intloop（イントループ） × High Performer Consultant

プログラム作成(Java入門) I
～データ型・変数・配列・メソッド～

概要

プログラミング初心者がつまづきがちな、概念の説明に主眼をおいて懇切丁寧に説明します。
今更聞けない基本的なことから便利な使用例まで
図を用いて「具体的」「分かりやすく」「端的」に解説していきます。
（概念の理解を目的とするため、構文に関する解説は行いません）
プログラミングでつまずいている方、是非ご一読ください。

今回は、データ型・変数・配列・メソッドがテーマです。
最後までお付き合いいただけますと幸いです。

目次

1. はじめに

2. 変数とは

3. データ型とは

4. 便利な変数「配列」

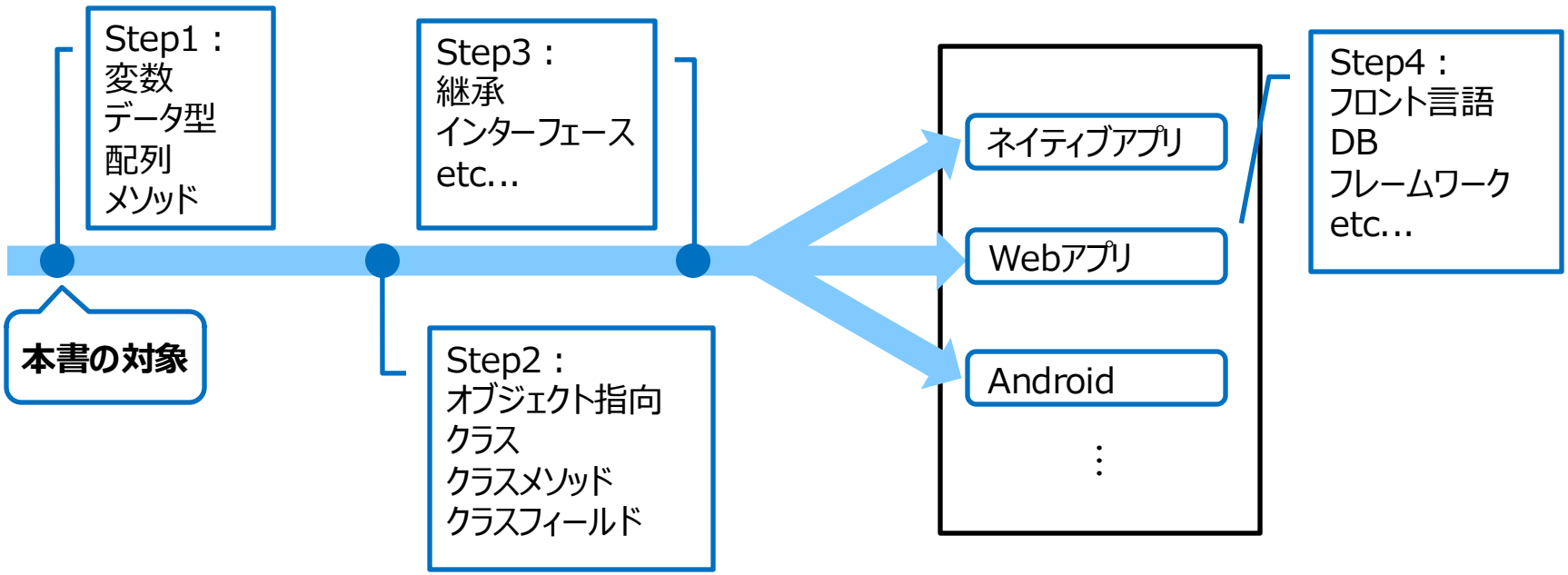
5. メソッドとは

6. まとめ

はじめに

プログラミングをするなら一度は触れておきたいJavaですが、
「データ型やメソッドの概念がわからない…」「オブジェクト指向って何…」などの理由で、
挫折した方は多いのではないでしょうか。
本項講ではJavaプログラミングでつまずきがちな要点について解説していきます。

