Java

参照型の構造とメソッド作成

17時間目

オブジェクト指向とデータ型

Javaの特徴である**オブジェクト指向**

を理解するには

Stringなどのデータ型を深く理解する必要がある

データ型

1 整数：1、2、3、 0 、-1、 － 2 、-3 など

データ型名称	説明
long	億を超える数字。ビッグデータなどに使われる。
int	**30億くらいまでの数字。（最も一般的に使われる。）**
short	127までの数字。（年齢などに使われる。）
byte	????

大

小

2 小数：1.1、0.5、3.14 など

データ型名称	説明
double	**通常はコレしか使わない。**
float	???????

大

小

データ型

3 真偽値：true、false

データ型名称	説明
boolean	**通常はコレしか使わない。**

4 文字：a、あ、 1など

データ型名称	説明
char(キャラ)	**1文字だけのもの（出力する際は、シングルクオテーションで囲む）**

5 文字列：abc、あいう、123など

データ型名称	説明
String	**1文字以上の文字（＝文章）。（出力する際は、ダブルクオテーションで囲む） ※実は、文字列は、1文字ずつのchar型を別々に処理した後に、くっつけて文字列（文章）にしている。**

データ型

プリミティブ型（＝基本データ型＝実体）

参照型（=オブジェクト型）

1 整数

データ型名称
long
int
short
byte

2 小数

データ型名称
double
float

3 真偽値

データ型名称
boolean

4 文字：

データ型名称
char(キャラ)

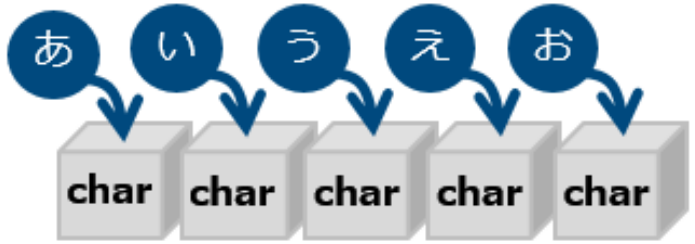
5 文字列

データ型名称
String

参照型(String)の中身の動き

String abc=“あいうえお”;

① char型でそれぞれ別の箱に代入（＝格納）

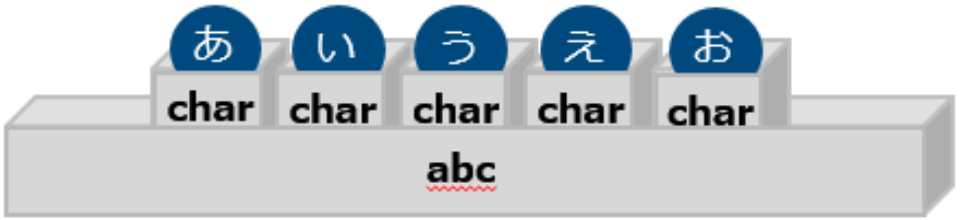


② コピー（=メモリに保存）

プリミティブ型
（＝実体）の処理

System.out.println(abc);

