プログラミング基礎

第4回

連絡：小テスト実施

- 10月20日（木）
- 範囲：1回（コマンドライン引数）～4回（メソッド2）
- 参照不可
- 席指定

  （Webにも同内容掲載済み）

メソッド

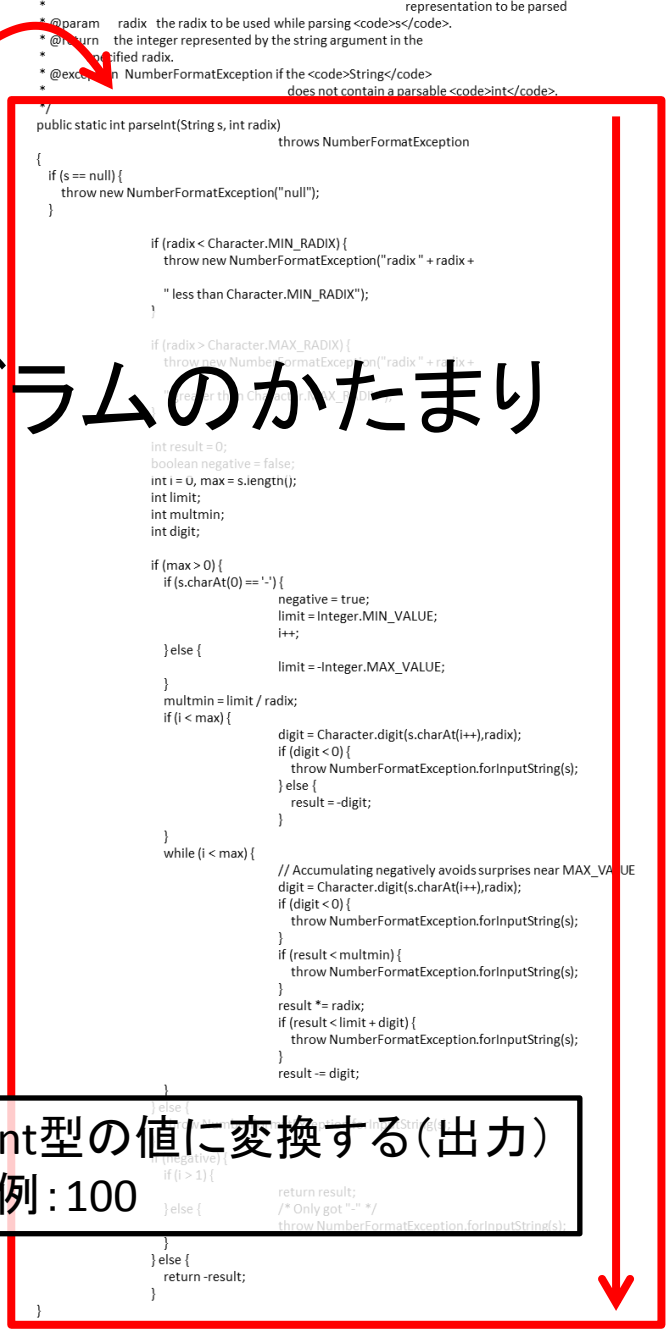
- 特定の機能を実現するプログラムのかたまり

①**引数**に文字列を渡すと（入力）…
例："100"

price = Integer.parseInt(args[0]);

③変換した値が代入演算子で
price変数に記憶される

②int型の値に変換する（出力）
例：100



```
                                                representation to be parsed
 * @param     radix   the radix to be used while parsing <code>s</code>.
 * @return    the integer represented by the string argument in the
 *            specified radix.
 * @exception NumberFormatException if the <code>String</code>
 *                                  does not contain a parsable <code>int</code>.
 */
public static int parseInt(String s, int radix)
                                  throws NumberFormatException
{
  if (s == null) {
    throw new NumberFormatException("null");
  }

                if (radix < Character.MIN_RADIX) {
                  throw new NumberFormatException("radix " + radix +

                  " less than Character.MIN_RADIX");
                }

                if (radix > Character.MAX_RADIX) {
                  throw new NumberFormatException("radix " + radix +
                  " greater than Character.MAX_RADIX");
                }

                int result = 0;
                boolean negative = false;
                int i = 0, max = s.length();
                int limit;
                int multmin;
                int digit;

                if (max > 0) {
                  if (s.charAt(0) == '-') {
                                  negative = true;
                                  limit = Integer.MIN_VALUE;
                                  i++;
                  } else {
                                  limit = -Integer.MAX_VALUE;
                  }
                  multmin = limit / radix;
                  if (i < max) {
                                  digit = Character.digit(s.charAt(i++),radix);
                                  if (digit < 0) {
                                    throw NumberFormatException.forInputString(s);
                                  } else {
                                    result = -digit;
                                  }
                  }
                  while (i < max) {
                                  // Accumulating negatively avoids surprises near MAX_VALUE
                                  digit = Character.digit(s.charAt(i++),radix);
                                  if (digit < 0) {
                                    throw NumberFormatException.forInputString(s);
                                  }
                                  if (result < multmin) {
                                    throw NumberFormatException.forInputString(s);
                                  }
                                  result *= radix;
                                  if (result < limit + digit) {
                                    throw NumberFormatException.forInputString(s);
                                  }
                                  result -= digit;
                  }
                } else {
                  throw NumberFormatException.forInputString(s);
                }
                if (negative) {
                  if (i > 1) {
                                  return result;
                  } else {        /* Only got "-" */
                                  throw NumberFormatException.forInputString(s);
                  }
                } else {
                  return -result;
                }
}

/**
 * Parses the string argument as a signed decimal integer. The
 * characters in the string must all be decimal digits, except that
 * the first character may be an ASCII minus sign <code>'-'</code>
 * (<code>'&amp;#92;u002D'</code>) to indicate a negative value. The resulting
 * integer value is returned, exactly as if the argument and the radix
```