◆Javaプログラミングの４ステップ



はじめに

プログラミングは習得すれば非常に有益なスキルとなりますが、身に付けるは容易ではありません。
STEP1では主に次のような点でつまずくケースが多くみられます。

◆こんな理由でつまずくことが多い！

- 変数への代入「i=i+1」って何…
- 配列の使い方がわからない…
- メソッドのデータ型って何？書き方は？
- メソッドの引数？戻り値って…？

原因は…

- 変数/配列の利便性を感じられない
- メソッドの呼び出し構造が理解できない

変数/配列を使うことで具体的にどのような利点があるのかコードをみて一目瞭然！
メソッドの構成、呼び出し構造について、図とコードを見ながら解説！

1. はじめに

2. 変数とは

3. データ型とは

4. 便利な変数「配列」

5. メソッドとは

6. まとめ

変数とは：変数の定義

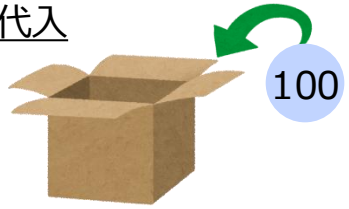
変数とは、特定の範囲内で変動する値を指し、一般に箱のイメージで表現されます。
同じ値を使用する箇所を変数に置き換えることで、修正が容易になったり、プログラムを簡略化できます。

