

プログラミング基礎 I

第4回

変数、条件分岐、繰り返し（演習）

九州産業大学理工学部

pk@is.kyusan-u.ac.jp

演習回について

- 第4回は演習回
 - 講義回で勉強した内容を演習回で復習する
 - 新たな内容は多くは出ないが、演習問題が多く出題される
- 講義資料や教科書を参考に
 - 解けない場合は1～3回に戻って復習！

変数宣言、型、代入、演算

変数宣言

型 変数名;

型

int 整数

double 小数

String 文字列

代入

変数名 = データ;

変数の初期化

型 変数名 = 値;

演算子(四則演算,余りを求める)

+ - * / %

インクリメント、デクリメント演算子

++ --

複合代入演算子

+= -= *= /= %=

```
int x; //int型の変数 x を宣言
x = 100; // x に100を代入
System.out.println(x); //xを表示
x++;
System.out.println("xの値は " + x );
x += 10;
System.out.println(x);
```

複合代入演算子でxを10増やす

初期化：一行で宣言して代入

```
double d = 3.14; //double型の 変数 d を宣言し3.14を代入
System.out.println(d + "は円周率");
```

```
String s = "Udon"; //String型の 変数 s を宣言,Udonを代入
System.out.println("昼は" + s + "を食べた");
```

文字列はダブルクォートで囲む

文字列を連結するときは + で繋ぐ

条件分岐 (if文)

条件分岐

```
if ( 条件 ) {  
    処理  
}
```

2 条件の分岐 if-else

```
if ( 条件 ) {  
    処理  
} else {  
    処理  
}
```

3 条件以上の分岐

```
if( 条件1 ){  
    処理  
} else if( 条件2 ) {  
    処理  
} else {  
    処理  
}
```

```
if( a >= 3 ) {  
    System.out.println( "aは3以上" );  
}
```

```
if( a == x ) {  
    System.out.println( "aとxは等しい" );  
} else {  
    System.out.println( "aとxは等しくない" );  
}
```

```
if( a < 60 ) {  
    System.out.println( "C" );  
} else if( a < 70 ){  
    System.out.println( "B" );  
} else {  
    System.out.println( "B" );  
}
```

繰り返し(for文、while文)

```
繰り返し for
  for( 初期化 ; 条件 ; 更新 )
  {
      処理
  }
```

```
繰り返し while
  while ( 条件 ) {
      処理
  }
```

```
//1～10まで1ずつ増やしながら，で区切って表示する
for( int i = 1 ; i <= 10 ; i++ ) {
    System.out.print(i+",");
}
```

```
//10から-5まで4ずつ減らしながら * で区切って表示する
for( int d = 10 ; d >= -5 ; d-=4 ) {
    System.out.print(d+"*");
}
```

```
//1から200までの3倍しながら表示する
int x = 1;
while( x <= 200 ) {
    System.out.print(x+" ");
    x *= 3;
}
```

括弧は開いたらすぐに閉じる

```
for ( ) {
```

```
}
```

```
if ( ) {
```

```
}
```

```
while ( ) {
```

```
}
```

まずは、この形を作ってから中身を書いていこう

インデント

- クラス、メソッド、if、for、while、switchなどの、プログラムのブロックの階層をわかりやすくするための「字下げ」のこと
- プログラムの可読性を保つために、正しいインデントでプログラムを書くことは必須

```
public class For2 {  
    public static void main(String[] args) {  
        for(int i = 1; i <= 10; i++) { //1から10  
            System.out.print(i+" ");  
            if (i % 2 == 0) { //もし偶数なら  
                System.out.println("偶数");  
            } else { //それ以外なら  
                System.out.println("奇数");  
            }  
        }  
    }  
}
```

(Javaにおける) 正しいインデント

- { } を記述した場合、そこで改行する
 - else の前の { は例外

```
public class Main03p1_25RS999 {  
    public static void main(String[] args) {  
  
        for(int i = 1; i <= 9; i+=1) {  
            if(i % 2 == 0) {  
                System.out.print("#");  
            } else {  
                System.out.print(i);  
            }  
        }  
        System.out.println();  
    }  
}
```

elseの前の } は例外 →

(Javaにおける) 正しいインデント

- { の次の行は、前の行の開始位置から、Tab文字(Tabのキー) 一文字、字下げをする

```
public class Main03p1_25RS999 { 改行
    Tab public static void main(String[] args) { 改行
        Tab for(int i = 1; i <= 9; i+=1) { 改行
            Tab if(i % 2 == 0) { 改行
                Tab System.out.print("#");
            } else { 改行
                Tab System.out.print(i);
            } 改行
        } 改行
        System.out.println();
    } 改行
} 改行
```