③ 配列として格納&各文字を連結

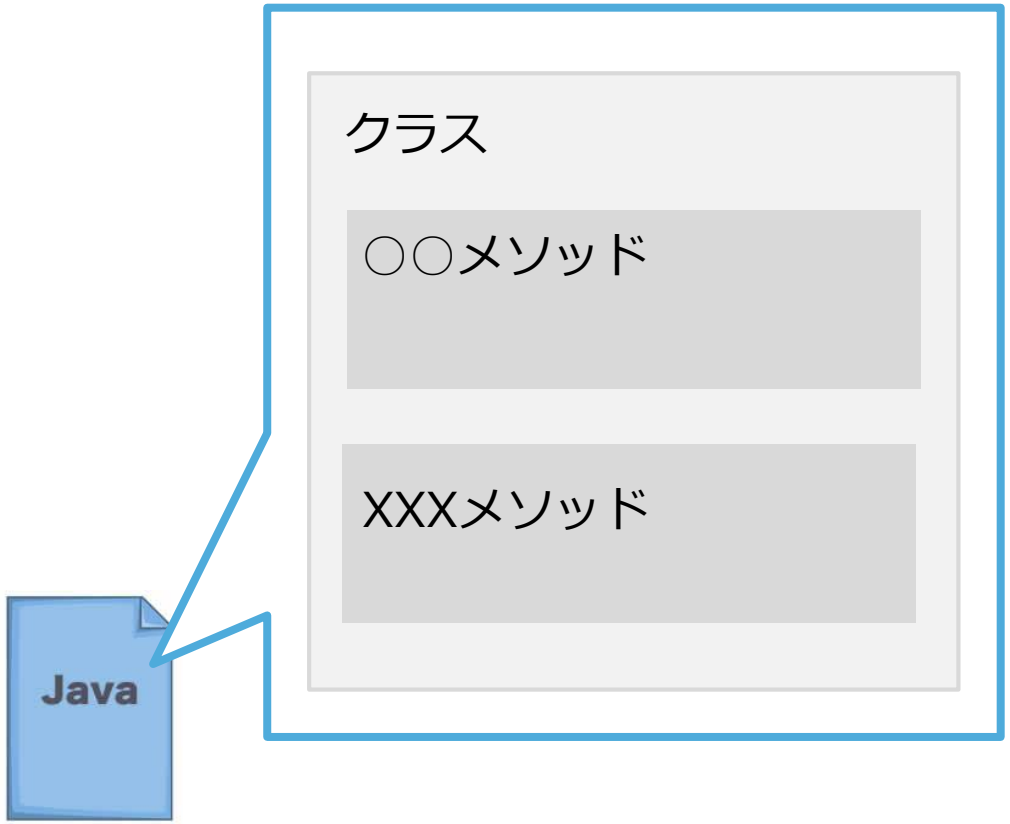


④ 連結して処理(出力)、
保存したモノを削除

あいうえお が出力（=表示）

参照側型
（＝オブジェクト型）
の処理

メソッドとは

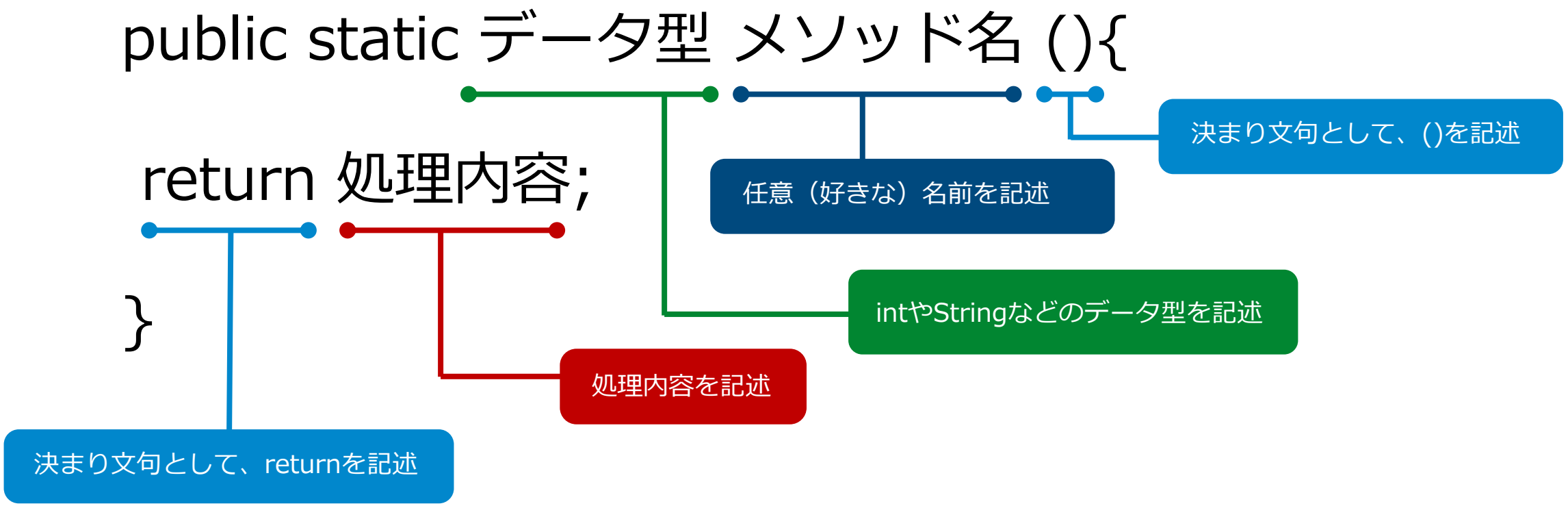


abc.java
（Javaファイル）

クラスの中には、
メソッドと呼ばれる処理が複数ある