◆変数は「宣言」「代入」「参照」をすることができる

せんげん
①宣言

箱(変数)を用意する。
その際に、箱の大きさと
中身の種類を決める。
（データ型で解説）

だいにゅう
②代入



用意されている箱に、
中身を入れる。
すでに入っている中身を捨て、
別の値に置き換える。

さんしょう
③参照

箱の中身を確認する。
あるいは、その中身を使って
作業をする。
（ただし、中身はなくならない）

◆変数は任意の値を格納できる（ただし、決まった形式のものしか入らない）

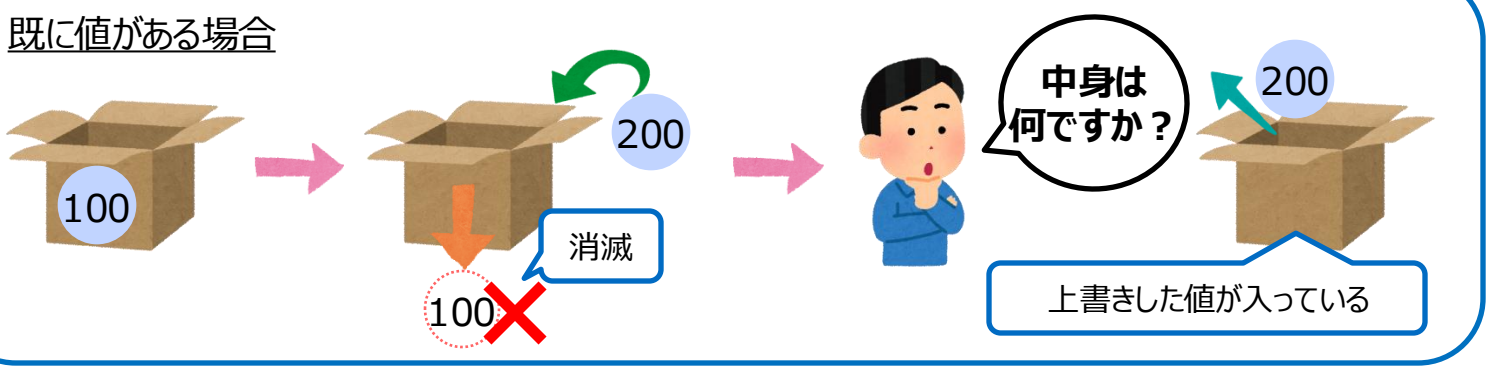
変数とは：代入・上書き

変数に値を入力した場合、最後に代入した値のみが格納されます。
それ以前に入力していた値は消滅してしまうため注意が必要です。

◆変数に値を代入する

空の場合

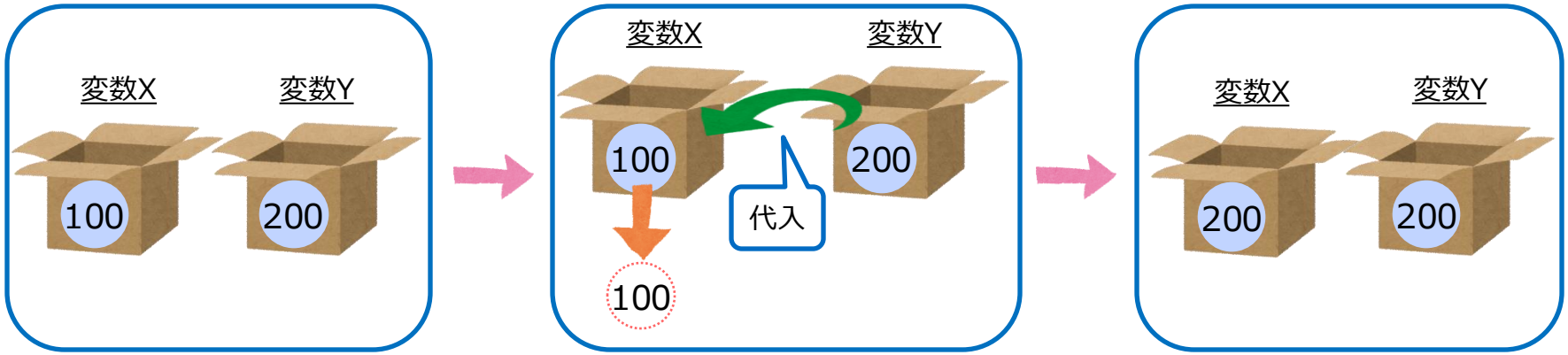
既に値がある場合



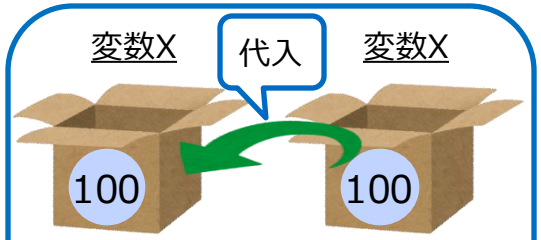
変数とは：代入・上書き

変数の代入では、ある変数の値を別の変数に代入することができます。
また、自分自身の値を代入することも可能です。

◆別の変数の値を代入する



◆自分自身の値を代入する



一見無駄な行為ですが、
これを利用して、
次のような作業が行えます。

変数「x」に1を足す場合

```
public class Sample01{
  public static void main(String args[]){

    int x = 5;      //変数を宣言
    x = x + 1;

  }
}
```

「x」に「x+1」を代入している

Tips

プログラミングの「＝」は
”等しい”という意味ではなく、
“代入”を意味します。
ここでは、変数「x」に「x+1」を
代入するため、
xに6が代入されます。

変数とは：変数の使い方

変数は次のような様々なパターンに対応する場合に使用することで、
変更する場所を変数の値のみにとどめることができる。

◆変数を使った場合と使わない場合では変更の容易さが違う
（例：九九5の段の表示をしたい場合）

変数を使わないと・・・

```
public class Sample02{
  public static void main(String args[]){

    System.out.println(5*1);
    System.out.println(5*2);
    System.out.println(5*3);
    System.out.println(5*4);
    System.out.println(5*5);
    System.out.println(5*6);
    System.out.println(5*7);
    System.out.println(5*8);
    System.out.println(5*9);

