・戻り値のないラムダ式もあります。
　例えば、インタフェースの抽象メソッドが void 型なら、ラムダ式も値を返しません。

```
str -> System.out.println(str)          //void display(String str);
```

String を挿入します

ブロック

・本体に、いくつかの文を書きたい場合はブロックにします。

```
book -> {                                // String item(Book book);
    System.out.println(book.title());
    System.out.println(book.price());
    return  book.author();
}
```

　この例のラムダ式は、Book 型のオブジェクト book を引数に受け取り、そのタイトルと価格を表示し、最後に著者名を戻り値として返すように、メソッドを実装します。
　戻り値がある時は、ブロックで最後の文は return 文にする必要があります。

・文は 1 つでもブロックにします。

```
a -> { return a<2000; }                  // boolean test(int a);
```

　本体の末尾にセミコロン (;) を書くと、それは文になり、1 つであってもブロックにする必要があります。値を返す場合は、return を付けます。

変数

・フィールド変数をラムダ式の中で使用できます。
・ローカル変数は、final の付いた変数であればラムダ式の中で使うことができます。

```
final int max = 1000;
int zei = zeigaku( gaku -> gaku<max ? 0.15 : 0.35 );   // double taxRate(int gaku);
```

・final が付かない変数は、**実質的に final** ならラムダ式の中で使うことができます。