これまで

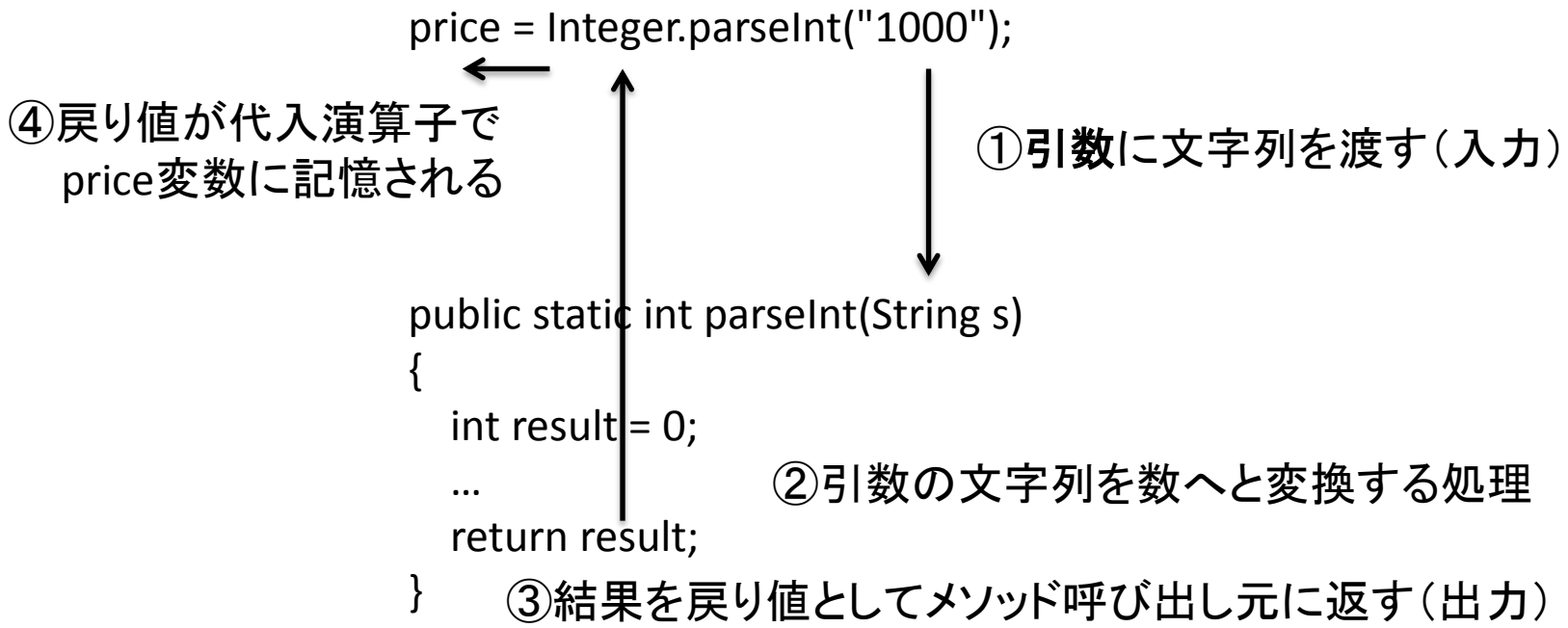
- メソッドの計算結果はすべて画面に表示

```
static void printTax(int x)
{
        int pricewithtax;
        pricewithtax = (int)(x * 1.05);
        System.out.println("税込:" + pricewithtax + "円");
}
```