  }
}
```

もし、7の段に変更したい場合
「5」をすべて変更しなくてはならない

変数を使うと…

変数を使うと変更が簡単！

```
public class Sample03{
  public static void main(String args[]){

    int x = 5; //変数を宣言

    System.out.println(x*1);          //変数「x」掛ける1
    System.out.println(x*2);          ・
    System.out.println(x*3);          ・
    System.out.println(x*4);          ・
    System.out.println(x*5);
    System.out.println(x*6);
    System.out.println(x*7);
    System.out.println(x*8);
    System.out.println(x*9);

  }
}
```

別の段へ変える場合でも
ここを変更するだけ！

変数とは：変数の使い方

次のような繰り返し構文内などで変数を使用することにより、
プログラムを簡略化することができます。
一般的に繰り返し構文内では変数「i」を用いて記載します。

◆変数iを使用することでプログラムを簡略化できる

繰り返し構文を使わない場合

```
public class Sample04{
  public static void main(String args[]){

    int x = 5;  //変数を宣言

    System.out.println(x*1);                    //変数「x」掛ける1
    System.out.println(x*2);                          ・
    System.out.println(x*3);                          ・
    System.out.println(x*4);                          ・
    System.out.println(x*5);
    System.out.println(x*6);
    System.out.println(x*7);
    System.out.println(x*8);
    System.out.println(x*9);

  }
}
```

繰り返し構文を使うと…

繰り返し構文内を使う場合

```
public class Sample05{
  public static void main(String args[]){

    for(int i = 0; i < 9; i++){
      System.out.println(5*i);
    }

  }
}
```