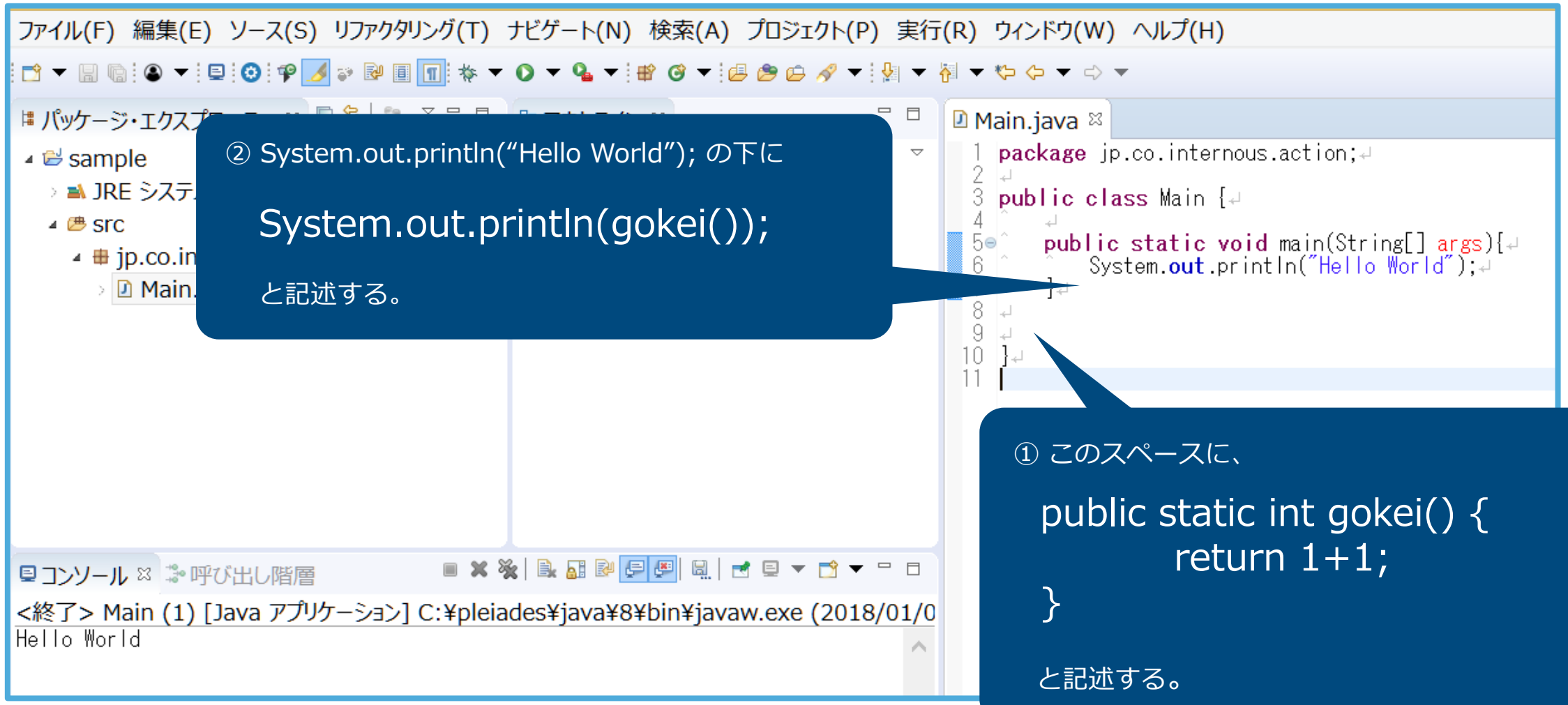
クラスとは、
JavaScriptやPHPで勉強した関数と似た書き方をする

メソッドの書き方


public static データ型 メソッド名 (){
return 処理内容;
}
決まり文句として、()を記述
任意（好きな）名前を記述
intやStringなどのデータ型を記述
処理内容を記述
決まり文句として、returnを記述

