


情報生命科学演習 Java文法, メソッド

東京大学院新領域・情報生命・特任准教授

加藤 毅

kato-tsuyoshi@k.u-tokyo.ac.jp

やること



- ・ Java文法の基本, メソッドを理解する
- ・ メソッドに対するデバッガの操作を学習する

絶対値の合計を計算するプログラム



Test06a.java

```
public class Test06a {
    public static void main( String[] args ){
        int a, b, c, d, e, f;
        int abs_a, abs_b, abs_c, abs_d, abs_e, abs_f;
        int sum;
        a = 3; b = -1; c = 5; d = -4; e = -7; f = 6;

        if ( a >= 0 ){
            abs_a = a;
        } else {
            abs_a = -a;
        }
        if ( b >= 0 ){
            abs_b = b;
        } else {
            abs_b = -b;
        }
        abs_c, abs_d, abs_e, abs_f も同様に計算する;
        sum = abs_a+abs_b+abs_c+ abs_d+abs_e+abs_f;
        System.out.println( "sum="+sum);
    }
}
```

} 変数 a の値の絶対値を
変数 abs_a に代入

} 変数 b の値の絶対値を
変数 abs_b に代入

なが～い
プログラムになった

同じような文が多い

もっと簡潔に
書けないか？

メソッドを使って簡潔なプログラムに



Test06b.java

```
public class Test06b {
    public static void main( String[] args ){
        int a, b, c, d, e, f;
        int abs_a, abs_b, abs_c, abs_d, abs_e, abs_f;
        int sum;
        a = 3; b = -1; c = 5; d = -4; e = -7; f = 6;

        abs_a = abs(a); abs_b = abs(b); abs_c = abs(c);
        abs_d = abs(d); abs_e = abs(e); abs_f = abs(f);
        sum = abs_a+abs_b+abs_c+ abs_d+abs_e+abs_f;
        System.out.println("sum="+sum);
    }
    public static int abs( int n ){
        int ret;
        if ( n >= 0 ){
            ret = n;
        } else {
            ret = -n;
        }
        return ret;
    }
}
```

変数 a の値の絶対値を
変数 abs_a に代入

変数 b の値の絶対値を
変数 abs_b に代入

} 絶対値を計算する
メソッド

メソッドを使うと
同様なコードを
一つにまとめられる

メソッド, 戻り値, 引数



Test06b.java

```
public class Test06b {
    public static void main( String[] args ){
        int a, b;
        int abs_a, abs_b;
        int sum;
        a = 3; b = -1;

        abs_a = abs(a);
        abs_b = abs(b);
        sum = abs_a+abs_b;
        System.out.println("sum="+sum);
    }
    public static int abs( int n ){
        int ret;
        if ( n >= 0 ){
            ret = n;
        } else {
            ret = -n;
        }
        return ret;
    }
}
```

変数 a の値の絶対値を
変数 abs_a に代入

変数 b の値の絶対値を
変数 abs_b に代入

戻り値の型 メソッド名 引数

メソッドabsは変数retの値を呼び出し元に返す

メソッドの一般形式



Test06b.java

```
public class Test06b {
    public static void main(
        int a, b;
        int abs_a, abs_b;
        int sum;
        a = 3; b = -1;

        abs_a = abs(a);
        abs_b = abs(b);
        sum = abs_a+abs_b;
        System.out.println("sum="+sum);
    }
    public static int abs( int n ){
        int ret;
        if ( n >= 0 ){
            ret = n;
        } else {
            ret = -n;
        }
        return ret;
    }
}
```

```
public static 戻り値の型 メソッド名(引数){
    変数宣言;
    文;
    return 式;
}
```

戻り値の型 メソッド名 引数

メソッドabsは変数retの値を呼び出し元に返す

練習6-b.



Test06b.java

```
public class Test06b {
    public static void main( String[] args ){
        int a, b;
        int abs_a, abs_b;
        int sum;
        a = 3; b = -1;

        abs_a = abs(a);
        abs_b = abs(b);
        sum = abs_a+abs_b;
        System.out.println("sum="+sum);
    }
    public static int abs( int n ){
        int ret;
        if ( n >= 0 ){
            ret = n;
        } else {
            ret = -n;
        }
        return ret;
    }
}
```

左のプログラムを
実行せよ

後でデバッガを
使って動作を
確認します

まとめ

- Java文法の基本, メソッドを理解する

- メソッドに対するデバッガの操作を学習する

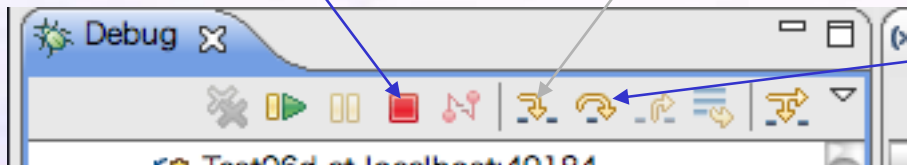
- 「Step Over」ボタンは, メソッドの呼び出しも含めて1行ずつ動作させる

- 「Step Into」ボタンは, メソッドの先頭のコードを実行する手前まで進める

停止ボタン

Step Into ボタン

Step Over ボタン



デバッガ: Step Over のみ使う場合



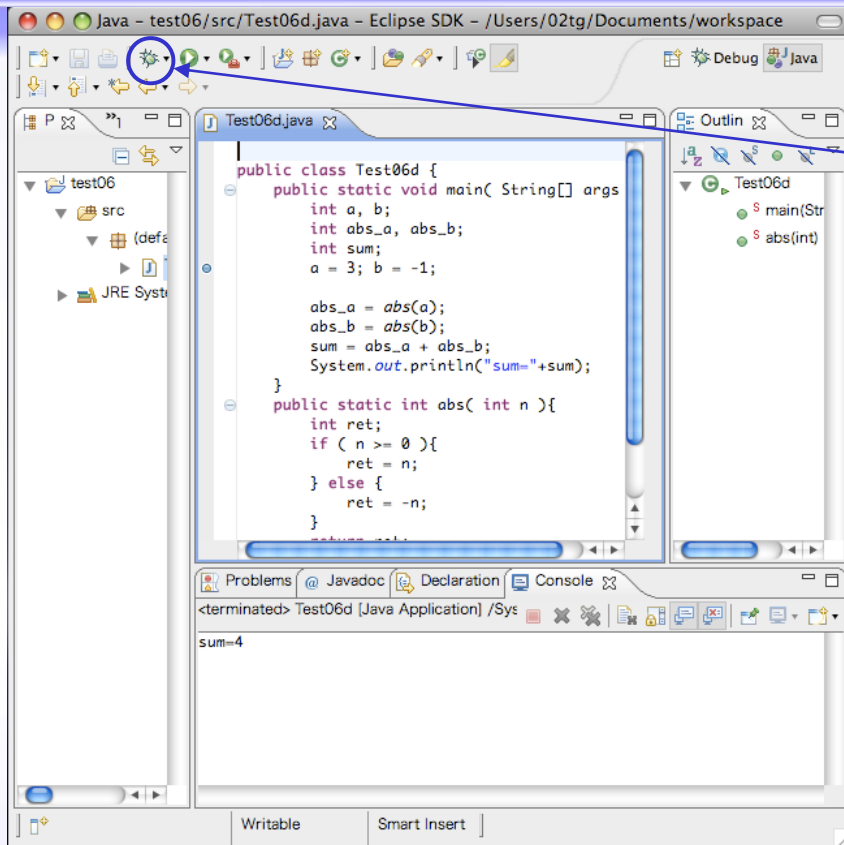
```
public class Test06d {
    public static void main( String[] args
        int a, b;
        int abs_a, abs_b;
        int sum;
        a = 3; b = -1;

        abs_a = abs(a);
        abs_b = abs(b);
        sum = abs_a + abs_b;
        System.out.println("sum="+sum);
    }
    public static int abs( int n ){
        int ret;
        if ( n >= 0 ){
            ret = n;
        } else {
            ret = -n;
        }
    }
}
```

