。

- 1メソッドにつき1種類のみ定義可能
- 最後の引数にのみ適応可能

// 最後に改行

100
100, 200

[出力]

Java基礎文法

練習問題 4 : ifとforとrandom

1. 練習問題 3 のプログラムを拡張し, arr[0]にグーが出た回数, arr[1]にチョキが, arr[2]にパーが出た回数をカウントし, それぞれの出現回数を表示するようにせよ.
 2. さらにこれを以下のようなメソッドにせよ.
 - 引数 : じゃんけん試行回数を示す数値 n
 - 戻り値 : グーチョキパーの出現回数が格納された長さ3のint型配列
- 乱数がちゃんとばらけているのかを試行回数を増やすことで確認できる.

Javaのメモリ領域

Javaプログラムのデータは、メモリ内のスタック領域、ヒープ領域というところに格納される。

基本型（プリミティブ型）の場合

```
byte bVal = 1;  
int iVal = 10;  
double dVal = 100.0;
```

bValは190番地から1byte
iValは200番地から4byte
dValは210番地から8byte

スタック領域

01011100 bVal = 1
1100100101000110
iVal = 10
0101010000111011
dVal = 100.0

ヒープ領域

0101110010101000
1100100101000110
1111011111011010
1010101111110011
0101010000111011
1011010101010101
1101100011010101

Javaのメモリ領域

Javaプログラムのデータは、メモリ内のスタック領域、ヒープ領域というところに格納される。

インスタンスの場合

```
Monster m1 = new Monster();
```

- ① : Monsterが入る領域をヒープ領域に確保する.
- ② : 参照値を格納する領域をスタック領域に確保する.
- ③ : ②の領域に①の参照値を格納する.

スタック領域

```
0101110010101000
1100100101000110
1111011111011010
1010101111110011
m1 = 195 00111011
1011010101010101
1101100011010101
```

ヒープ領域

```
0101110010101000
195番地 0101000110
Monsterが入る領域
  hp
  name
  ap
1101100011010101
```

Javaのメモリ領域

Javaプログラムのデータは、メモリ内のスタック領域、ヒープ領域というところに格納される。

配列の場合

```
int[] arr = new int[5];
```

- ① : `int[5]`が入る領域をヒープ領域に確保する。
- ② : 参照値を格納する領域をスタック領域に確保する。
- ③ : ②の領域に①の参照値を格納する。

スタック領域

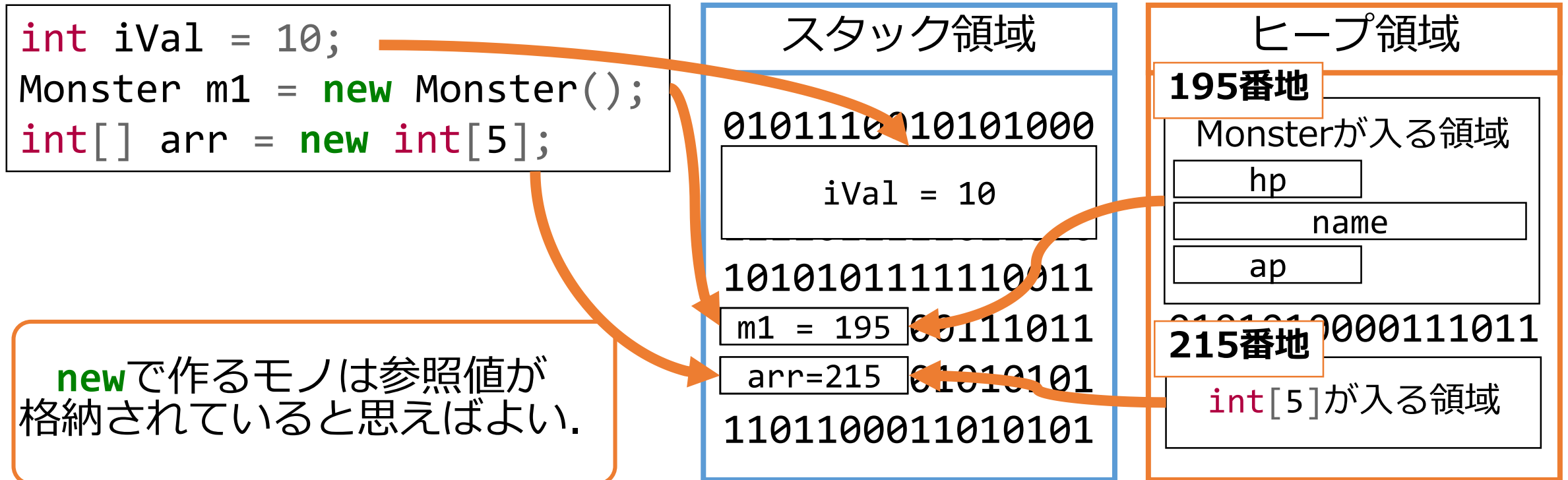
0101110010101000
1100100101000110
1111011111011010
1010101111110011
arr=215 00111011
1011010101010101
1101100011010101