メソッドを記述

前のレッスンで作成した「Main.java」の中に 次のコードを記述しましょう。

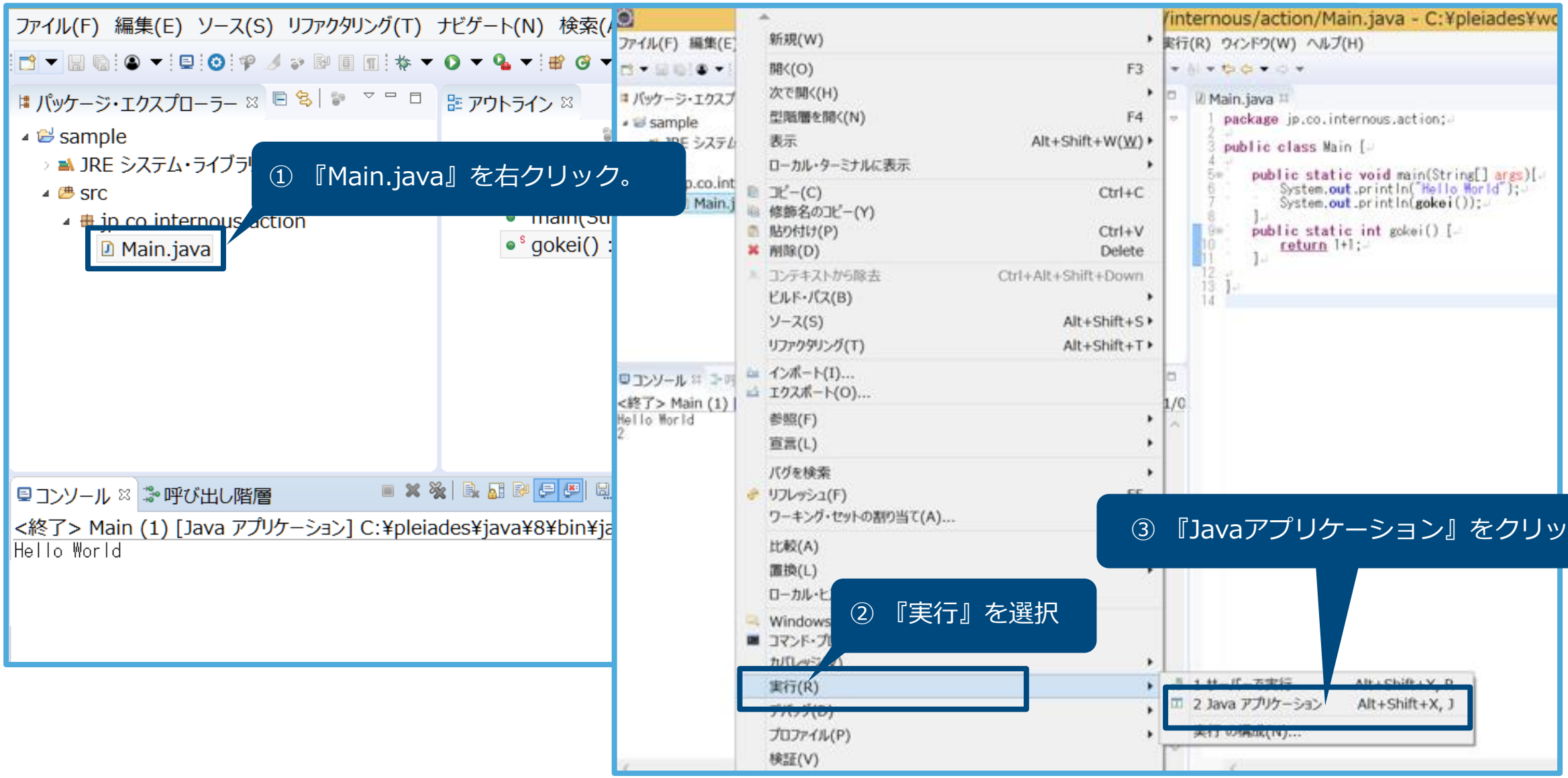


メソッドを記述

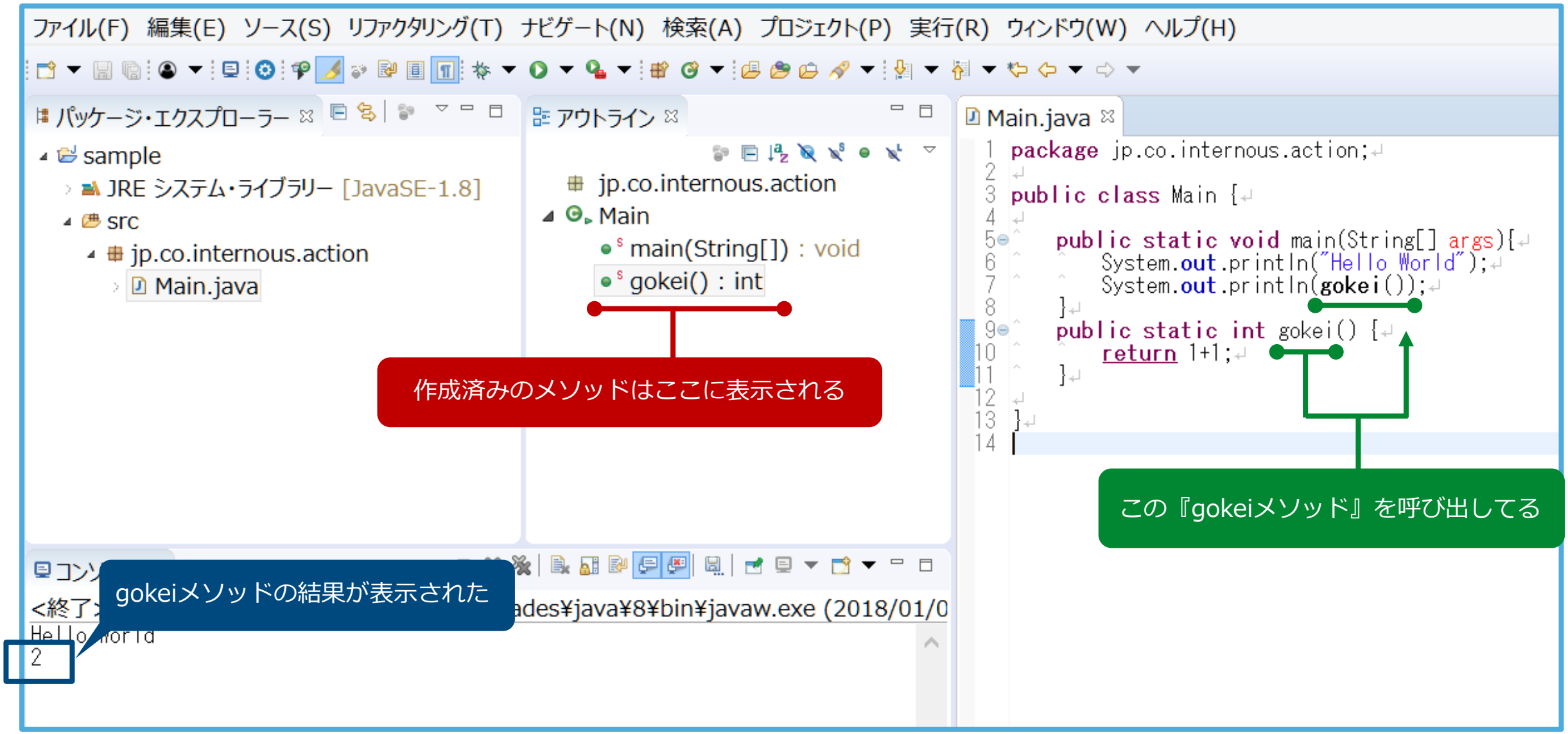
Main.java

```
1  package jp.co.internous.action;
2
3  public class Main {
4
5      public static void main(String[] args){
6          System.out.println("Hello World");
7          System.out.println(gokei());
8      }
9      public static int gokei() {
10         return 1+1;
11     }
12
13 }
14
```