ここをダブルクリックして
ブレークポイントをつける

前回と同様, Step Over ボタンだけで
デバッガを試してみる

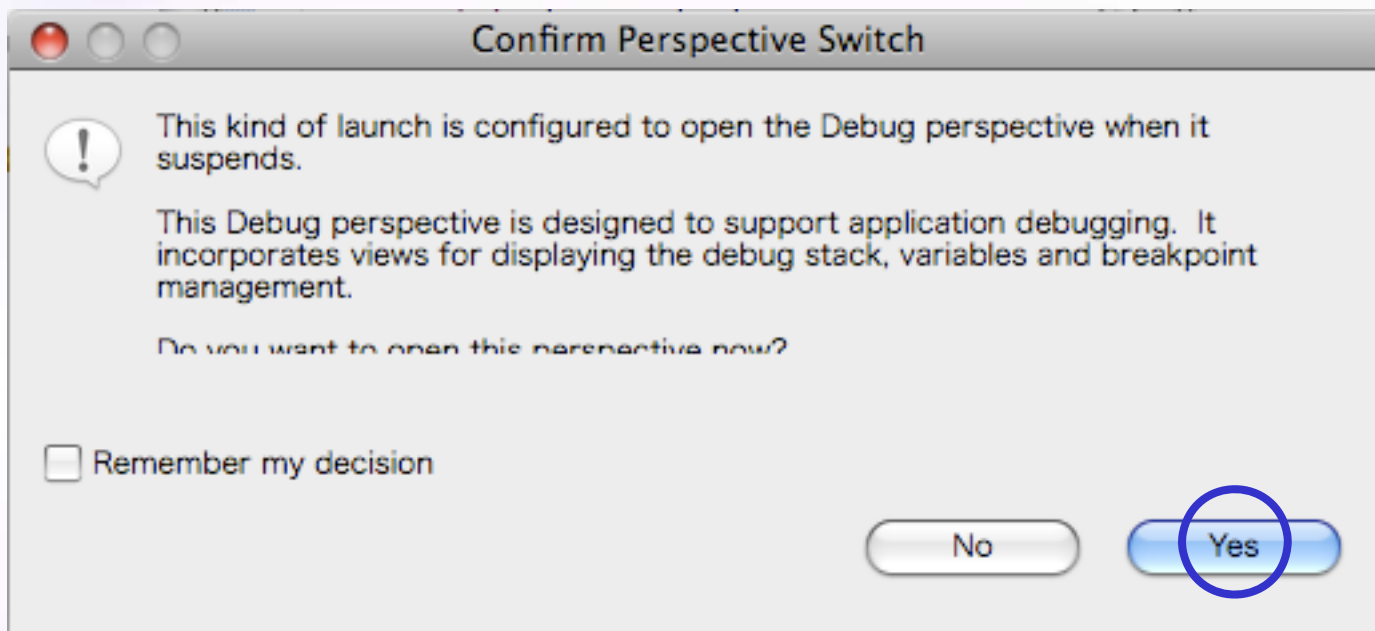
デバッガ: Step Over のみ使う場合



このボタンをクリックして
デバッガを起動する

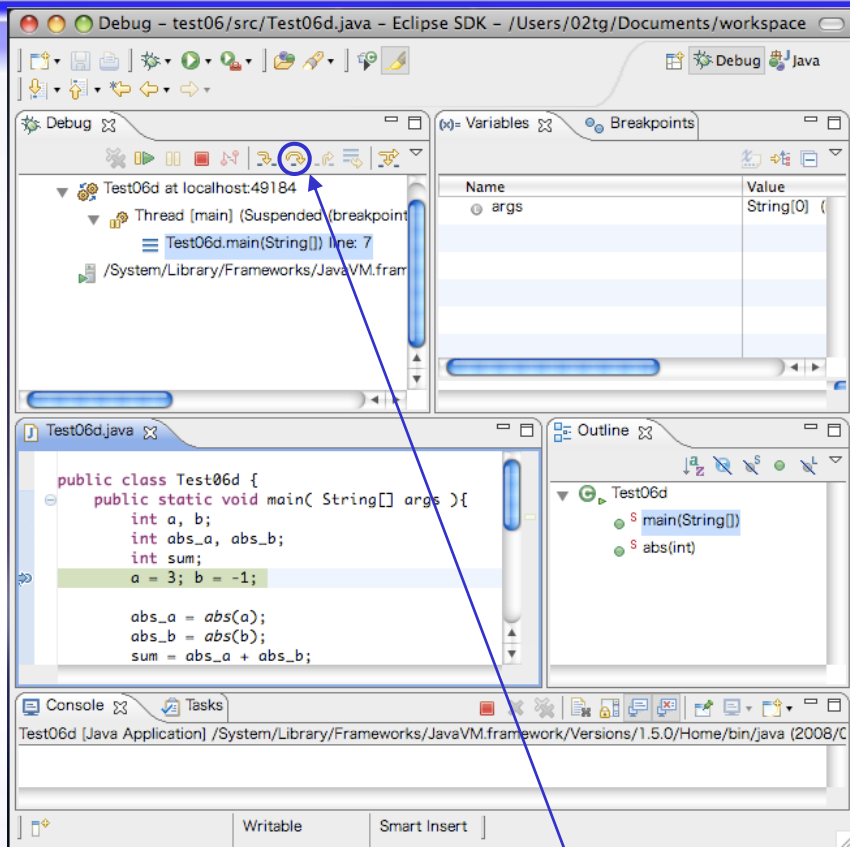
前回と同様, Step Over ボタンだけで
デバッガを試してみる

デバッガ: Step Over のみ使う場合

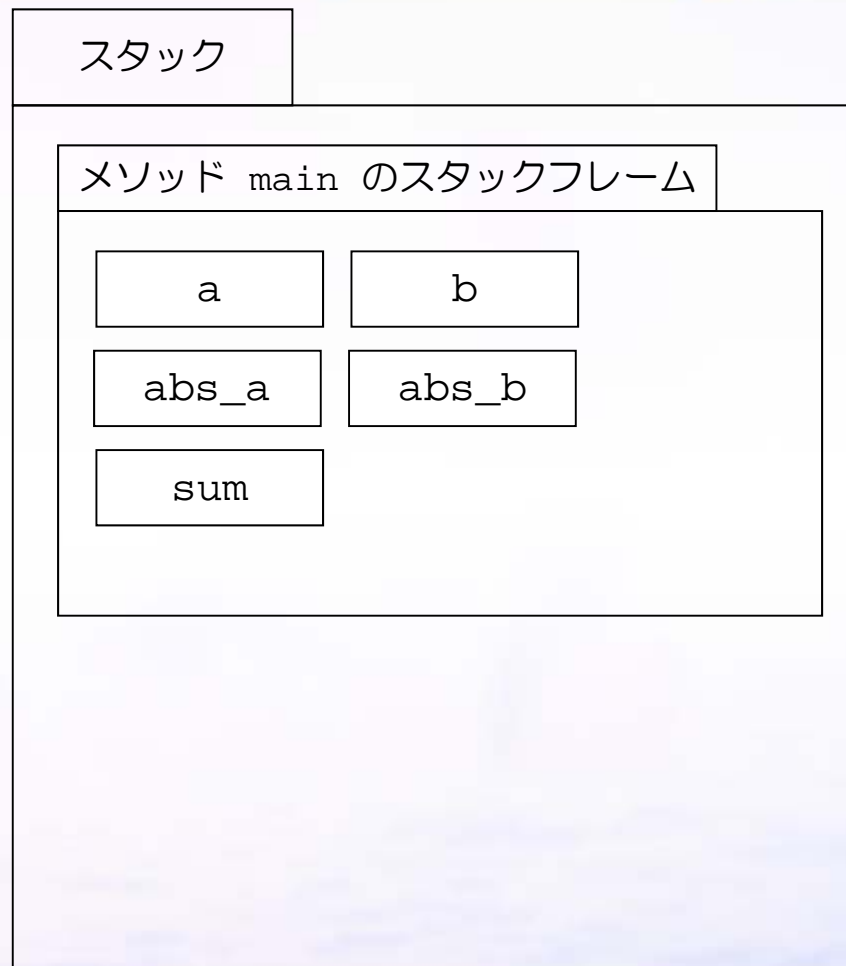


「デバッガの画面に切り替えてよいか？」と聞かれている。
「Yes」をクリックする

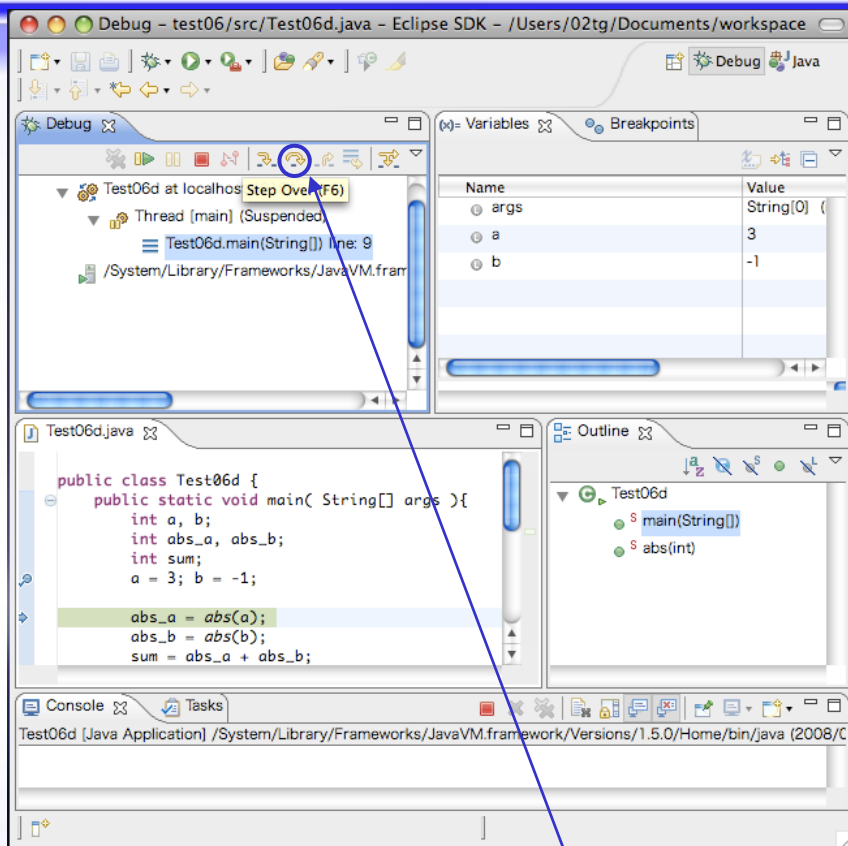
デバッガ: Step Over のみ使う場合