ヒープ領域

0101110010101000
215番地 00101000110
int[5]が入る領域
arr[0]=0 arr[1]=0
arr[2]=0 arr[3]=0
arr[4]=0
1101100011010101

Javaのメモリ領域

基本型（プリミティブ型）は変数領域に値が直接入る。
参照型（インスタンスや配列）は変数領域に参照値が入る。

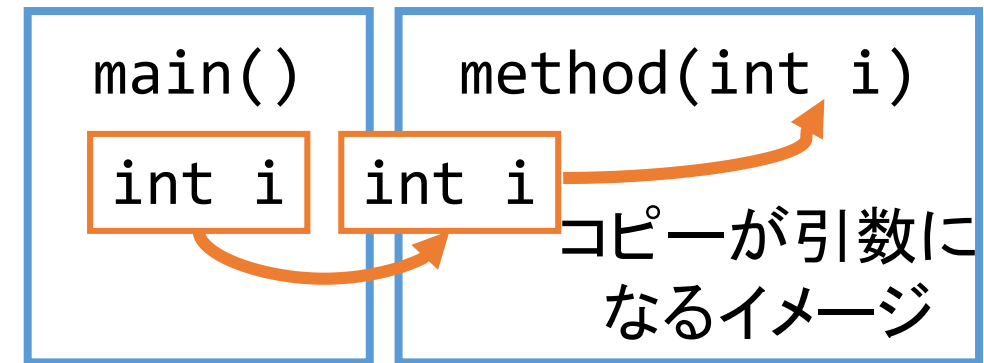


Javaのメモリ領域

int型等, 通常型の時

メソッドに引数で値を送るとき、
実物は送らず値だけを送る

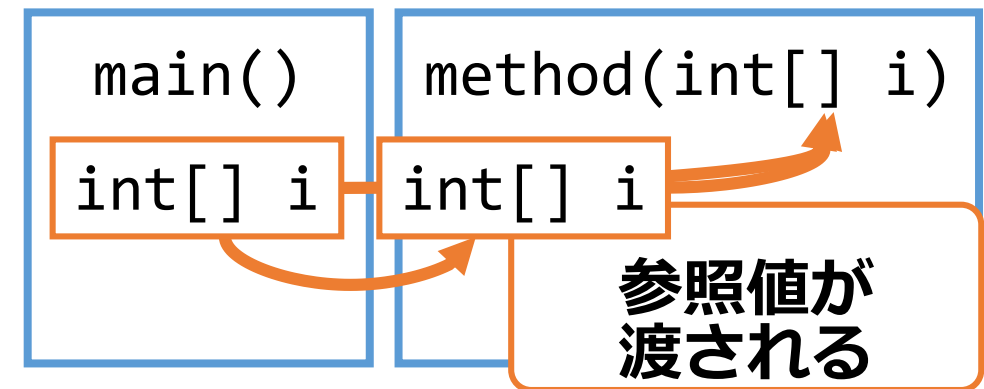
→メソッド内で値を編集しても、
呼び出し元変数は変わらない。



配列やインスタンス等, 参照型の時

メソッドに引数で値を送るとき、
実物をそのまま送るイメージ

→メソッド内で値を編集すると、
呼び出し元変数も変更される。



次週予告

※次週以降も計算機室

前半

クラス～カプセル化までの復習に関する講義を行う。

後半

講義内容に関するプログラミング演習課題に取り組む。