{ で改行したら Tab キーでインデント！

Tab文字について

- インデントで使う文字は大きく分けて2種類
 - Tab文字でインデントするケース
 - 半角スペース(スペース2文字や4文字等)でインデントするケース
 - どの文字でインデントするかは、そのチーム内のルールによる
 - ただし、
タブインデントとスペースインデントは混在してはならない
- Eclipse環境では、Tab文字でインデントする

正しいインデントでプログラムを書くコツ

- エディターの機能をしっかり利用する
- Eclipseでは、
行の最後で改行すれば勝手に正しいインデントになる

```
}  
System.out.println();  
  

```

- インデントが崩れてしまったら、
[Ctrl] + [A] で全選択後に、 **[Ctrl] + [I]** で自動整形
 - うっかり全削除等しないように注意

キーボードから入力を受け取る（数値）

- Scannerを用いることで、キーボードから値を入力することができるようになる 教科書P.32 リスト3.16

```
import java.util.*;

public class Sample0304_Input {
    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        Scanner sc = new Scanner(System.in);
        System.out.print("Please input a number: ");
        int a;
        a = sc.nextInt();
        System.out.println("The number is : " + a);
    }
}
```

Scannerを使う準備 1

Scannerを使う準備 2

キーボードから入力を受け取る（文字列）

- Scannerを用いることで、キーボードから値を入力することができるようになる 教科書P.33 リスト3.17

```
import java.util.*;

public class Sample0304_Input {
    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        Scanner sc = new Scanner(System.in);
        System.out.print("Please input a number: ");
        String s;
        s = sc.next();
        System.out.println("The string is : " + s);
    }
}
```

Scannerを使う準備 1

Scannerを使う準備 2

関係演算子 教科書P46

演算子	数学での書き方	意味	aが5の時の値
$a < 5$	$a < 5$	aは5より小さい、aは5未満	false
$a \leq 5$	$a \leq 5$	aは5以下	true
$a > 5$	$a > 5$	aは5より大きい	false
$a \geq 5$	$a \geq 5$	aは5以上	true
$a == 5$	$a = 5$	aは5と等しい	true
$a != 5$	$a \neq 5$	aは5と等しくない	false

発展：デバッガ（デバッグ機能）の使い方

- デバッガ（Debugger）
 - プログラムのバグなどの原因調査に用いるツール
 - プログラムの任意の場所で一時停止する、ステップ実行で動作を確認、一時停止した際の変数の値を確認、等の機能がある