デバッガが起動された
ステップオーバーをクリック



デバッガ: Step Over のみ使う場合



a=3, b=-1が代入された
ステップオーバーをクリック

スタック

メソッド main のスタックフレーム

a=3

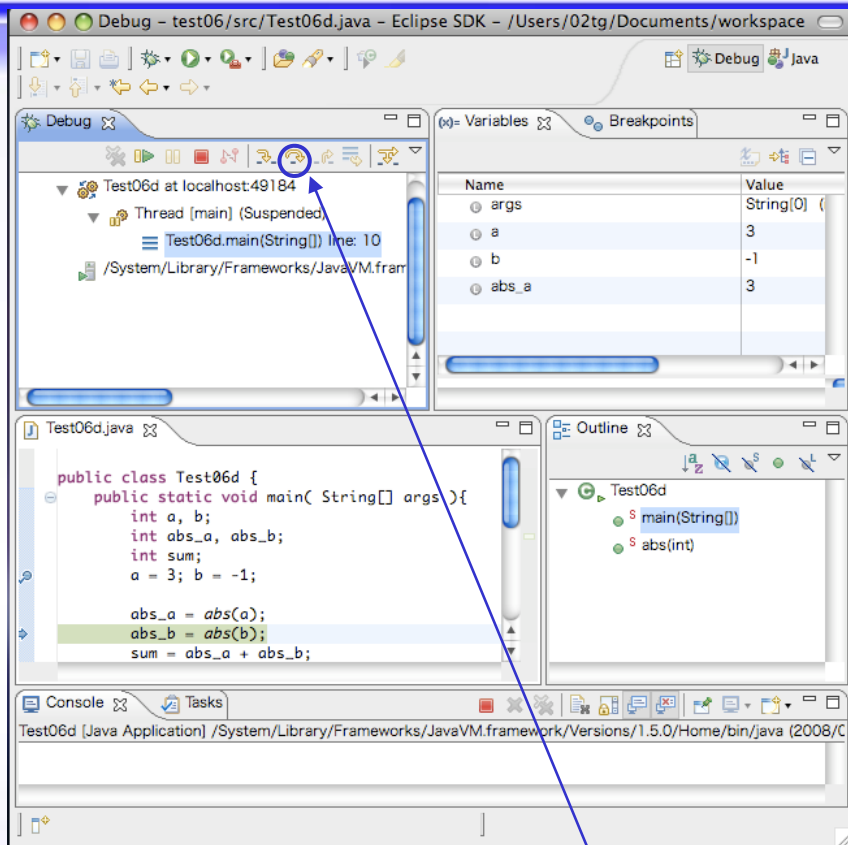
b=-1

abs_a

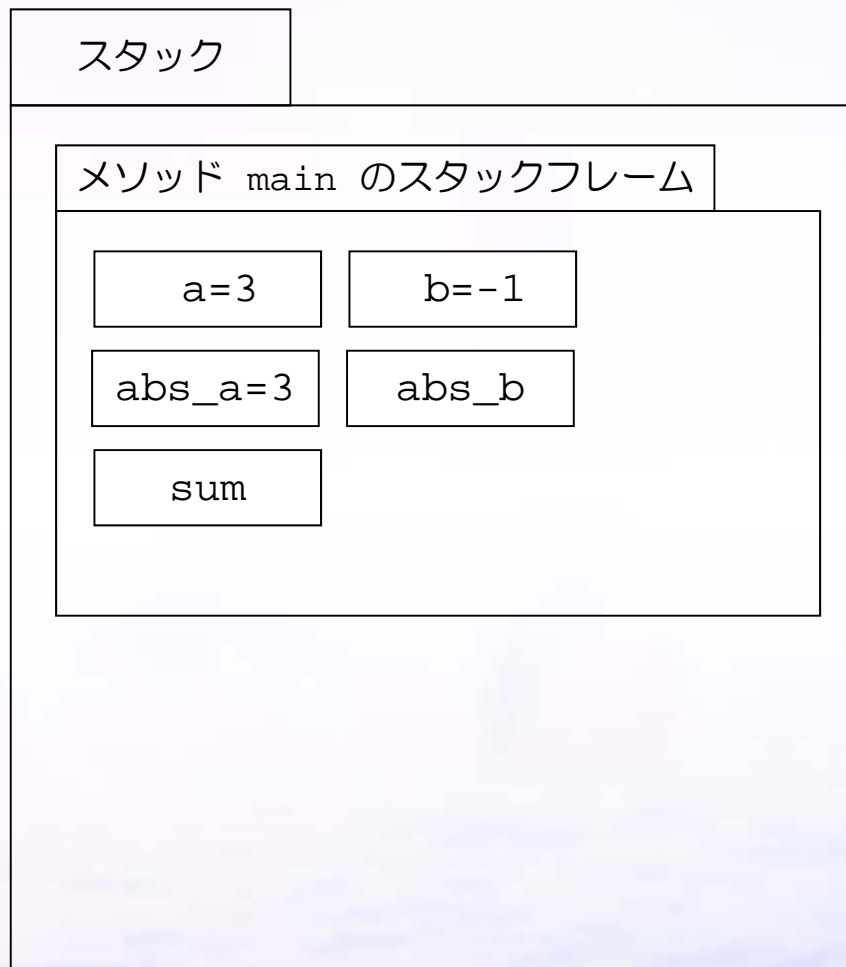
abs_b

sum

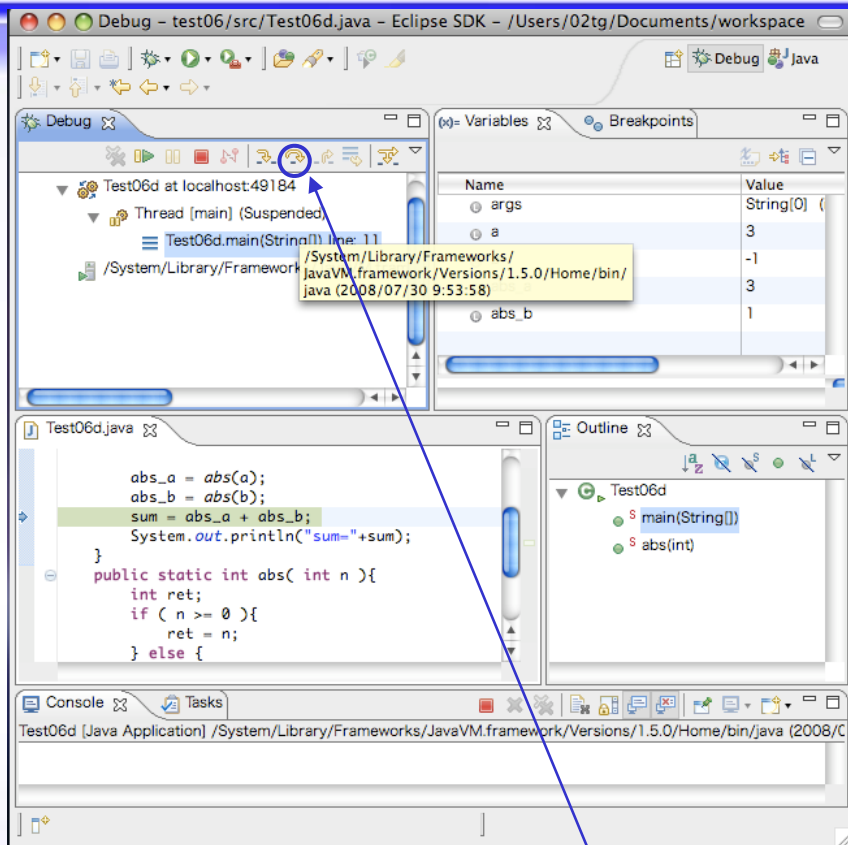
デバッガ: Step Over のみ使う場合



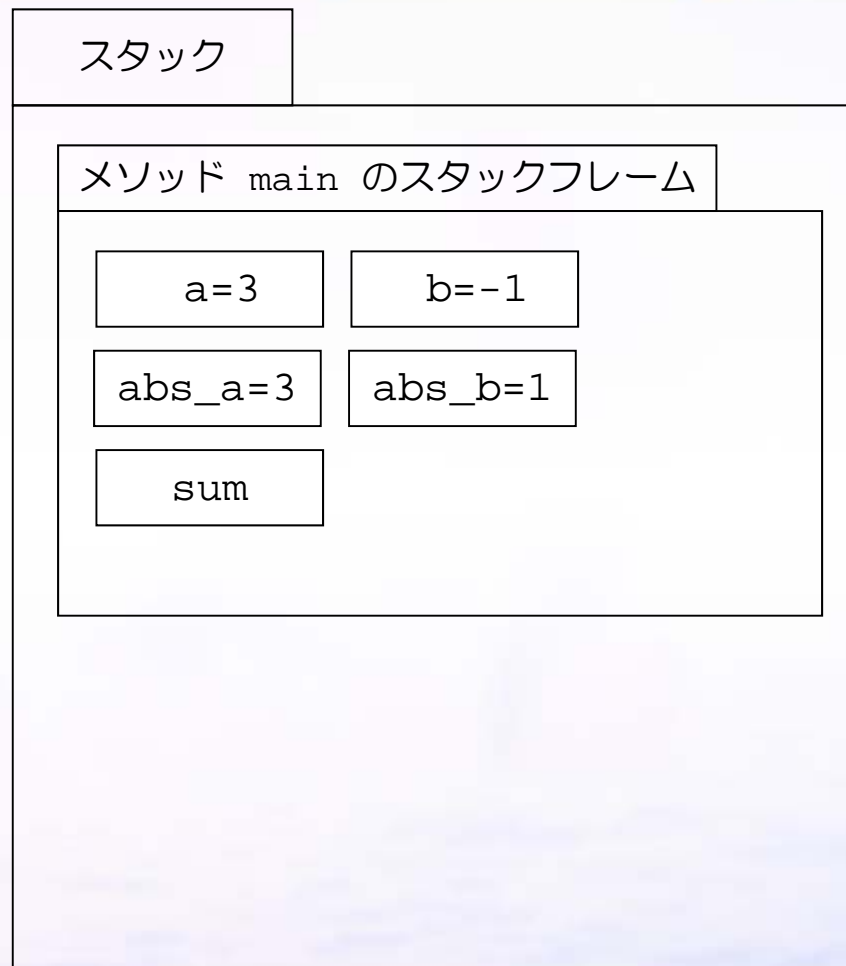
メソッド abs が呼び出され、結果が
変数 abs_a に代入された
ステップオーバーをクリック