繰り返し構文により1行で記述可能！

1. はじめに

2. 変数とは

3. データ型とは

4. 便利な変数「配列」

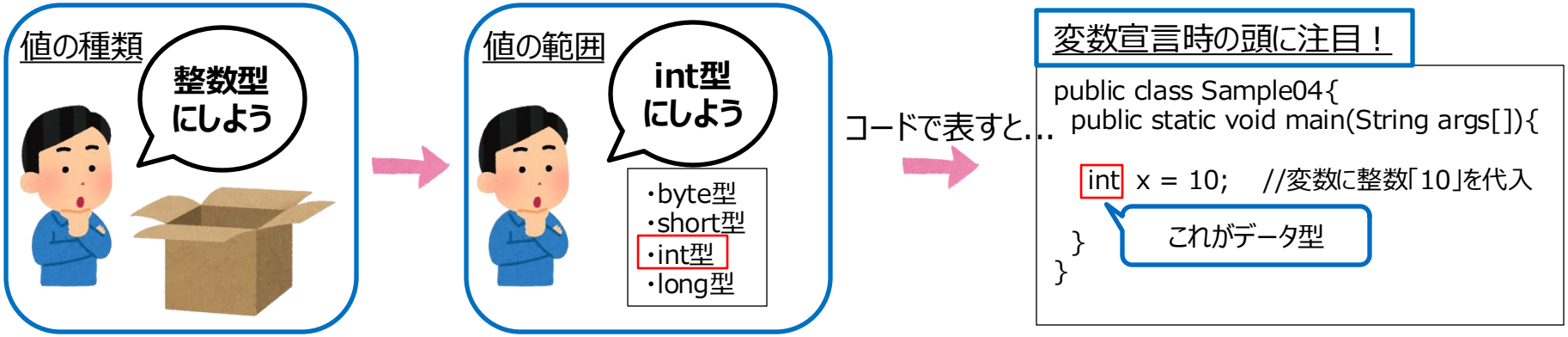
5. メソッドとは

6. まとめ

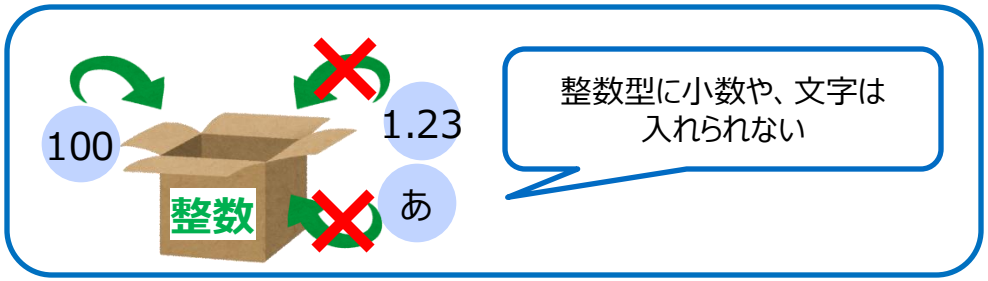
データ型とは：データ型とは

データ型とは格納される値がどのような性質を持っているのかを定義するものである。
（例：変数xは「2Byte」までの「整数」など）
データ型には複数の種類が存在し、数値の表現だけにしても「byte型」、「int型」、
「double型」等、いくつか用意されているため、必要に応じてデータ型を決める必要がある。

◆データ型で変数の性質を定めることができる



◆整数型に整数以外は入れられない



データ型とは：基本的なデータ型

次の表に基本的なデータ型の一覧を記載します。

データ型	種別	大きさ	説明
byte型	整数型	1バイト	-128～127までの整数
short型	整数型	2バイト	-32768～32767までの整数
int型	整数型	4バイト	-2147483648～2147483647までの整数　**※整数型は主にこれを使う**
long型	整数型	8バイト	int型では対応できない大きな範囲
float型	浮動小数点型	4バイト	±3.40E38～±1.40E45までの小数
double型	浮動小数点型	8バイト	float型では対応できない細かな範囲　**※小数型は主にこれを使う**
char型	文字型	2バイト	2バイトの文字コード
boolean型	論理型	-	真偽値（true、またはfalse）
String型	参照型※	-	文字列、型名の「S」は大文字なので注意

