オブジェクト指向プログラミング

第１回：ガイダンス

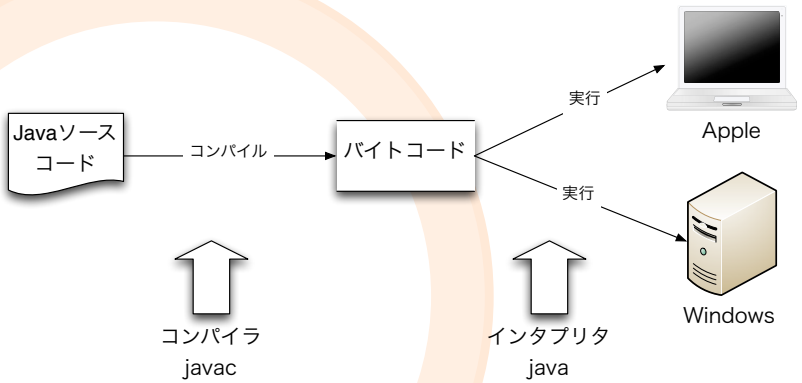
知識科学教育研究センター
金井 秀明

Javaって何

Java言語

- プログラミング言語の１つ
- 1995年Sun Microsystemsが発表
- 特徴
  - OS非依存： Write once, run anywhere
  - オブジェクト指向言語
  - 安全で生産性が高い

Javaの実行の流れ



プログラミングの基礎

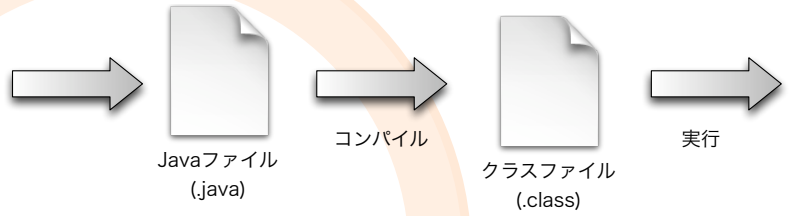
- 変数，データ型
- 式と演算子
- 条件分岐
- 繰り返し（ループ）

授業では

- K119 「基礎プログラミング」で履修した事項を前提として，授業を進めます．
  - 変数，データ型，演算子
  - 条件分岐
  - ループ
  - 配列
  - 関数

コンパイル・実行

作成の流れ



テキストエディタでJavaファイルを編集，保存する

コマンドプロントからjavacコマンドの実行

コマンドプロントからjavaコマンドの実行

Javaソースコード

- ソースコード：テキスト形式のプログラム
- 例）ファイル名：Hello.java

```
class Hello
{
        public static void main (String args[]) {
            System.out.println(”こんにちは”);
        }
}
```


コンパイル

- コンパイル：ソースコードをバイトコードに変換すること.
- コンパイラー：コンパイルするためのソフトウェア（javac)
- 例）Hello.javaをコンパイルする.

```
> javac Hello.java
```


コンパイルの後

- コンパイル後に生成されるファイルのことを「クラスファイル」とよぶ.
- 例）Hello.javaをコンパイルした後
  - Hello.classというファイルが生成される.

プログラムの実行

- 例）Hello.javaを実行する.

```
> java Hello
```

- 例）Hello.javaの実行結果

```
こんにちは
```

書式：クラスの構造

- Javaプログラムの単位はクラス(class)

```
class Hello {
        ここにクラスの中身を記述する
}
```

- プログラムの始まる部分：mainメソッド

```
class Hello {
  public static void main(String[] args) {
        ここにmainメソッドの中身を記述する
  }
}
```


書式：クラスの雛形

```
class クラス名 {
  public static void main(String[] args) {
        行わせた処理
  }
}
```


コードの要素

- クラス名「Smaple1」：プログラムグロックの基本単位，ファイル名
- main()メソッド：プログラムの処理が始まる
- 「{　}」で囲まれた部分：ブロック
- 文：１つの処理の単位，文の最後は「；」で終わる.
- コメント文：「//」から始まる文，「/* */」で囲まれた部分