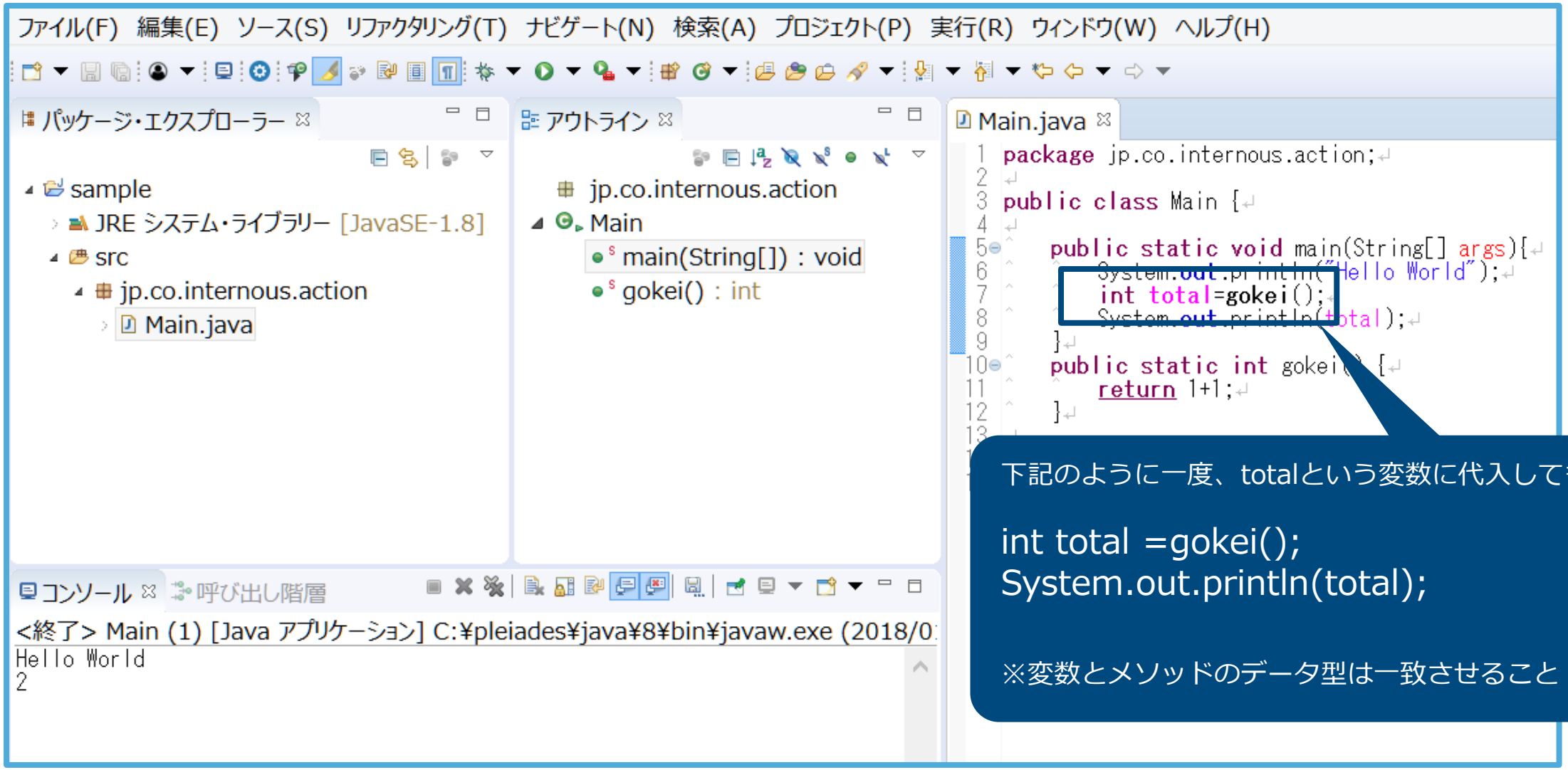
実行


ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(
パッケージ・エクスプローラー
アウトライン
sample
JRE システム・ライブラ
src
Main.java
gokei() :
① 『Main.java』を右クリック。
コンソール
呼び出し階層
<終了> Main (1) [Java アプリケーション] C:¥pleiades¥java¥8¥bin¥ja
Hello World
新規(W)
開く(O) F3
次で開く(H)
型階層を開く(N) F4
表示 Alt+Shift+W(W)
ローカル・ターミナルに表示
コピー(C) Ctrl+C
修飾名のコピー(Y)
貼り付け(P) Ctrl+V
削除(D) Delete
コンテキストから除去 Ctrl+Alt+Shift+Down
ビルド・パス(B)
ソース(S) Alt+Shift+S
リファクタリング(T) Alt+Shift+T
インポート(I)...
エクスポート(O)...
参照(F)
宣言(L)
バグを検索
リフレッシュ(F)
ワーキング・セットの割り当て(A)...
比較(A)
置換(L)
実行(R)
プロファイル(P)
検証(V)
② 『実行』を選択
/internous/action/Main.java - C:¥pleiades¥wo
実行(R) ウィンドウ(W) ヘルプ(H)
Main.java
package jp.co.internous.action;
public class Main {
public static void main(String[] args){
System.out.println("Hello World");
System.out.println(gokei());
}
public static int gokei() {
return 1+1;
}
}
2 Java アプリケーション Alt+Shift+X, J
③ 『Javaアプリケーション』をクリック。

メソッドの実行結果


ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
パッケージ・エクスプローラー
sample
JRE システム・ライブラリー [JavaSE-1.8]
src
jp.co.internous.action
Main.java
アウトライン
jp.co.internous.action
Main
main(String[]) : void
gokei() : int
作成済みのメソッドはここに表示される
Main.java
1 package jp.co.internous.action;
3 public class Main {
5 public static void main(String[] args){
6 System.out.println("Hello World");
7 System.out.println(gokei());
8 }
9 public static int gokei() {
10 return 1+1;
11 }
13 }
この『gokeiメソッド』を呼び出してる
gokeiメソッドの結果が表示された
<終了>
ades¥java¥8¥bin¥javaw.exe (2018/01/0
Hello World
2