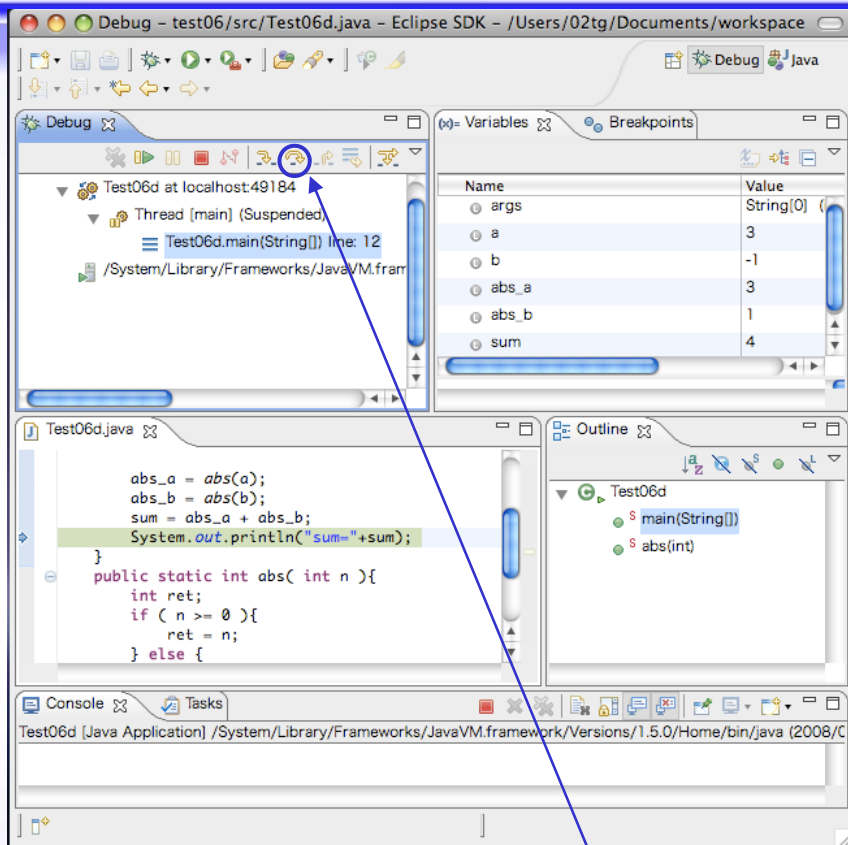
デバッガ: Step Over のみ使う場合



メソッド abs が呼び出され、結果が
変数 abs_b に代入された
ステップオーバーをクリック



デバッガ: Step Over のみ使う場合



スタック

メソッド main のスタックフレーム

a=3

b=-1

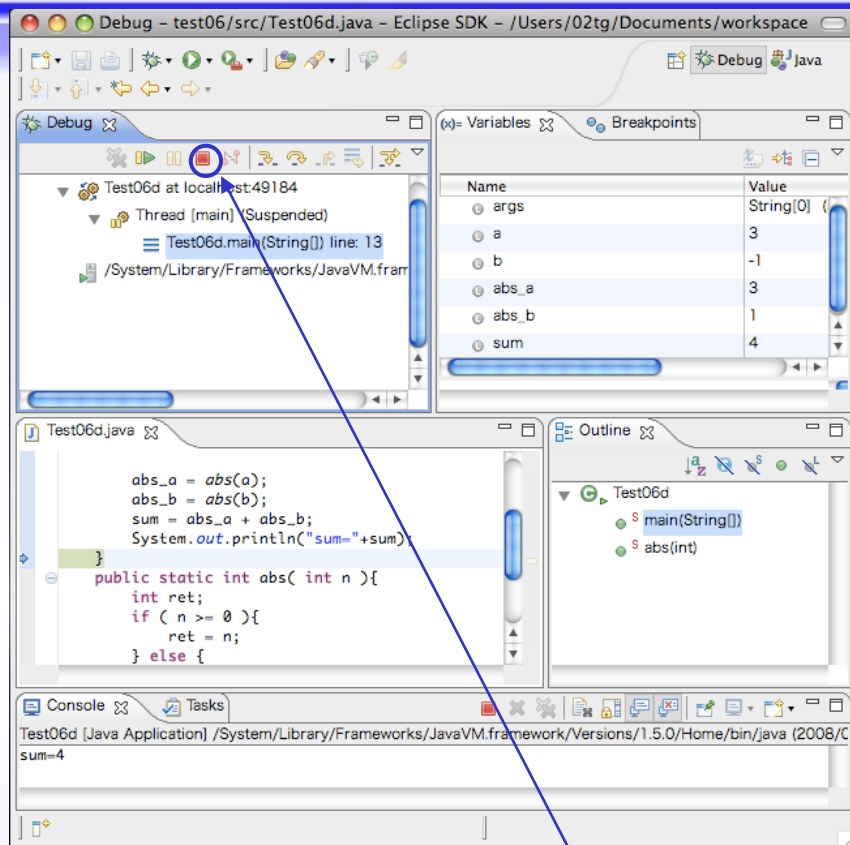
abs_a=3

abs_b=1

sum=4

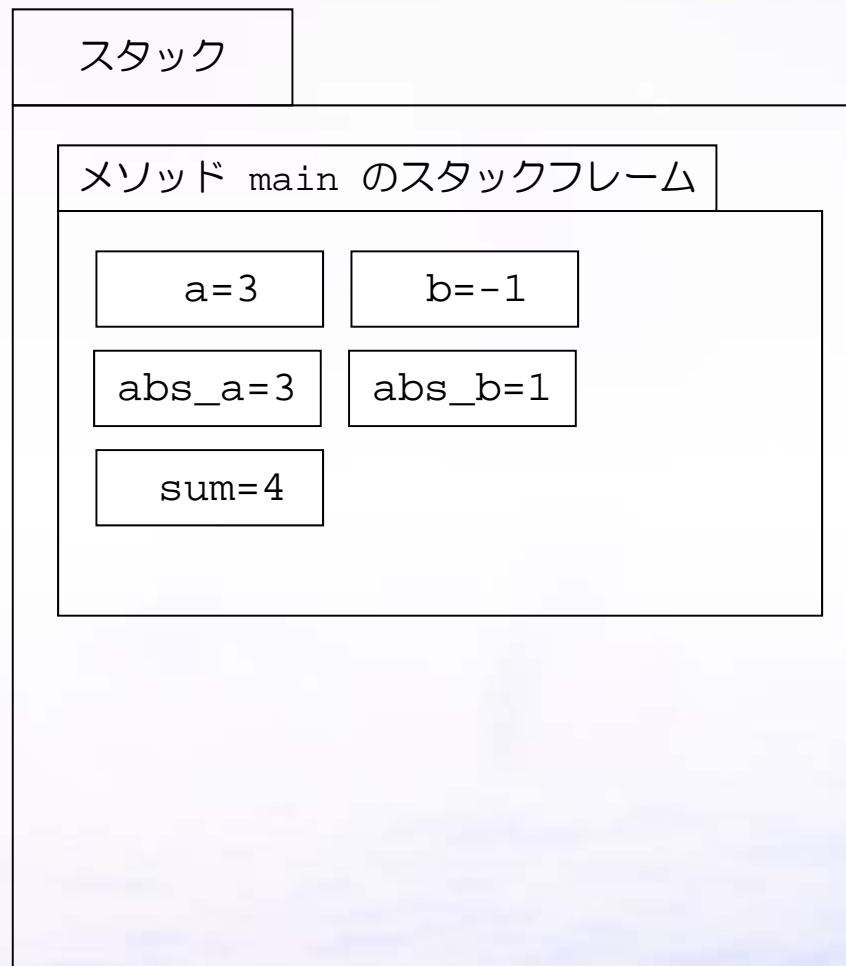
式 abs_a+abs_b の値が計算され、
結果が変数 `sum` に代入された
ステップオーバーをクリック