例

```
// 画面に文字を出力するコード
class Sample1
{
  public static void main (String args[])
  {
    System.out.println(”ようこそJavaへ!”);
    System.out.println(”Javaをはじめましょう”);
  }
}
```

練習

- 以下のように実行すると，

```
> java Rensyu1
```

- 画面に次のように表示するJavaプログラムを作成，実行してください

```
Javaは以下のような特徴をもつ.
OS非依存
オブジェクト指向
安全で生産性が高い
```


復習問題１

- キーボードから入力したint型の整数x,yを加算，減算，乗算，乗算した値を表示するプログラムを作成せよ.

復習問題２

- キーボードから入力したdouble型の半径をもつ円の円周の長さと面積を表示するプログラムを作成せよ.

復習問題３

- キーボードから入力したdouble型の２つの実数値のうち大きい方の値を表示するプログラムを作成せよ.

復習問題４

- キーボドから入力したテストの点数に応じて，成績を表示するプログラムを作成せよ．
  - 0～59:D, 60～69:C, 70～79:B, 80～100:A

復習問題５

- 以下のように表示するプログラムをfor文を用いて作成せよ．

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```


復習問題6

- 2009年現在の日本の人口は127,522,000人です．年0.5%の割合で人口が減少した場合，2010年の人口は何人になるか計算して，以下のように表示するプログラムを作成せよ．

復習問題7

- 2009年現在の日本の人口は127,522,000人です．年0.5%の割合で人口が減少した場合，2030年の人口は何人になるか計算して，以下のように表示するプログラムを作成せよ．

復習問題８

- 各学生の平均点を計算し，その値を配列averageに格納し，表示するプログラムを作成せよ

配列score

	英語	数学	社会	理科
学生A	60	50	80	80
学生B	70	80	70	90
学生C	40	90	60	70

→ 配列average

平均

復習：メソッド

メソッド#1

- プログラムの処理の一部分をまとめて名前をつけたものを「メソッド」と呼ぶ.
- 通常，プログラムを書くときは
  - mainに全ての処理を書くのでなく，ある程度のまとまり毎に複数のメソッドに分割する.
  - 似たような処理を１つのメソッドにまとめる

プログラムをすっきりと書くことができる

メソッド#2

- メソッドのメリット
  - 可読性向上
  - コード量が減らせる
- 例
  - System.out.println
  - main
    - mainメソッドはjavaコマンドによって最初に実行される特別なメソッド

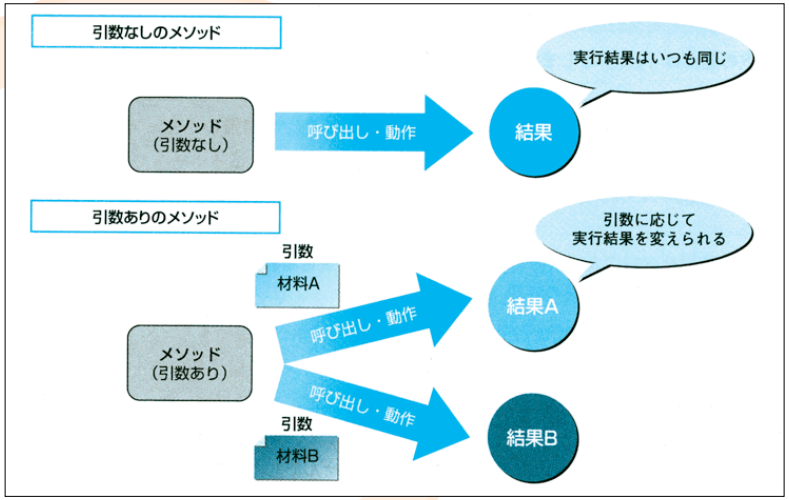
例

```
class Sample {
  public static void main(String[] args) {
    System.out.println("呼ぶ前");
    doSomething();
    System.out.println("呼ぶ後");
  }

  static void doSomething() {
    System.out.println("メソッドが呼ばれた");
  }
}
```