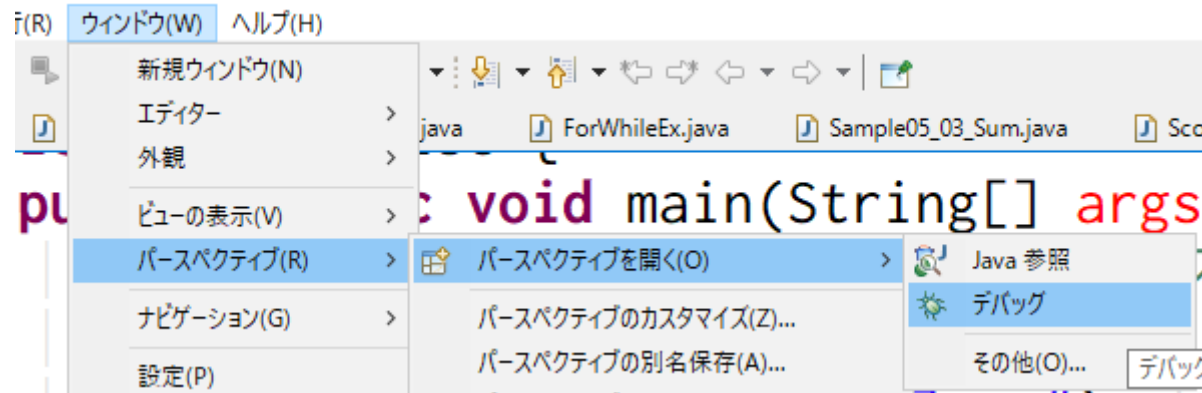
※Eclipseの扱いにまだ慣れていない場合は利用しなくても良い

注意：よく理解した上で利用すること

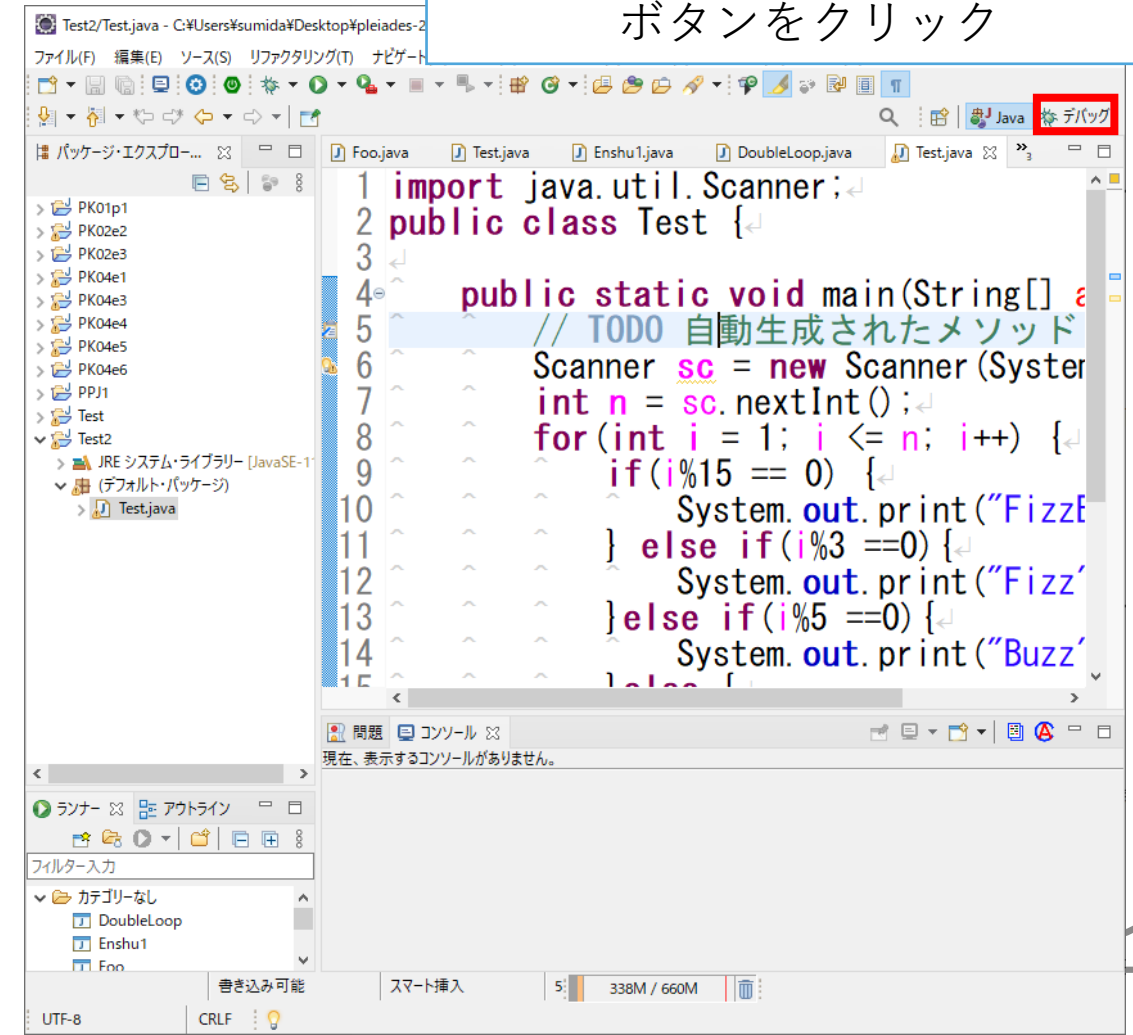
- 複雑なプログラムを開発する際には必須の機能
 - 動画中でも度々使っているが、
使えないと問題を解けないわけではない
 - つまり、本科目では使えなくても一切問題ない
- ただし、使い方を知っておくと今後便利になる
 - 似たような機能は大抵のIDEに実装されているため、
Javaだけでなく他の言語を学ぶ際にも役立つ