終わったらデバッガを終了させる

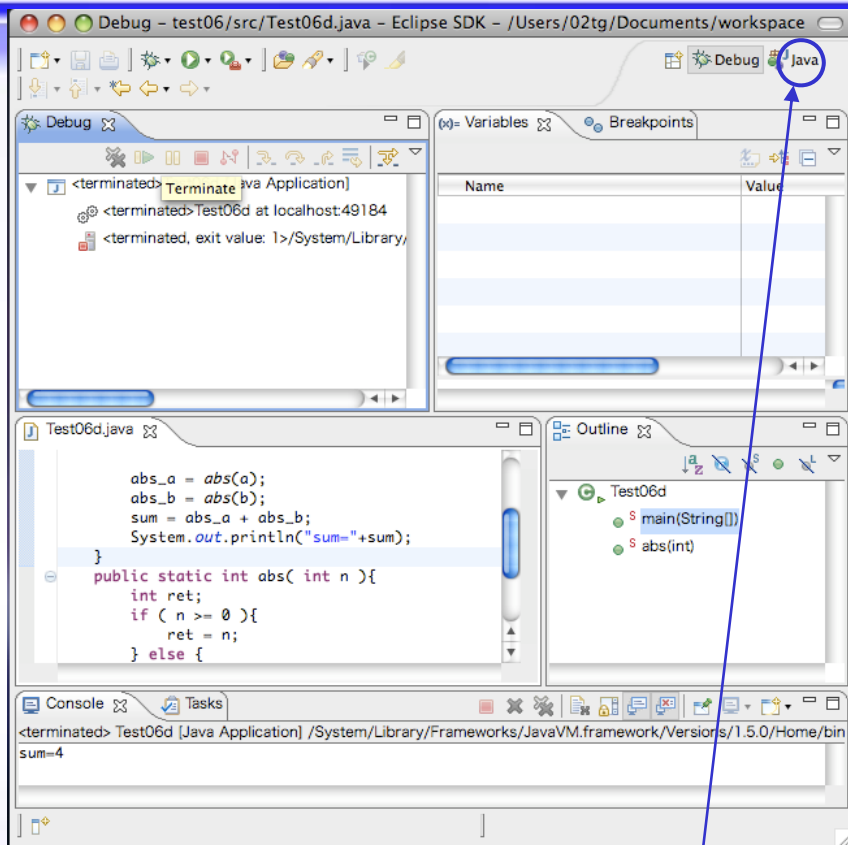


結果が画面に出力された

停止ボタンをクリック



終わったらデバッガを終了させる



結果が画面に出力された
これをクリックすると、デバッガの画面から
通常の画面に戻る

スタック

メソッド main のスタックフレーム

a=3

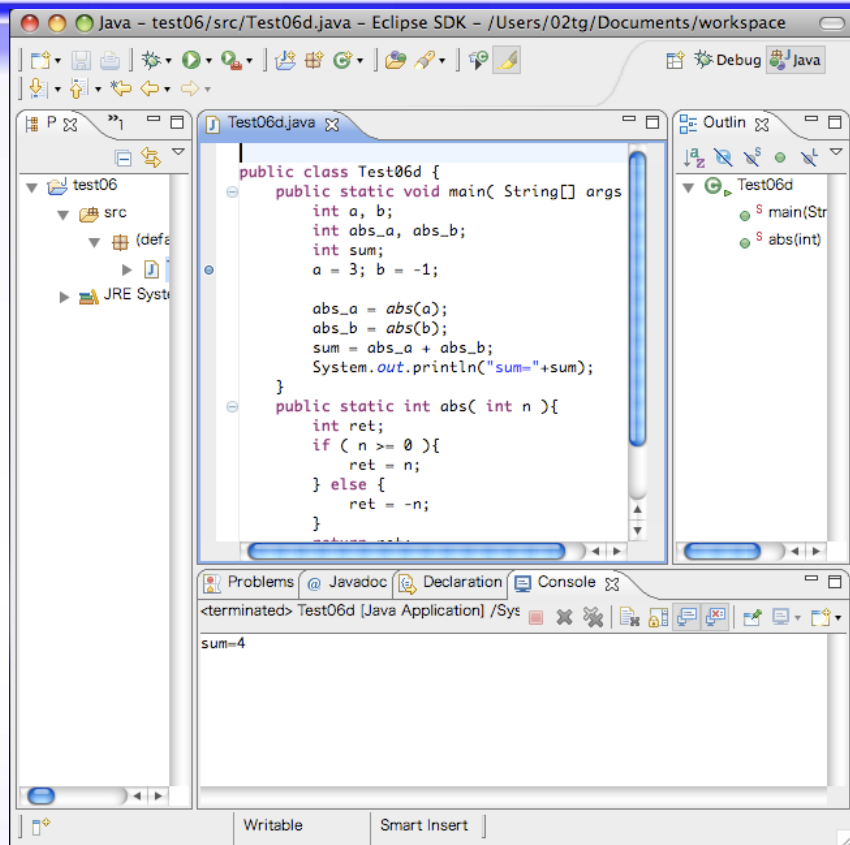
b=-1

abs_a=3

abs_b=1

sum=4

終わったらデバッガを終了させる



通常の画面に戻った

スタック

メソッド main のスタックフレーム

a=3

b=-1

abs_a=3

abs_b=1

sum=4

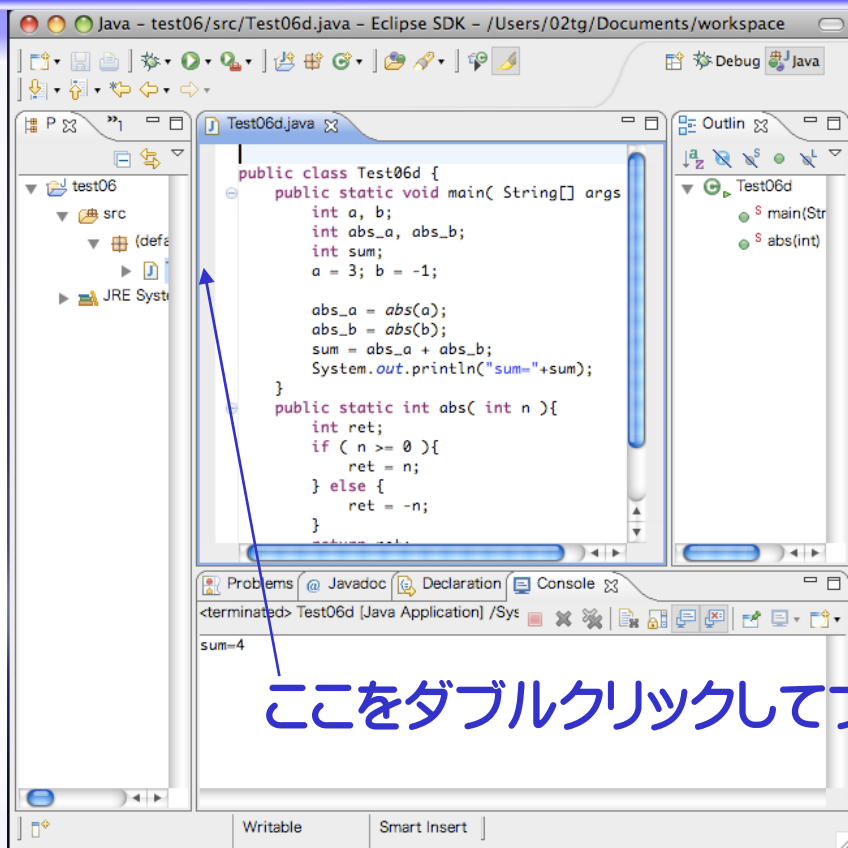
まとめ

- ・ Java文法の基本, メソッドを理解する
- ・ メソッドに対するデバッガの操作を学習する
 - 「Step Over」ボタンは, メソッドの呼び出しも含めて1行ずつ動作させる