例

```
class Sample2 {
  public static void main(String[] args) {
    add(1, 2);
    add(20, 30);
  }

  static void add(int a, int b){
    int sum =a+b;
    System.out.println("結果="+ sum);
  }
}
```


引数

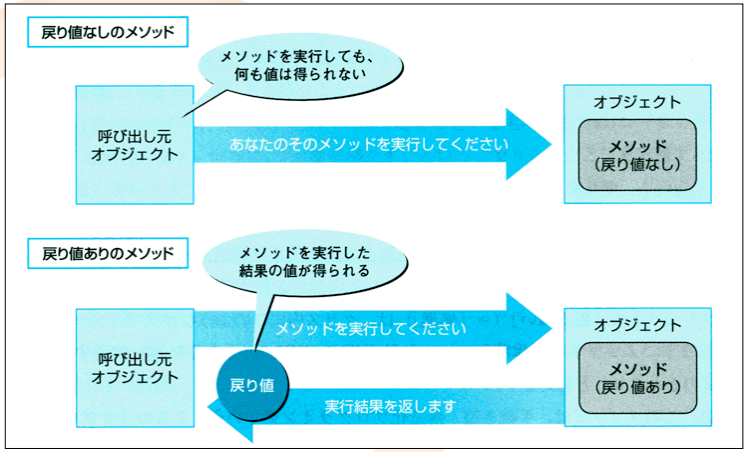


出典：立山秀利「Javaのオブジェクト指向がゼッタイにわかる本」秀和システム

例

```
class Sample3 {
  public static void main(String[] args) {
    double result = add(1, 2);
    System.out.println("結果="+ result);
  }

  static double add(int a, int b){
    double sum =(double)(a+b);
    return sum;
  }
}
```

○ 戻り値



出典：立山秀利「Javaのオブジェクト指向がゼッタイにわかる本」秀和システム

○ “お願い”に対して#1

- ”お願い”された動作を行う際
  - ”材料（=引数）”を全く必要としないタイプ
  - 動作に必要な”材料（=引数）”を受け取るタイプ
- 引数：メソッドに何らかの情報（値）を渡すとき使う．その情報のこと．

○ “お願い”に対して#2

- “お願い”された動作した後
  - お願いされた動作をして終わり
  - お願いされた動作をやり，その結果を依頼元のオブジェクトに返す（戻り値）
- 戻り値：メソッドから返される情報（値）

○ メソッドの書き方

```
修飾子 戻り値型 メソッド名(引数1, 引数2, ...) {
    メソッド内の処理
}
```

引数の構成は<データ型><名前>

- int add(double a, int b)
- public static void main (String[] args)

メソッド（戻り値・引数）

	戻り値あり	戻り値なし

引数あり	戻り値の型 メソッド名(引数1, 引数2, ...)
	{
	処理の文;
	
	return 戻り値;
	}
		void メソッド名(引数1, 引数2, ...) {
		処理の文;
		
		}

引数なし	戻り値の型 メソッド名() {
	処理の文;
	
	return 戻り値;
	}
		void メソッド名() {
		処理の文;
		
		}

練習１

- void printSeason(int m)
  - m:12,1,2のとき，Winterと表示する
    テキスト
  - m:3,4,5のとき，Springと表示する
  - m:6,7,8のとき，Summerと表示する
  - m:9,10,11のとき，Fallと表示する

練習２

- ３つのint型の整数値を受け取って，その最大値を求める返すメソッドmaxを作成せよ．

練習３

- ３人分の身長，体重を入力し，それぞれの最大値を表示するプログラムを作りましょう．
  - 各データは整数値とする．
  - 各データは配列に入れることとする．
  - 練習２で作成したメソッドmaxを用いる．

練習４

- int型の配列を受け取って，その最大値を求める返すメソッドmaxOfを作成せよ．

練習５

- n人分の身長，体重を入力して，それぞれの最大値を表示するプログラムを作りましょう．
  - 各データは整数値とする．
  - 各データは配列に入れることとする．
  - 練習４で作成したメソッドmaxOfを用いる．