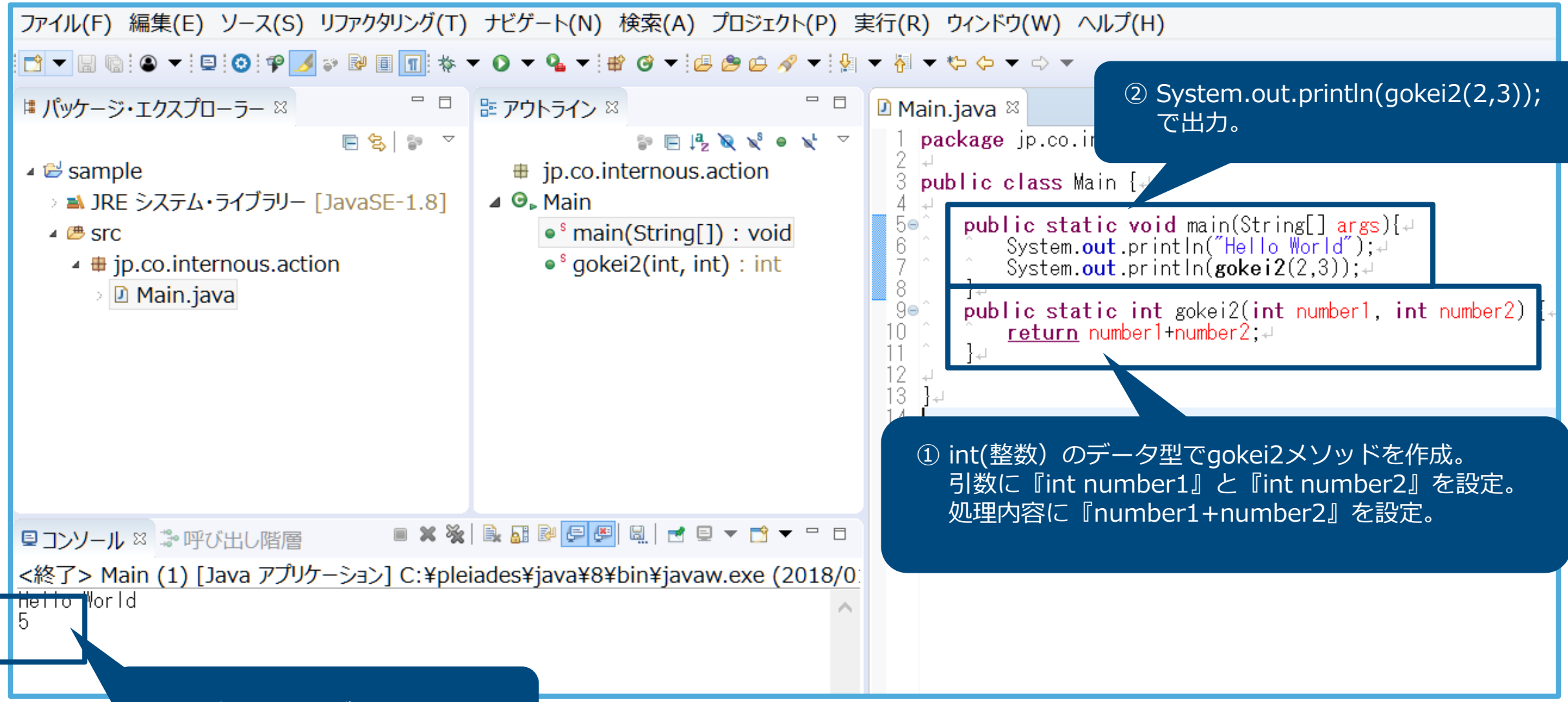
別の方法でもメソッドを出力可能



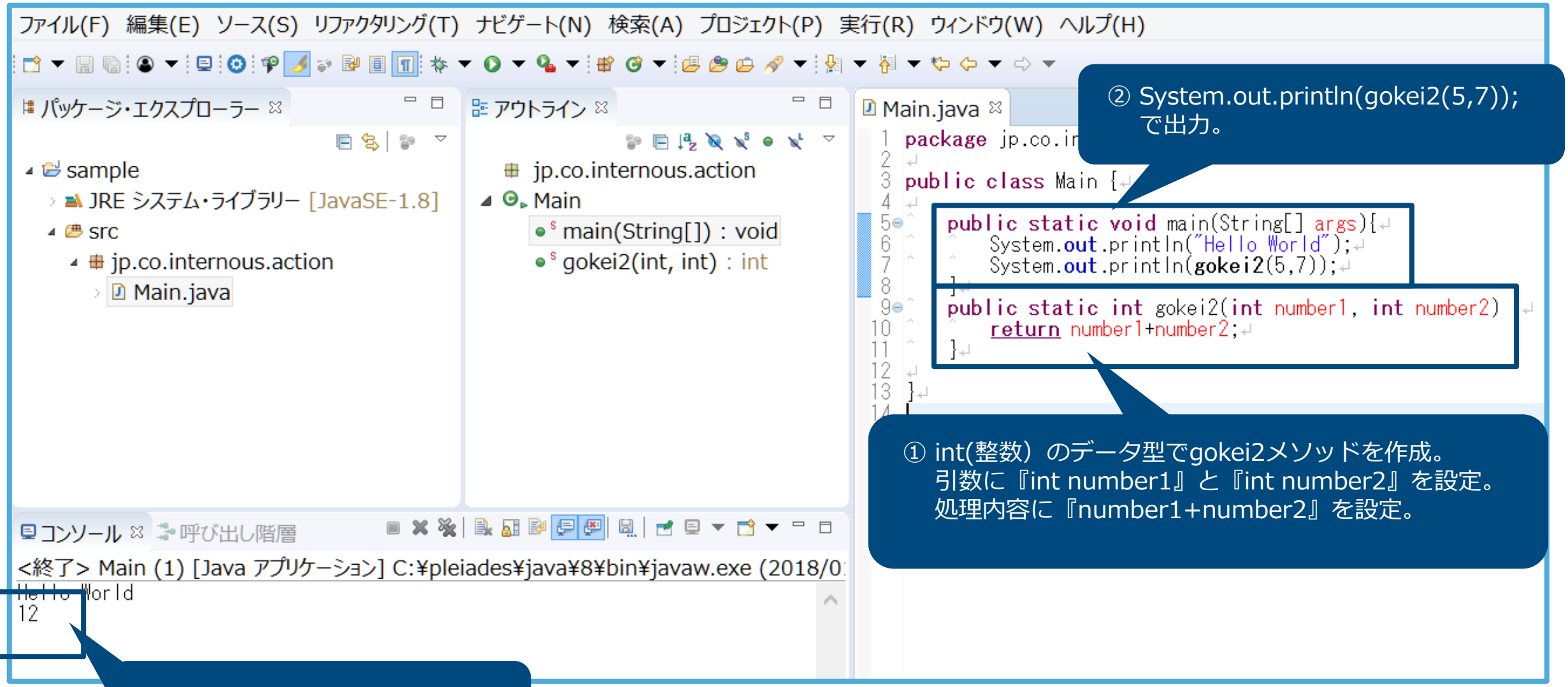
いろいろなメソッド

いくつかメソッドの事例を見てみましょう。

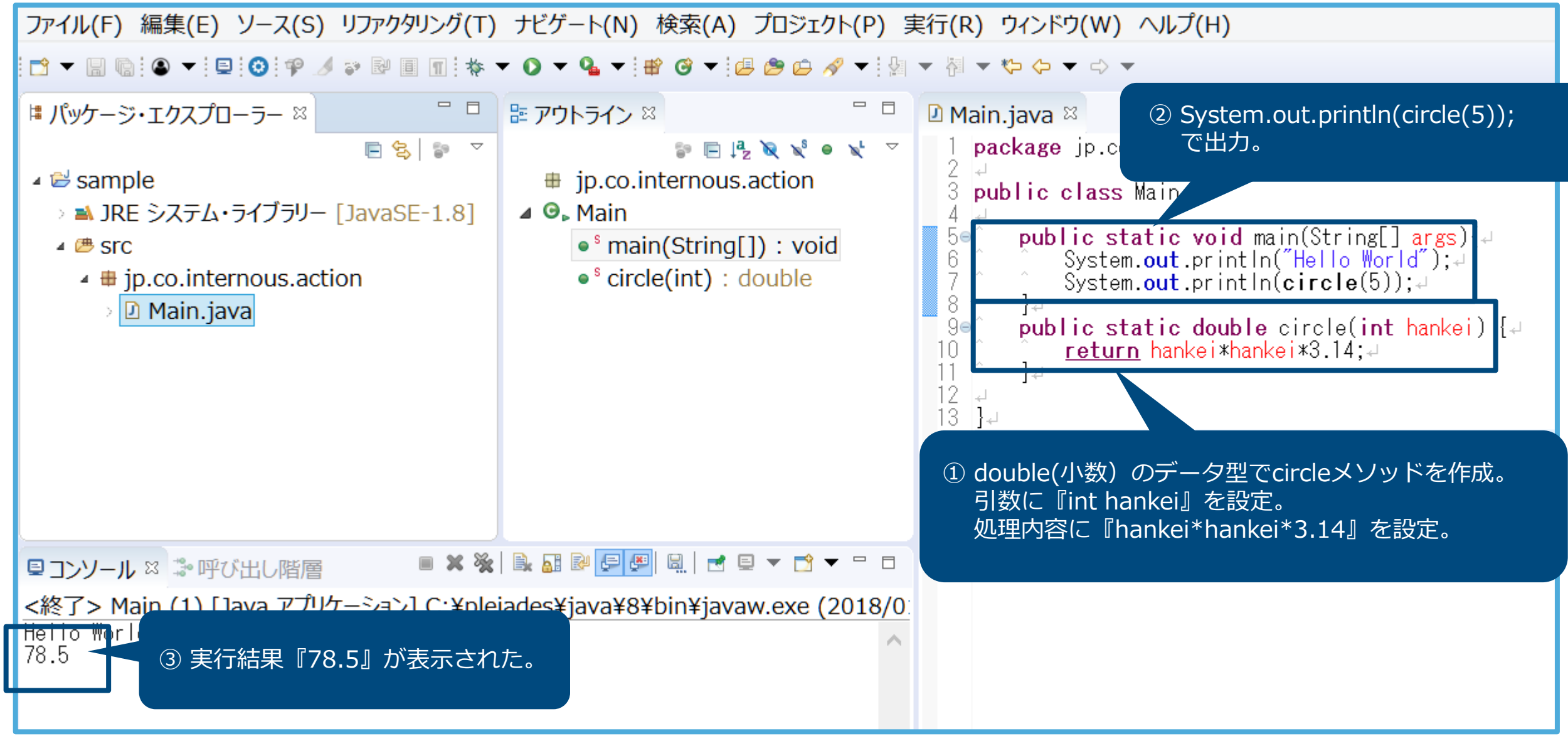
参考　2つの数の合計値①



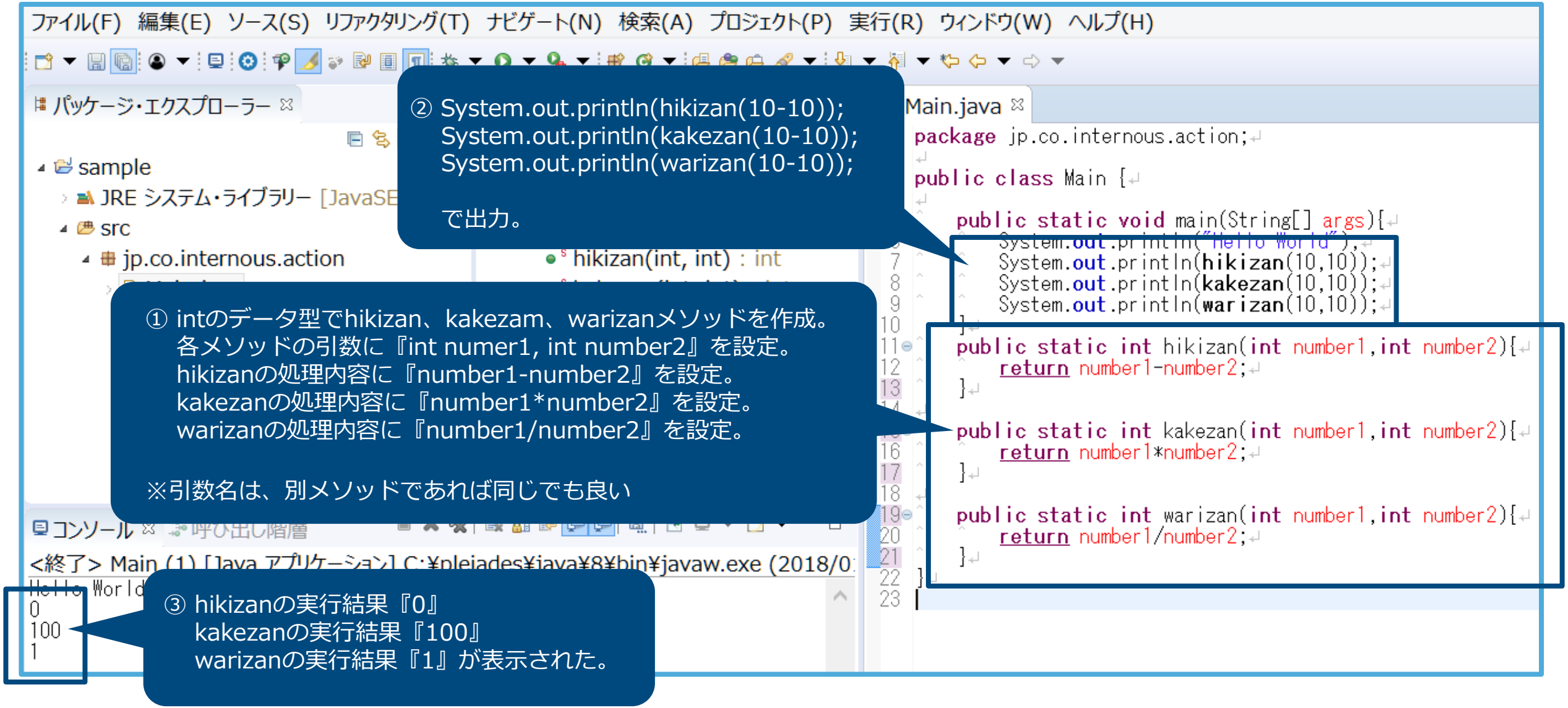
参考　2つの数の合計値②



参考　円の面積


ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
パッケージ・エクスプローラー
sample
JRE システム・ライブラリー [JavaSE-1.8]
src
jp.co.internous.action
Main.java
アウトライン
jp.co.internous.action
Main
main(String[]) : void