- 「Step Into」ボタンは, メソッドの先頭のコードを実行する手前まで進める



デバッガ: Step Into ボタンを使うと

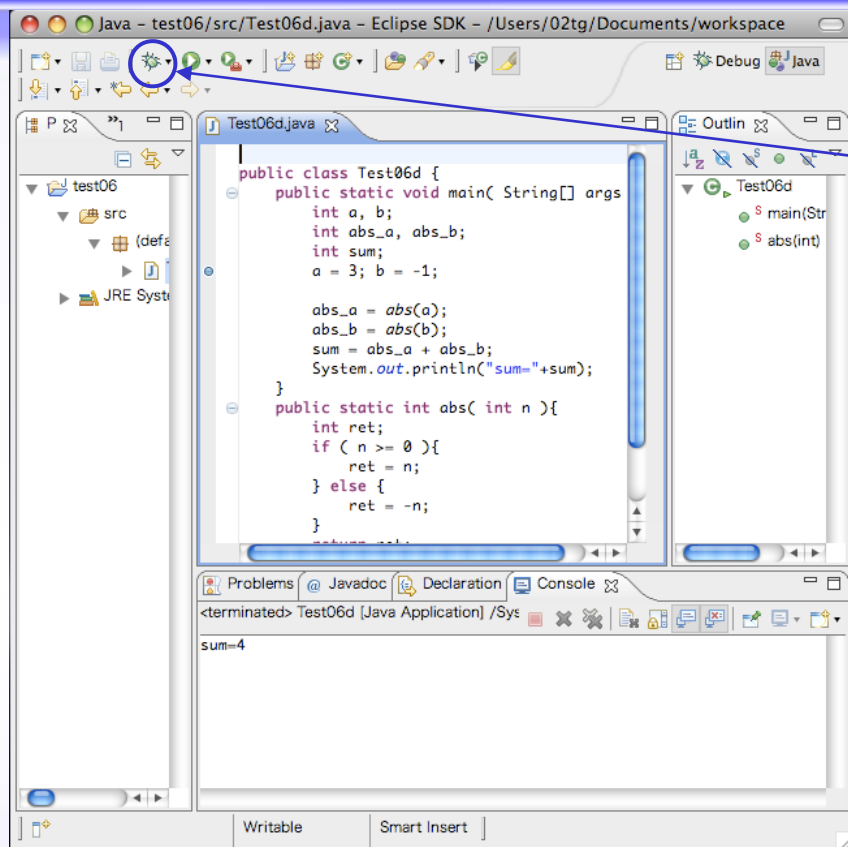


ここをダブルクリックしてブレークポイントを作る

前回と同様, Step Over ボタンだけで
デバッガを使ってみる

前回と同じ作業

デバッガ: Step Into ボタンを使うと

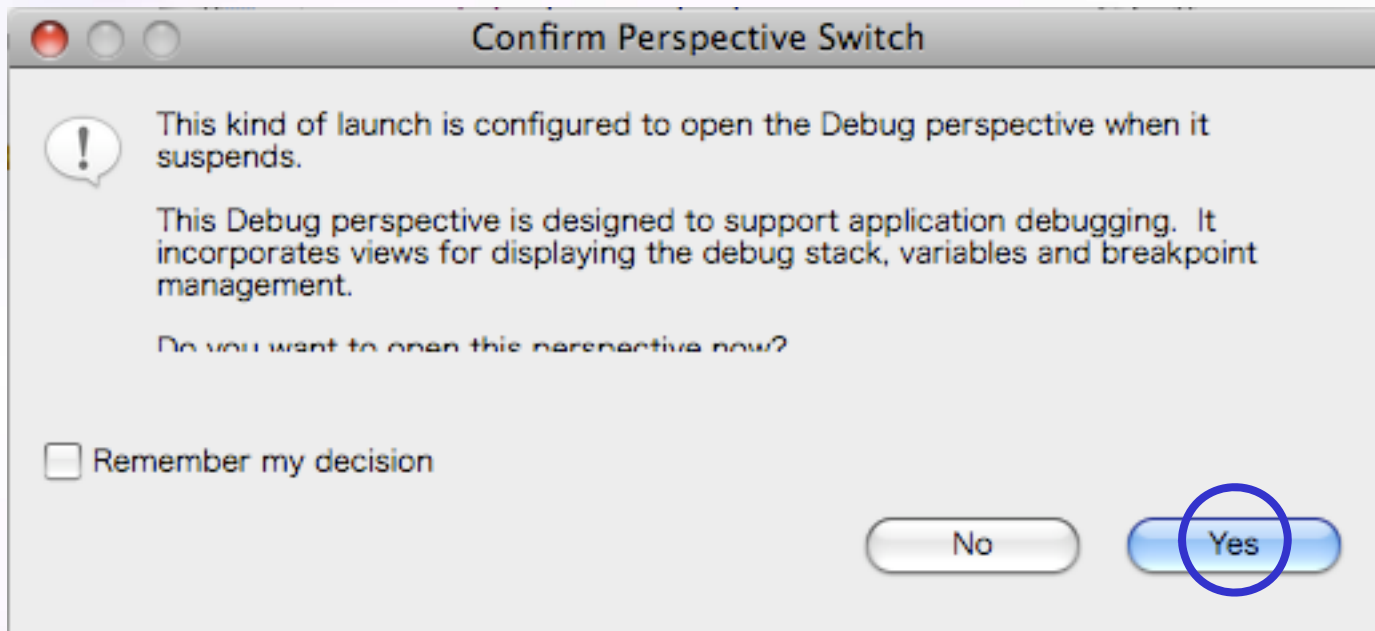


このボタンをクリックして
デバッガを起動する

前回と同様, Step Over ボタンだけで
デバッガを使ってみる

前回と同じ作業

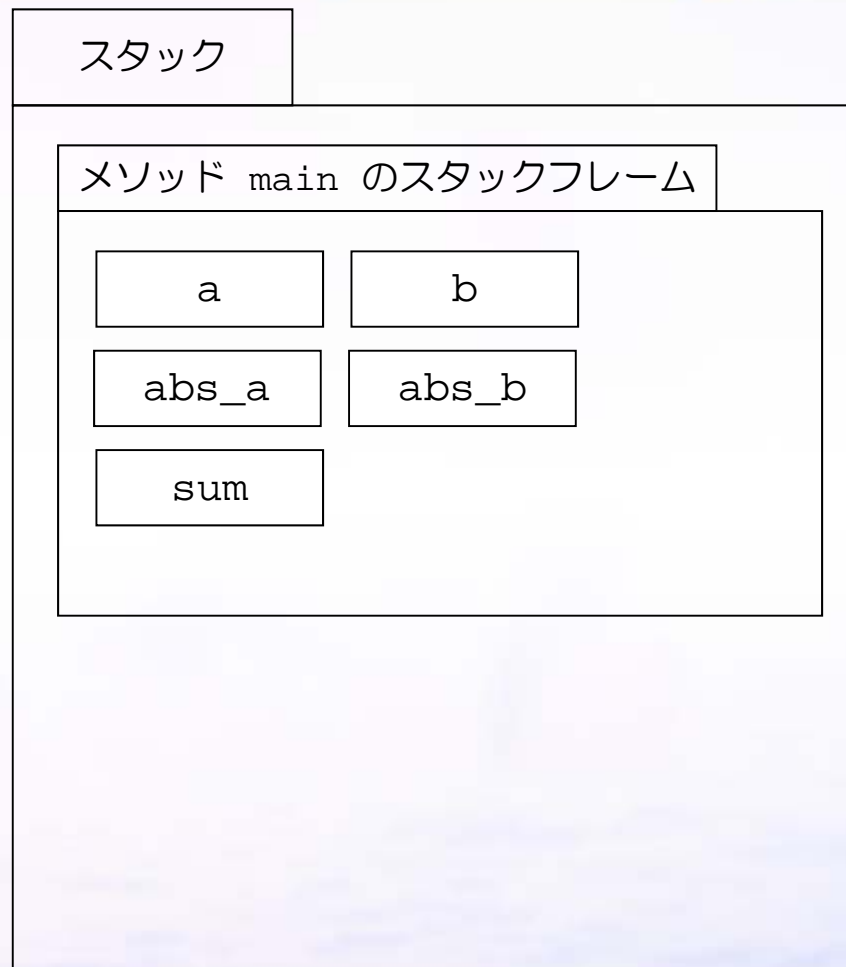
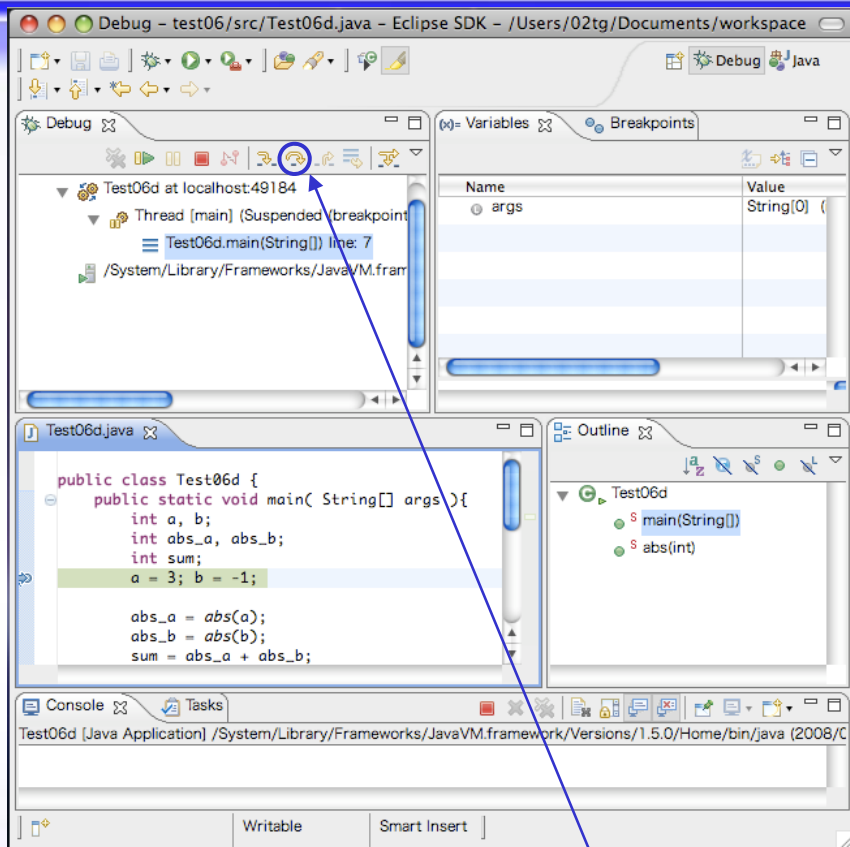
デバッガ: Step Into ボタンを使うと