多重定義#1

- 練習２と練習４で，複数の整数値から最大のものを求めるメソッドを作成した．
  - 練習２では，int max(int a, int b, int c)
  - 練習３では，int maxOf(int[] a);
- 機能的に同じなのだから，同じ名前のメソッドの方が，メソッドを使う場合分かりやすい．

多重定義#2

- Javaでは，１つのクラス中に同じ名前のメソッドを複数定義できる．これを多重定義（オーバロード）という．
- 例えば，
  - int max(int a, int b, int c);
  - int max(int[] a);

例：max#1

```
static int max(int a, int b, int c) {
  int max = a;
  if (b > max) max = b;
  if (c > max) max = c;
  return max;
}

static int max(int[] a) {
  int max = a[0];
  for (int i=1; i<a.length; i++) {
    if (a[i] > max) max = a[i];
  }
  return max;
}
```


例：max#2

```
static int max(int a, int b, int c) {
  int max = a;
  if (b > max) max = b;
  if (c > max) max = c;
  return max;
}

static int max(int a, int b, int c) {
  ....
  }

static double max(double a, double b, double
c){
  double max = a;
  if (b > max) max = b;
  if (c > max) max = c;
  return max;
}

...
```


メソッドのシグネチャ

- Javaでは，オーバロードされたメソッドから，必要なメソッドを見分ける方法として，メソッドのシグネチャを利用する．
  - メソッドのシグネチャとは，メソッドの「名前」，「引数の並び（型，個数）」のこと

```
int add(int a, int b) {...};
double add(double a, double b){...};
```


プログラムに引数を渡す

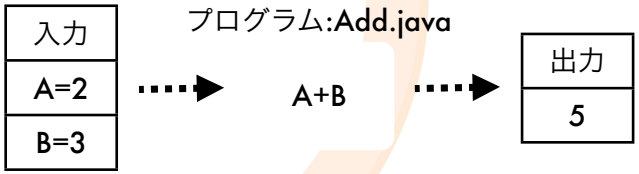
String[] args

```
class ArgsSample {
  public static void main(String[] args) {
      for (int i=0; i< args.length; i++) {
        System.out.println("args["+i+"]="+ args[i]);
      }
  }
}
```

```
>java ArgsSample aaa bbb

args[0]=aaa
args[1]=bbb
```


プログラムの引数

- 例えば，「A+B」というプログラムで，プログラム実行時にオペランド（AやB）の値を指定できると便利
- そのようなときに引数を使う



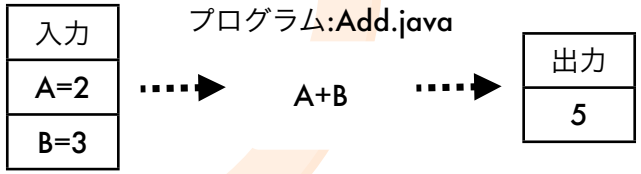
引数の渡し方

- 書式

```
java <クラス名> <引数1> <引数2> ...
```

- 例

```
java add 2 3
```



プログラムでは#1

- 書式：プログラム中で引数を使う
  - args[0]が１つ目の引数，args[1]が２番目の引数
  - プログラム中では，文字列（String型）として渡される．

```
//引数の練習 Hikisuu.java
class Hikisuu {
  public static void main (String[] args) {
    String s0 = args[0]; String s1 = args[1];
    System.out.println(args[0]);
    System.out.println(args[1]);
  }
}
```

プログラムでは#1

```
//引数の練習 Add.java
class Add {
  public static void main(String[] args) {
    System.out.println(args[0]+args[1]);
    }
}
```


プログラムでは#2

```
//引数の練習 Add.java
class Add {
  public static void main(String[] args) {
    int num0 = Integer.parseInt(args[0]);
    int num1 = Integer.parseInt(args[1]);
    System.out.println(num0+num1);
    }
}
```


文字列を他の型に変換

- 文字列をint型に変換

```
int num = Interger.parseInt(str);
```

- 文字列をdouble型に変換

```
double num = Double.parseDouble(str);
```