String型などの参照型について、
詳しく知りたい方は調べてみましょう

Tips

データ型は参照型を含めると　複数存在します。
本項では触れませんがクラスは参照型に含まれます。

1. はじめに

2. 変数とは

3. データ型とは

4. 便利な変数「配列」

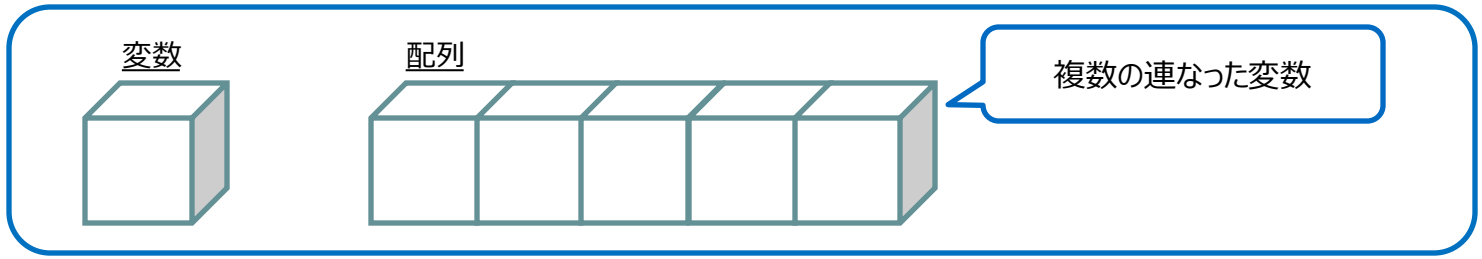
5. メソッドとは

6. まとめ

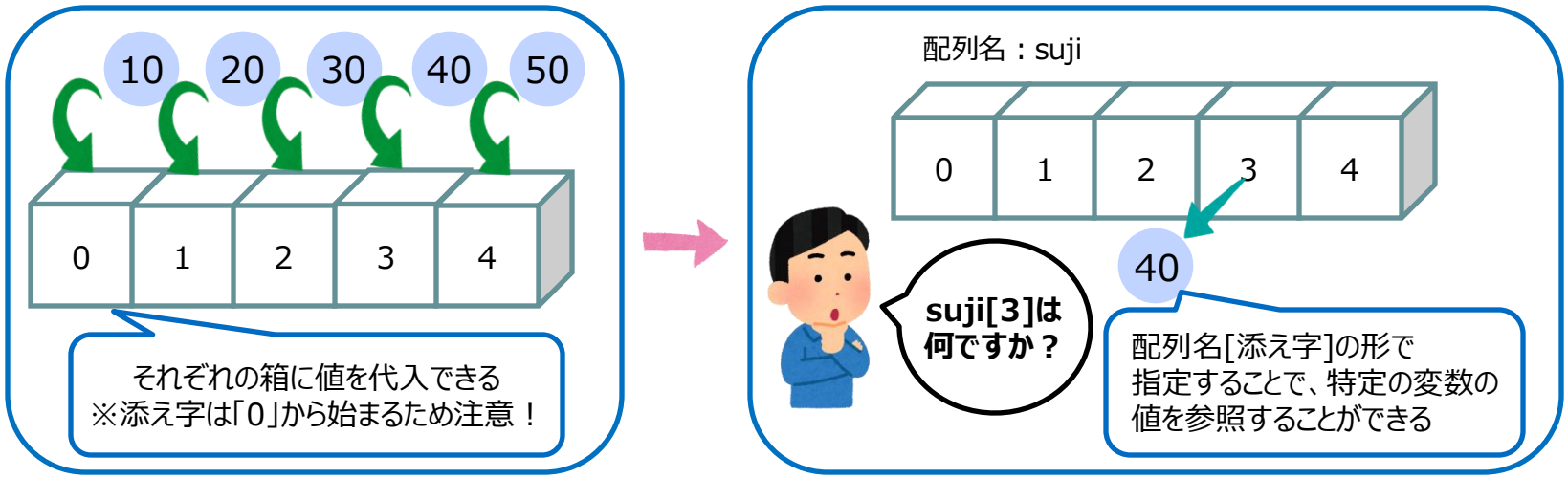
便利な変数「配列」：配列とは

配列とは、同じ名前を持った複数の変数の塊です。
複数の関連した変数を定義する場合、配列を用いることでプログラムを簡略化できます。
配列は「添え字」と呼ばれる配列内の番地によってどの値を参照するかを指定します。

◆配列は変数が連なっているイメージ



◆「添え字」によって箱を指定することができる



便利な変数「配列」：配列の実用例

配列は、複数の関連する情報を扱う際に使用するケースが多くみられます。
具体的には次のようなケースなどで配列を使用すると便利です。

◆10人の年齢を格納する場合を変数と配列で比較すると・・・

変数を使う場合

```
public class Sample06{
  public static void main(String args[]){

    int age0 = 20;
    int age1 = 23;
    int age2 = 41;
    int age3 = 22;
    int age4 = 19;
    int age5 = 34;
    int age6 = 56;
    int age7 = 23;
    int age8 = 43;
    int age9 = 11;

  }
}
```

10回宣言が必要

配列で宣言すると…

配列を使う場合

```
public class Sample07{
  public static void main(String args[]){

    int[] age =
{20,23,41,22,19,34,56,23,43,11};

  }
}
```