「デバッガの画面に切り替えてよいか？」と聞かれている。
「Yes」をクリックする

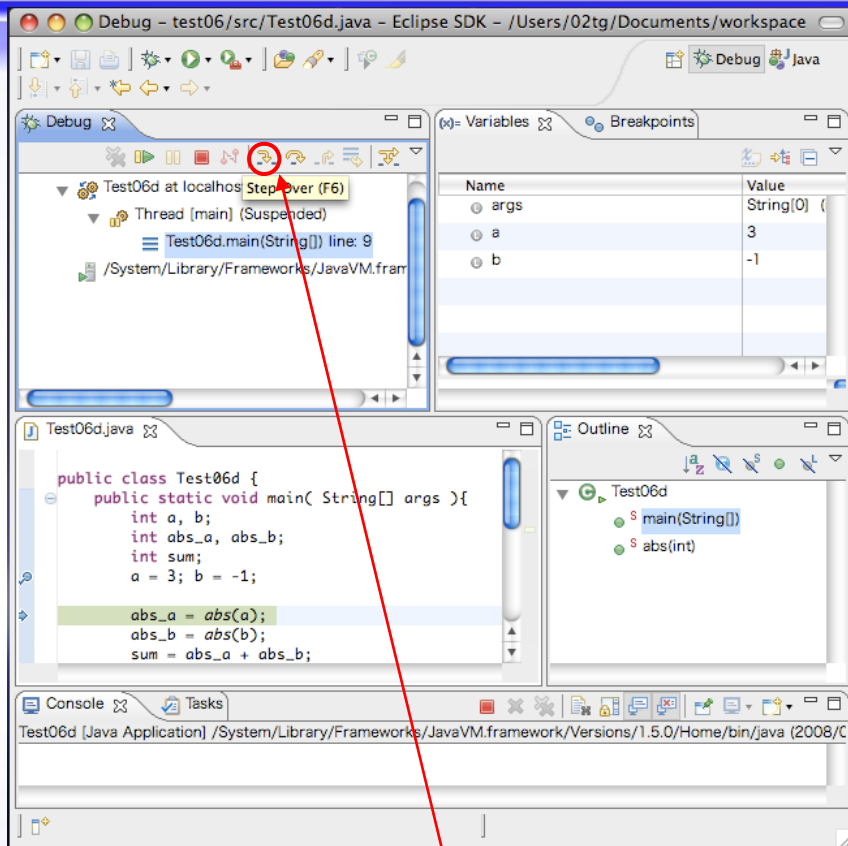
前回と同じ作業

デバッガ: Step Over のみ使う場合

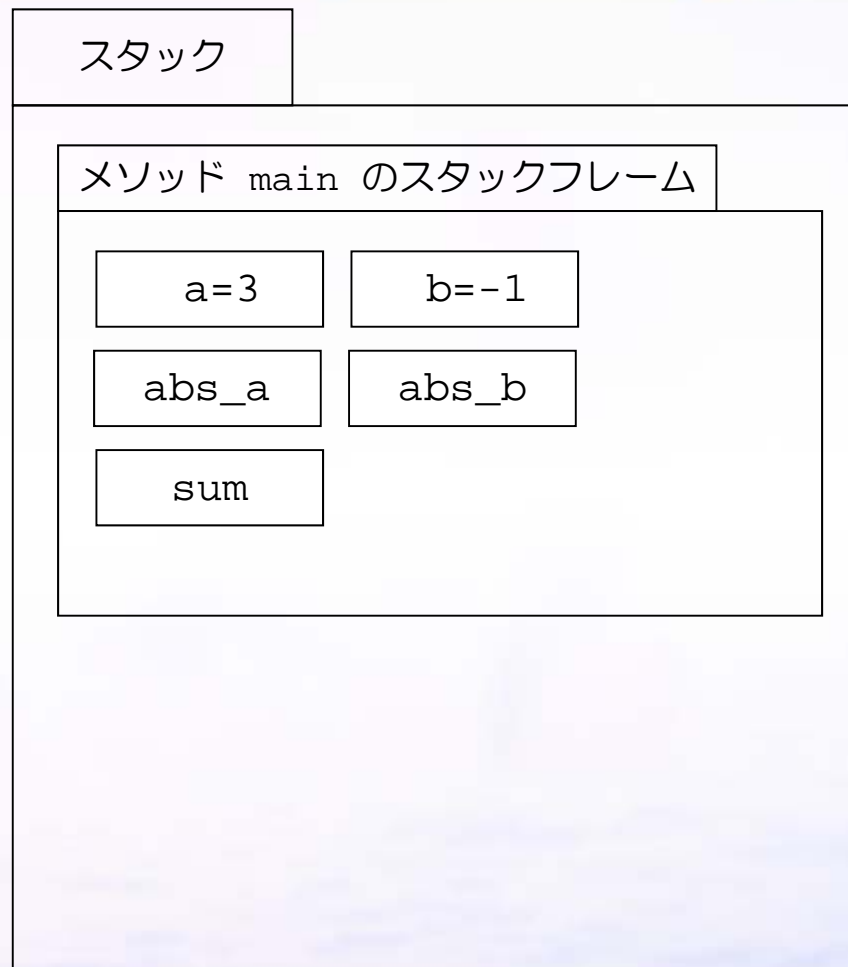


デバッガが起動され、最初のブレークポイントまで処理が進んだ
ステップオーバーをクリック

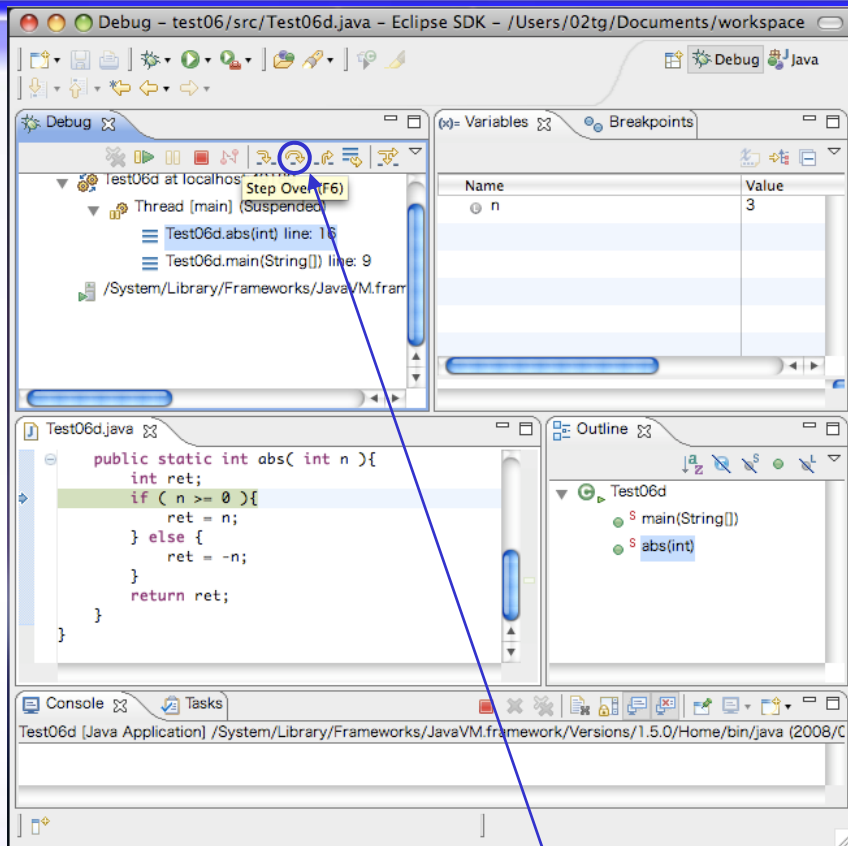
デバッガ: Step Into ボタンを使うと



