値渡し

べき乗（x の n 乗）を求めるメソッドを作成しましょう。x が **double** 型で、n が **int** 型であれば、プログラムは **List 7-5** のようになります。

List 7-5 Chap07/Power.java

```
// べき乗を求める

import java.util.Scanner;

class Power {

   //--- xのn乗を返す ---//
   static double power(double x, int n) {
      double tmp = 1.0;

      for (int i = 1; i <= n; i++)
         tmp *= x;    // tmpにxを掛ける
      return tmp;
   }

   public static void main(String[] args) {
      Scanner stdIn = new Scanner(System.in);

      System.out.println("aのb乗を求めます。");
      System.out.print("実数a：");  double a = stdIn.nextDouble();
      System.out.print("整数b：");  int b = stdIn.nextInt();

      System.out.println(a + "の" + b + "乗は" + power(a, b) + "です。");
   }
}
```

実行例

```
aのb乗を求めます。
実数a：5.5
整数b：3
5.5の3乗は166.375です。
```

n が整数ですから、x を n 回掛け合わせた値が x の n 乗です。メソッド *power* では、1.0 で初期化された変数 *tmp* に対して、x の値を n 回掛けています。**for** 文が終了したときの *tmp* の値が、x の n 乗です。

＊

Fig.7-5 に示すように、仮引数 x は実引数 a の値で初期化され、仮引数 n は実引数 b の値で初期化されます。このような、メソッド間で引数として《値》がやり取りされるメカニズムは、**値渡し**（*pass by value*）と呼ばれます。

重要 メソッド間の引数の受渡しは、値渡しによって行われる。

そのため、呼び出された側のメソッド *power* の中で、受け取った仮引数の値を変更したとしても、呼出し側の実引数が影響を受けることはありません。

本のコピーをとって、それに赤鉛筆などで何か書き込んでも、もとの本は、何の影響も受けません。それと同じ原理です。メソッドの中では、仮引数の値を自由気ままに“いじくって”構わないのです。

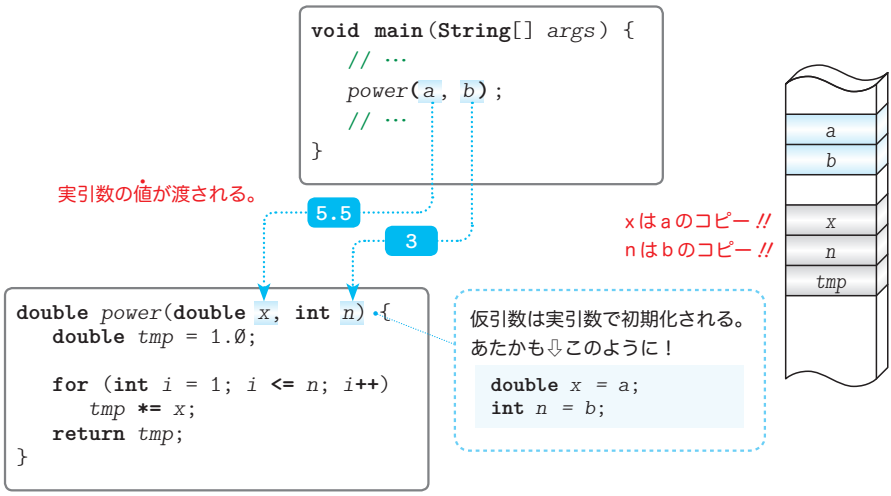

Fig.7-5　メソッド呼出しにおける引数の授受（値渡し）

x の値を *n* 回だけ掛け合わせる処理を、*n* の値を 5，4，…，1 とカウントダウンしていきながら行ってみましょう。

そのように書きかえたメソッド *power* が **List 7-6** です。

▶ クラスの宣言や **main** メソッドなどは省略していますので、**List 7-5** にならって補完しましょう。

List 7-6　Chap07/Power2.java

```
//--- xのn乗を返す ---//
static double power(double x, int n) {
    double tmp = 1.0;

    while (n-- > 0)
        tmp *= x;    // tmpにxを掛ける
    return tmp;
}
```

繰返しを制御するための変数 *i* が不要となり、メソッドはコンパクトになっています。

重 要　値渡しのメリットを活かすと、メソッドはコンパクトになる。

メソッド *power* の実行が終了するときの仮引数 *n* の値は 0 となりますが、呼出し側である **main** メソッドの実引数 *b* の値が 0 になることはありません。

演習 7–4

1 から *n* までの全整数の和を求めて返却するメソッドを作成せよ。

```
int sumUp(int n)
```