circle(int) : double
Main.java
1 package jp.c
2
3 public class Main
4
5 public static void main(String[] args)
6 System.out.println("Hello World");
7 System.out.println(circle(5));
8 }
9 public static double circle(int hankei) {
10 return hankei*hankei*3.14;
11 }
12
13 }
② System.out.println(circle(5));
で出力。
① double(小数）のデータ型でcircleメソッドを作成。
引数に『int hankei』を設定。
処理内容に『hankei*hankei*3.14』を設定。
コンソール
呼び出し階層
<終了> Main (1) [Java アプリケーション] C:¥pleiades¥java¥8¥bin¥javaw.exe (2018/0
Hello Worl
78.5
③ 実行結果『78.5』が表示された。

参考　複数のメソッド


② System.out.println(hikizan(10-10));
System.out.println(kakezan(10-10));
System.out.println(warizan(10-10));
で出力。
① intのデータ型でhikizan、kakezam、warizanメソッドを作成。
各メソッドの引数に『int numer1, int number2』を設定。
hikizanの処理内容に『number1-number2』を設定。
kakezanの処理内容に『number1*number2』を設定。
warizanの処理内容に『number1/number2』を設定。
※引数名は、別メソッドであれば同じでも良い
③ hikizanの実行結果『0』
kakezanの実行結果『100』
warizanの実行結果『1』が表示された。
0
100
1