デバッグ機能を開く

- 「ウィンドウ」→
「パースペクティブを開く」→
「デバッグ」

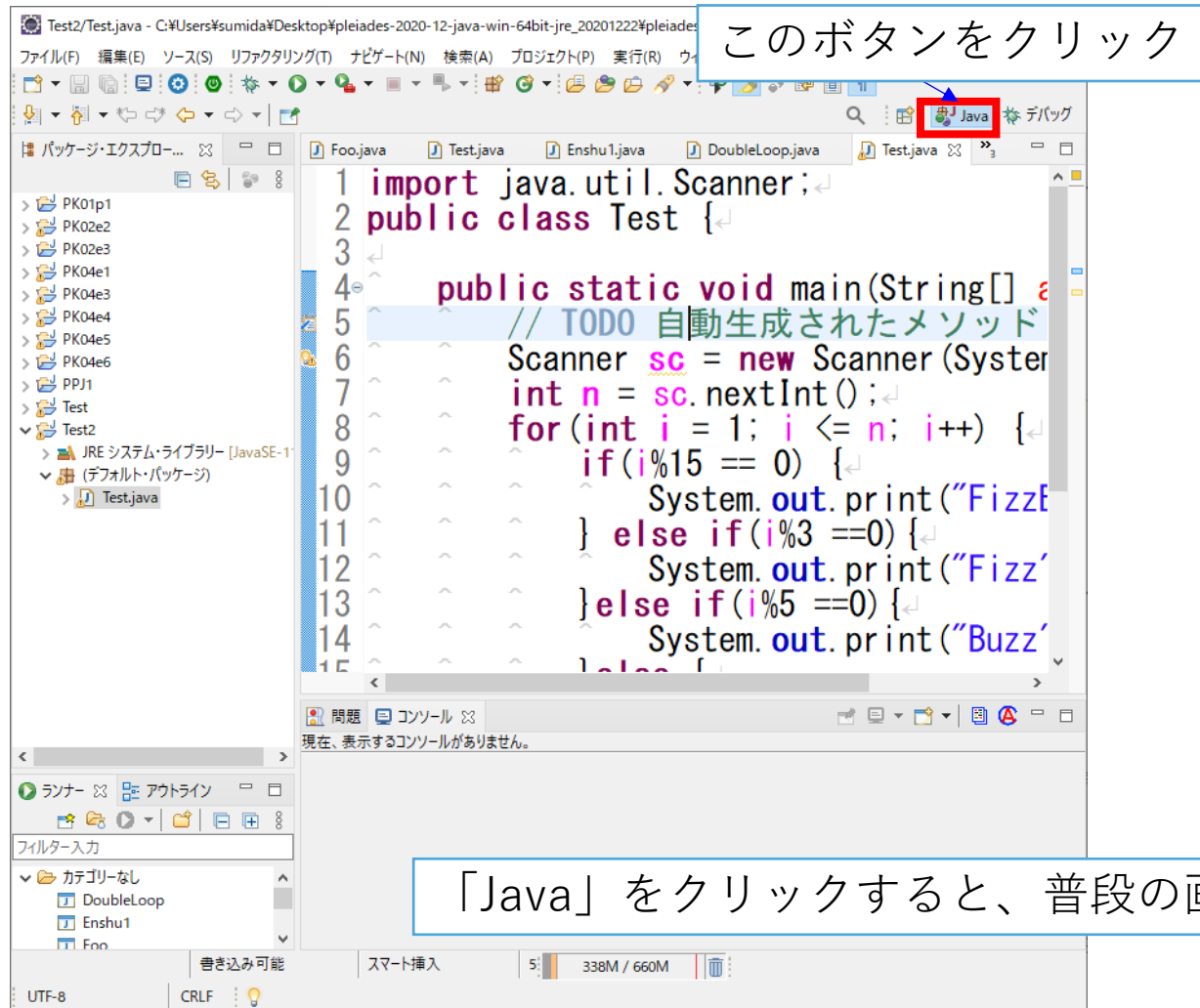


「デバッグ」が既にある場合は、
ボタンをクリック



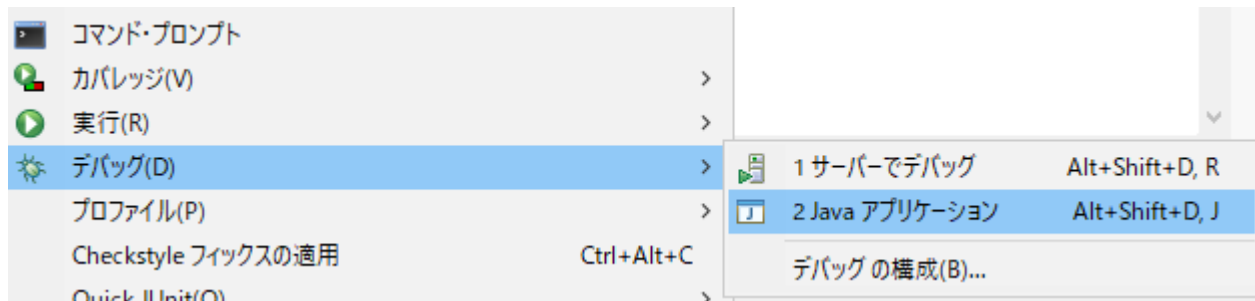
普段の画面への戻し方

- 右上にある「Java」をクリックする



デバッグモードで実行する

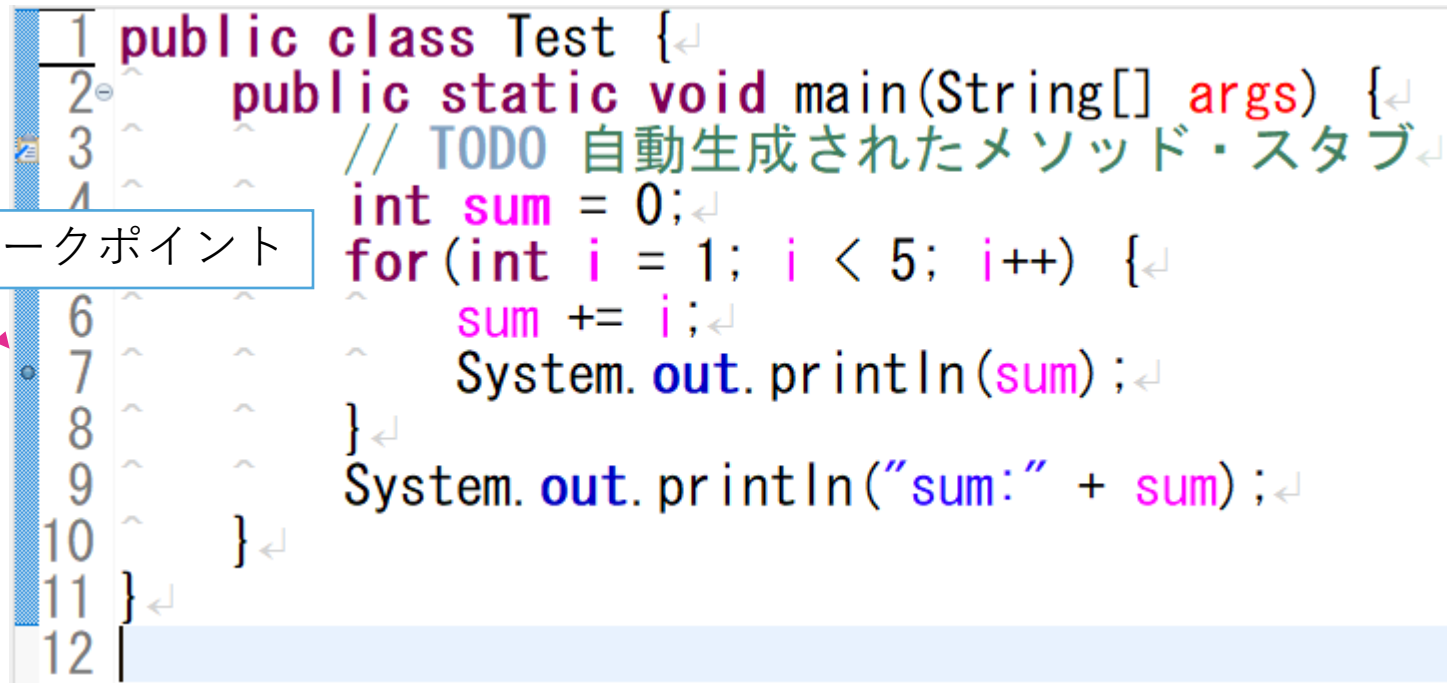
- 「デバッグ」→「Javaアプリケーション」



- このままだと、通常の実行と変わらない

ブレークポイントを設定する

- ブレークポイントを設定することで、設定した箇所でプログラムを一時停止することが出来る
- 止めたい行番号の左をダブルクリック



```
1 public class Test {  
2     public static void main(String[] args) {  
3         // TODO 自動生成されたメソッド・スタブ  
4         int sum = 0;  
5         for(int i = 1; i < 5; i++) {  
6             sum += i;  