配列を使えばコードをかなり簡略化できる！
※配列のデータ型は末尾に[]をつける

便利な変数「配列」：添え字を変数にする利便性

配列の添え字は変数で記述されることが多いです。
繰り返し構文中では処理の周回ごとに別の値を取得することが可能となります。

◆10人の年齢を表示したい場合、添え字を変数で表現すると・・・

添え字に変数を使わない場合

```
public class Sample08{
  public static void main(String args[]){

    int[] age = {20,23,41,22,19,34,56,23,43,11};

    System.out.println(age[0]);
    System.out.println(age[1]);
    System.out.println(age[2]);
    System.out.println(age[3]);
    System.out.println(age[4]);
    System.out.println(age[5]);
    System.out.println(age[6]);
    System.out.println(age[7]);
    System.out.println(age[8]);
    System.out.println(age[9]);
  }
}
```

10行必要だが…

添え字を変数にして
forループさせる

添え字に変数を使う場合

```
public class Sample09{
  public static void main(String args[]){

    int[] age = {20,23,41,22,19,34,56,23,43,11};

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

添え字を変数にすることで、
コードを簡略化！

1. はじめに

2. 変数とは

3. データ型とは

4. 便利な変数「配列」

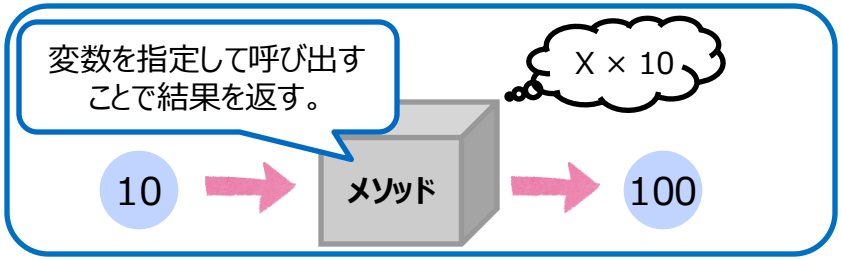
5. メソッドとは

6. まとめ

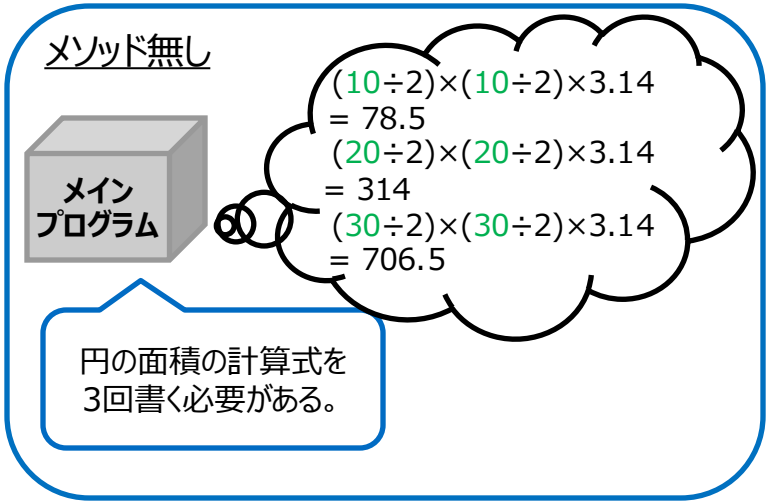
メソッドとは：メソッドの使い方

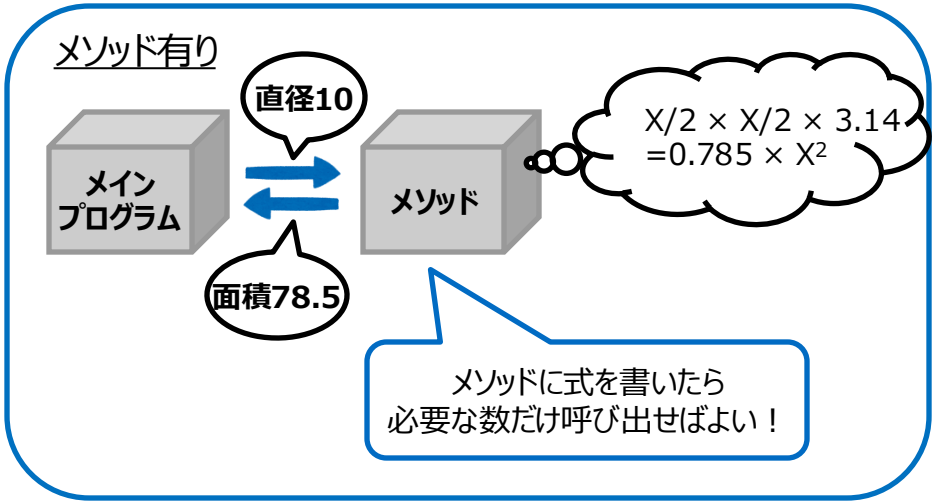
メソッドとはある処理をまとめたものです。
繰り返し行う処理をメソッドとしてまとめることで、その処理を1行で呼び出すことができます。

◆メソッドは関数のようなもの



◆一度書いてしまえば呼び出すだけ！
（例：直径10と20と30の円の面積を求めたい場合）

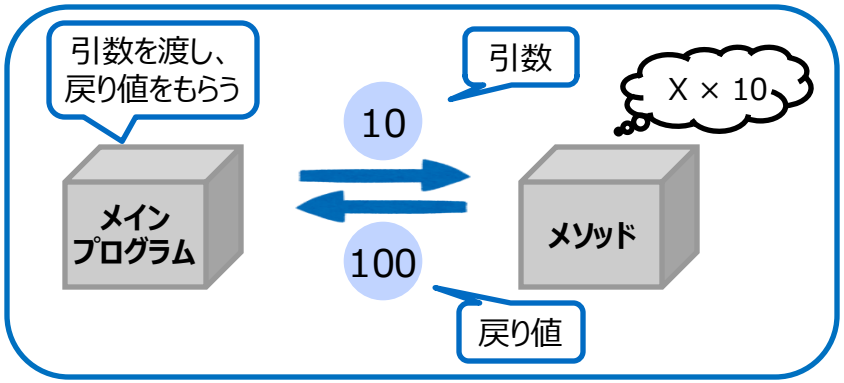




メソッドとは：メソッドの引数と戻り値

メソッドには引数と戻り値が存在する。
メソッドの処理において外部から指定する必要のある値を「引数」。
処理の結果を「戻り値」と呼ぶ。

◆引数と戻り値とは



◆引数の指定