今回

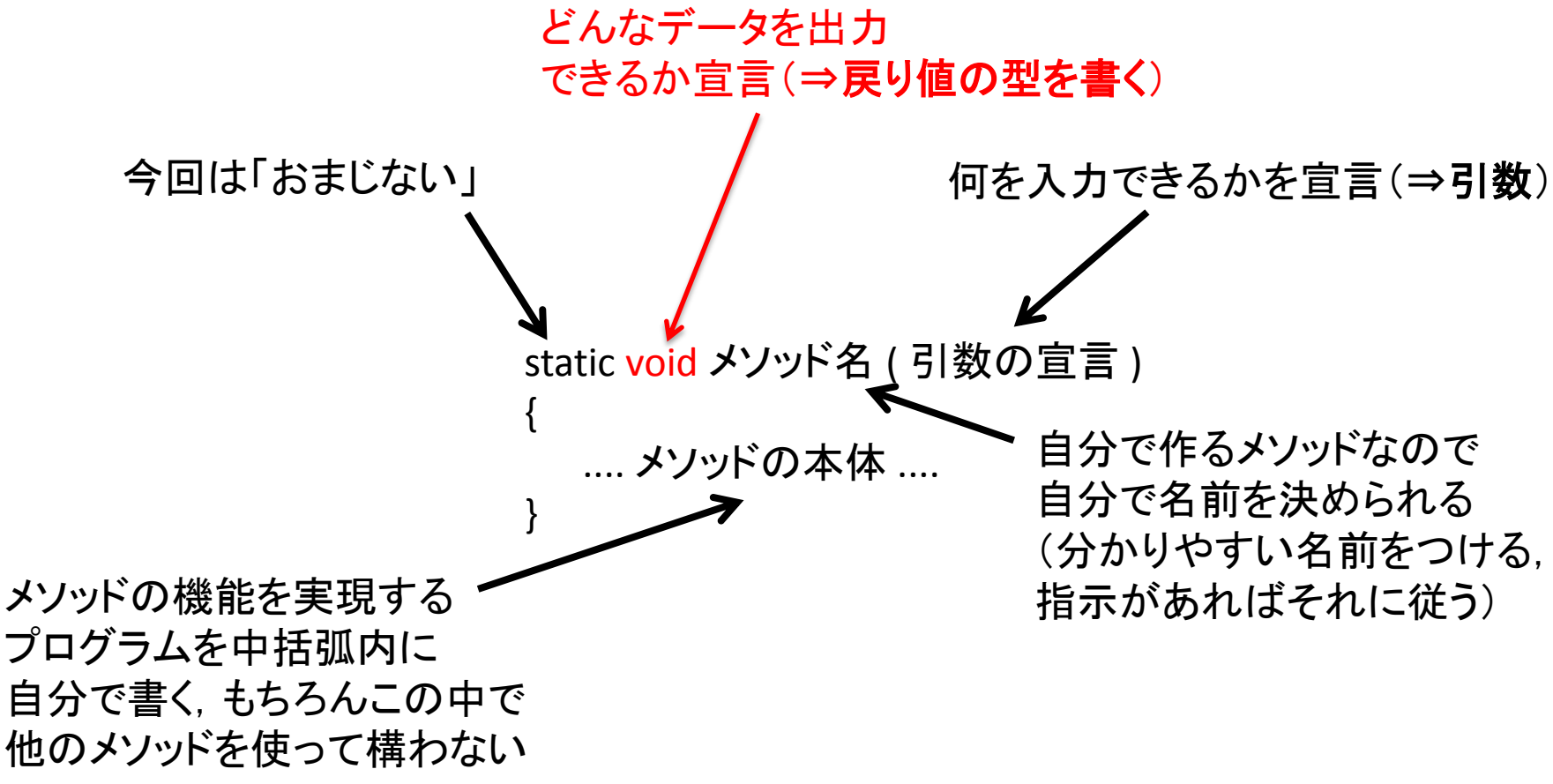
- メソッドの計算結果をプログラム内で利用したい
  ⇒戻り値の利用

具体例：



メソッドの書き方

- Web資料より引用



戻り値

具体例：

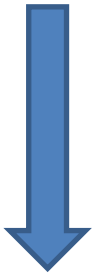
「parseIntメソッドはint型の値を戻り値として呼び出し元へ返します（出力します）」

```
public static int parseInt(String s)
{
    int result = 0;
    …
    return result;
}
```