7             System.out.println(sum);  
8         }  
9         System.out.println("sum:" + sum);  
10    }  
11 }  
12
```

The image shows a code editor with a Java file. A blue vertical bar on the left side of the editor indicates the line numbers. A small blue circle, representing a breakpoint, is placed on the left margin next to line 7. A red arrow points from a text box to this breakpoint.

青い丸がブレークポイント

ブレークポイント設定中の動作

- ブレークポイントの箇所でプログラムが停止する

```
1 public class Test {  
2     public static void main(String[] args) {  
3         // TODO 自動生成されたメソッド・スタブ  
4         int sum = 0;  
5         for (int i = 1; i < 5; i++) {  
6             sum += i;  
7             System.out.println(sum);  
8         }  
9     }  
10 }  
11  
12
```

青い縦帯の中に→が表示

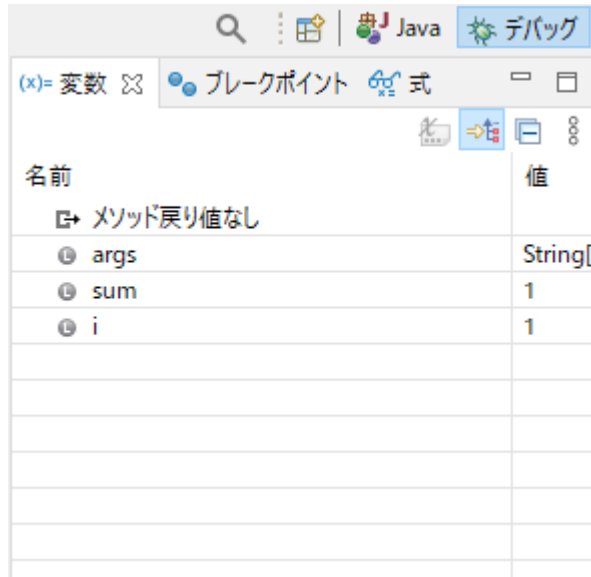
停止行のプログラムの背景色が変わる

「ステップオーバー」と「続行」

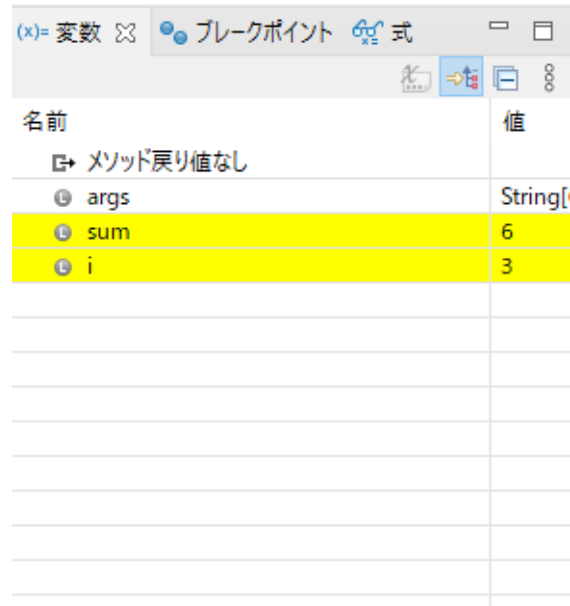
- プログラム停止中に、「F6」キーを押すとステップオーバー
 - F6キーを押すごとに1行ずつプログラムが進む
- プログラム停止中に、「F8」キーを押すと続行
 - プログラムの実行が再開され、次のブレークポイントで停止する

停止中の変数の値の確認

- 「変数」ウィンドウで変数の値を確認できる



名前	値
args	String[]
sum	1
i	1



名前	値
args	String[]
sum	6
i	3

- 通常はコンソール出力しないと確認できない、実行途中の変数の値を確認できる