```
public class Sample10{
  public static void main(String args[]){
    System.out.println(areaCircle(10));
  }

  public static  double areaCircle (int diameter){
    double area = 3.14 * (diameter/2) * (diameter/2);
    return area;
  }

}
```

引数

戻り値

◆メソッド部分の解説

```
                ①      ②          ③
public static  double  areaCircle (int diameter){
        ④
  double area = 3.14 * (diameter/2) * (diameter/2);

  return area;
}       ⑤
```

①：メソッドの型を指定しています。
「return」によって指定される戻り値は、ここで指定する型と同じ必要があります。

②：メソッドの名前です。メインプログラムから呼び出す際に指定します。

③：メインプログラムからメソッドに渡す引数の型と、その引数をメソッド内で扱う際の変数名を指定します。

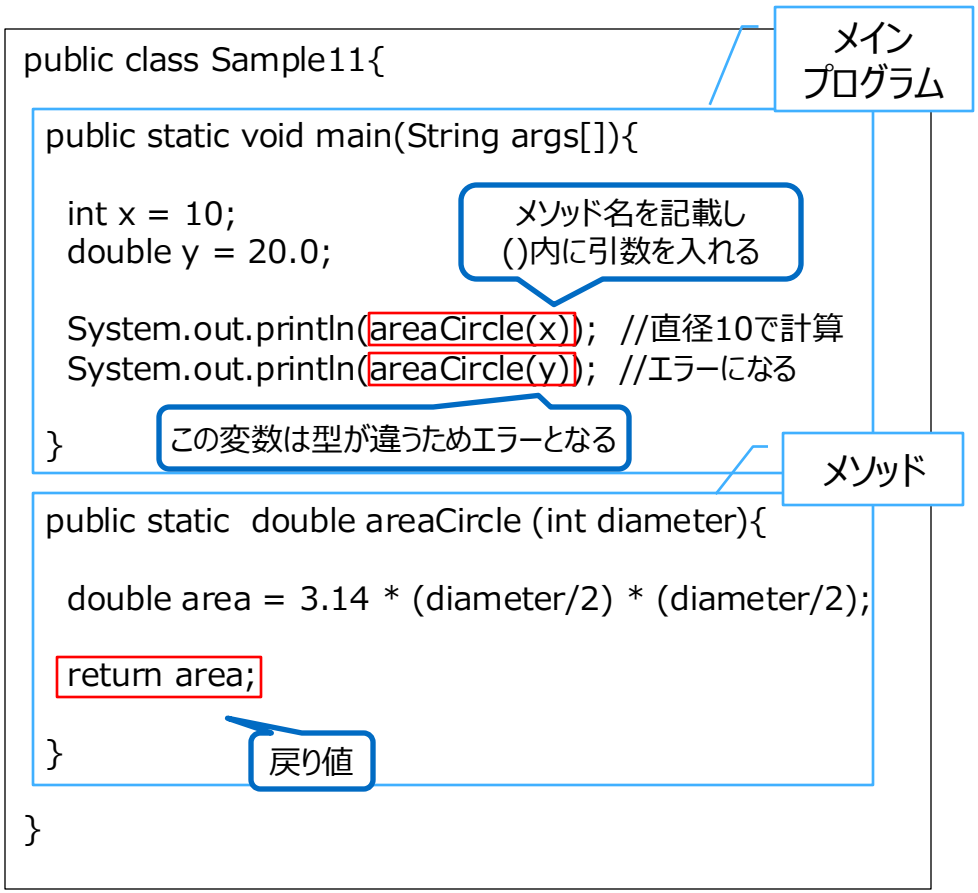
④：変数の宣言（⑤の戻り値として使用するためメソッドの型と同じである必要があります）

⑤：メソッドが最終的にメインプログラムに返す値を指定します。
①のメソッドの型と同じ型の変数や値を指定する必要があります。

メソッドとは：メソッドの使用例

メソッドの呼び出しは本項内で特につまずきやすい点です。
具体的なコードをもとに解説をしていきます。

◆メソッドの定義と呼び出し方



Tips

メソッドの型には「void」という型が用意されています。
void型とは戻り値のないデータ型です。
例えば、「指定された文字を出力する」といった働きだけのメソッドであれば、戻り値が必要ないためvoid型を指定します。
void型は戻り値がないため「return」の記述は省略可能です。

1. はじめに

2. 変数とは

3. データ型とは

4. 便利な変数「配列」

5. メソッドとは

6. まとめ

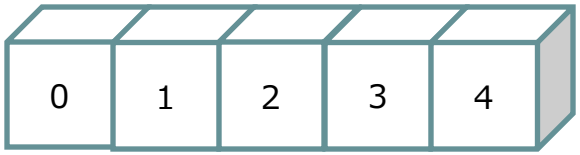
まとめ

本項で取り上げた用語についてまとめです。

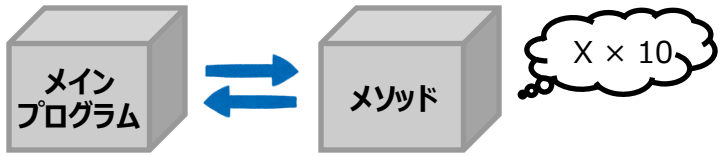
◆変数
・特定の範囲内で変動する値
・変数は「宣言」「代入」「参照」ができる
・決められた型以外のものは入れられない

◆データ型
・変数に格納するデータの性質を定義するもの
・データ型は変数の宣言時に変数名の前に記載
・データ型には「整数」「小数」「文字」などが存在する

◆配列
・同じ名前を持った複数の変数の塊
・「添え字」という番地によって箱を指定する
・添え字は変数で指定することができる



◆メソッド
・ある処理をまとめたもの
・「引数」を受け取り「戻り値」を返す
・メソッドのデータ型と「return」のデータ型は等しい



最後までご覧いただきありがとうございました。
「クラス」「オブジェクト指向」をテーマにしたコンテンツも作成予定です。
そちらもあわせてご一読いただけましたら幸いです。

INTLOOPについて

さまざまな経営課題の解決を支援するコンサルティング事業を主軸に、テクノロジーを駆使しビジネスモデルの変革を目指すデジタルトランスフォーメーション事業、 システムの開発・導入を支援するテクノロジーソリューション事業、専門性の高い人材をご紹介する人材ソリューション事業の4事業を柱に事業を展開。

常にお客様の視点に立つことを第一義に考え、お客様の課題に対して最適なソリューションを提供し続けています。

お問合せ

下記フォームよりお問合せください。

https://www.intloop.com/contact/general/

記載の企業ロゴデザインについて

記載している企業のロゴ、商標は企業が提示しているガイドラインを確認したうえで記載しています。
デザイン、商標についての著作権は、それぞれの企業に帰属しています。

免責事項

この文書に記載されている情報は一般的なものであり、特定の個人や組織に対するアドバイスを提供するものではありません。掲載情報の正確さについてできる限りの努力をしていますが、その正確性や適切性を保証するものではありません。
何らかの行動をとられる場合は、本資料の情報のみを根拠とせず、専門家による適切な分析・アドバイスをもとにご判断ください。当資料を用いて行う一切の行為、被った損害・損失に対しては当社は一切の責任を負いかねます。予めご了承ください。
当資料の著作権は当社にあります。当資料の転載、流用、転売など、ダウンロードされたご本人様以外のご利用は固くお断りさせていただきます。