return x; で，xを戻り値として呼び出し元へ返し，**メソッドを終了する**

戻り値

```
price = Integer.parseInt("1000");
```



戻り値によって，メソッド呼出し後，
メソッドが計算結果に置き換わる
と考えてよい

```
price = 1000;
```

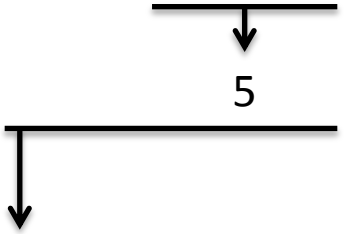
戻り値

- よって次のような書き方も可能

```
//2つの値から大きい方を返すメソッド
static int max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

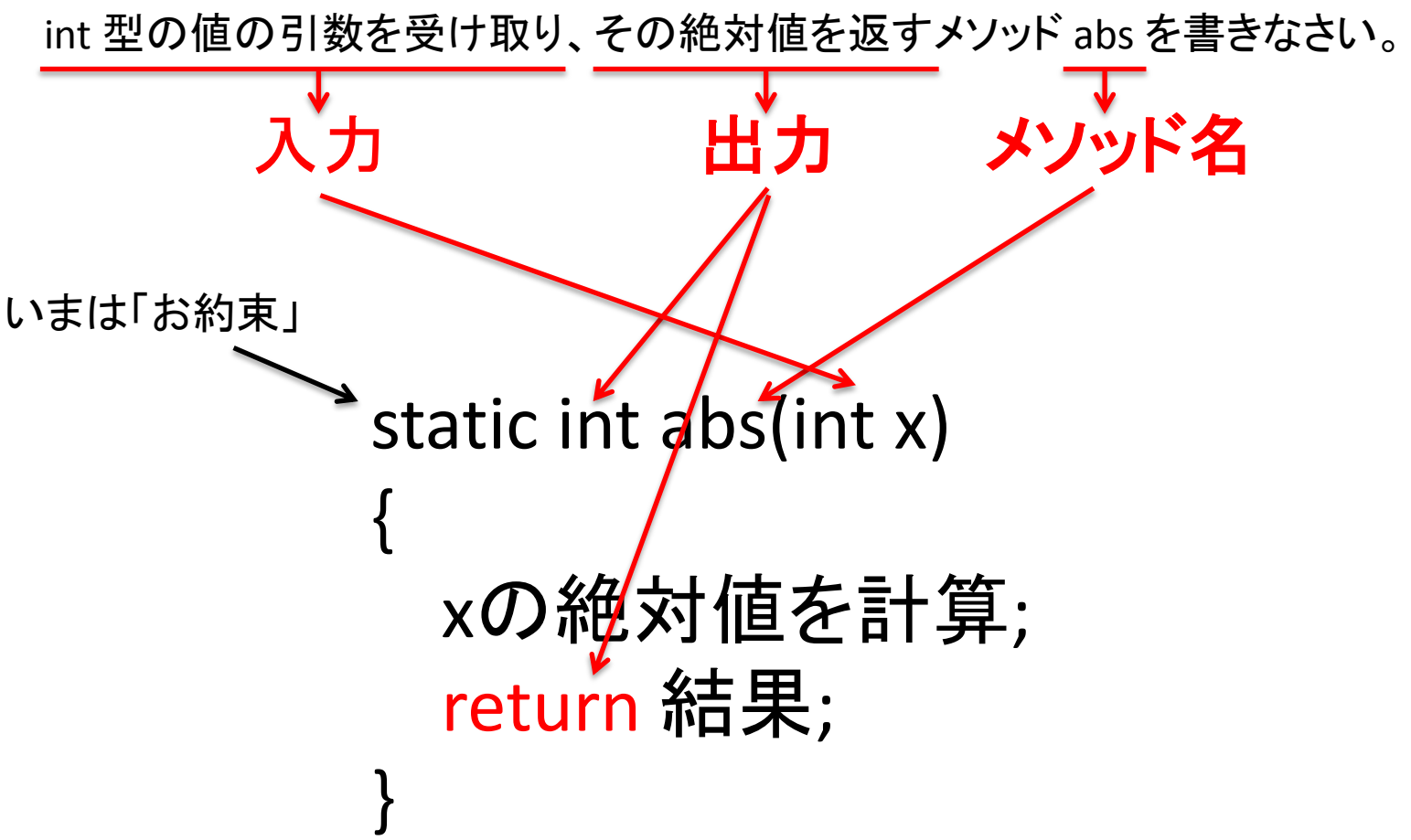
```
int a = 1;
int b = 5;
int c = 3;
int max;

max = max(a, max(b, c));
```



max = 5;

メソッドまとめ


int 型の値の引数を受け取り、その絶対値を返すメソッド abs を書きなさい。
入力
出力
メソッド名
いまは「お約束」
static int abs(int x)
{
xの絶対値を計算;
return 結果;
}

演習

- 値段（int型の値）を引数として，その税込値段を返すcalcTaxメソッドを作成せよ．また，calcTaxメソッドを利用し以下のプログラムを作成せよ

実行例：