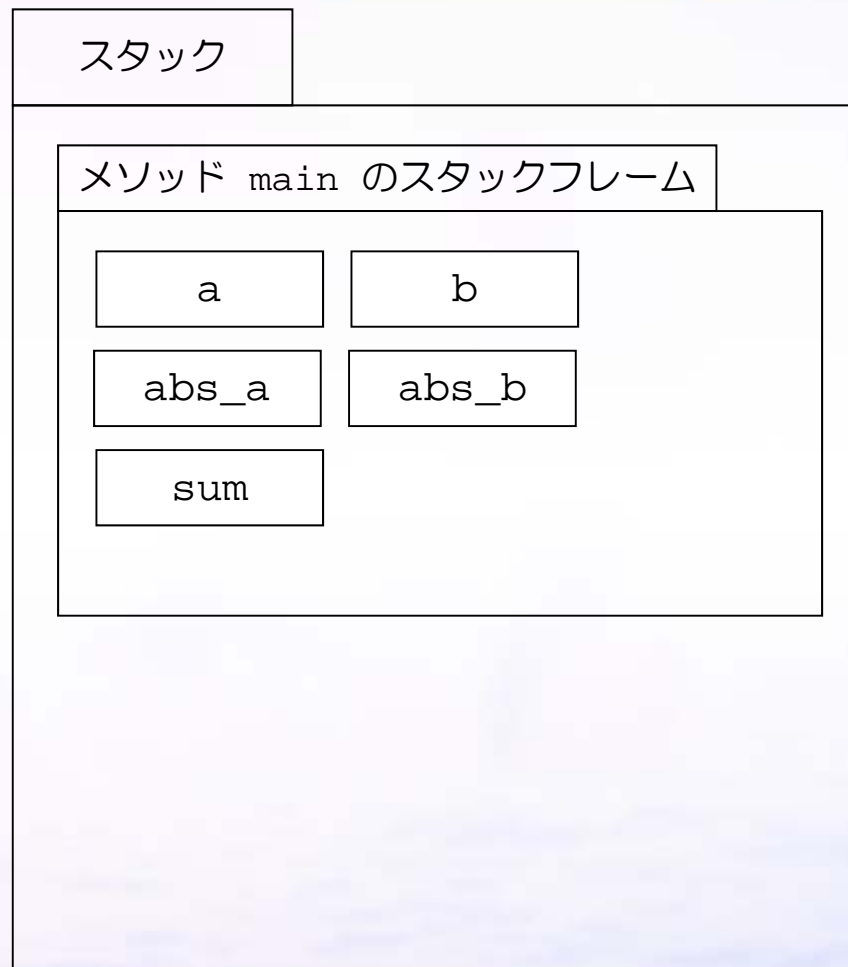
a=3, b=-1が代入された
Step Intoをクリック



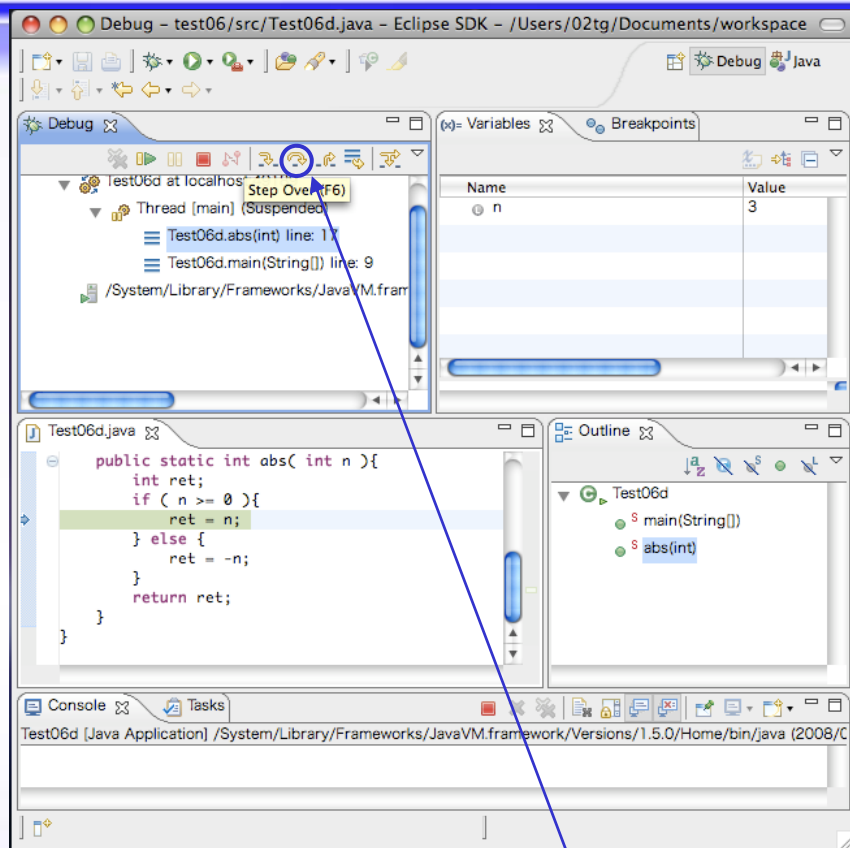
デバッガ: Step Into ボタンを使うと



メソッド abs の最初のステップまで
進んだ
ステップオーバーをクリック



デバッガ: Step Into ボタンを使うと



**n >= 0 なので
ステップオーバーをクリック**

スタック

メソッド main のスタックフレーム

a=3

b=-1

abs_a

abs_b

sum

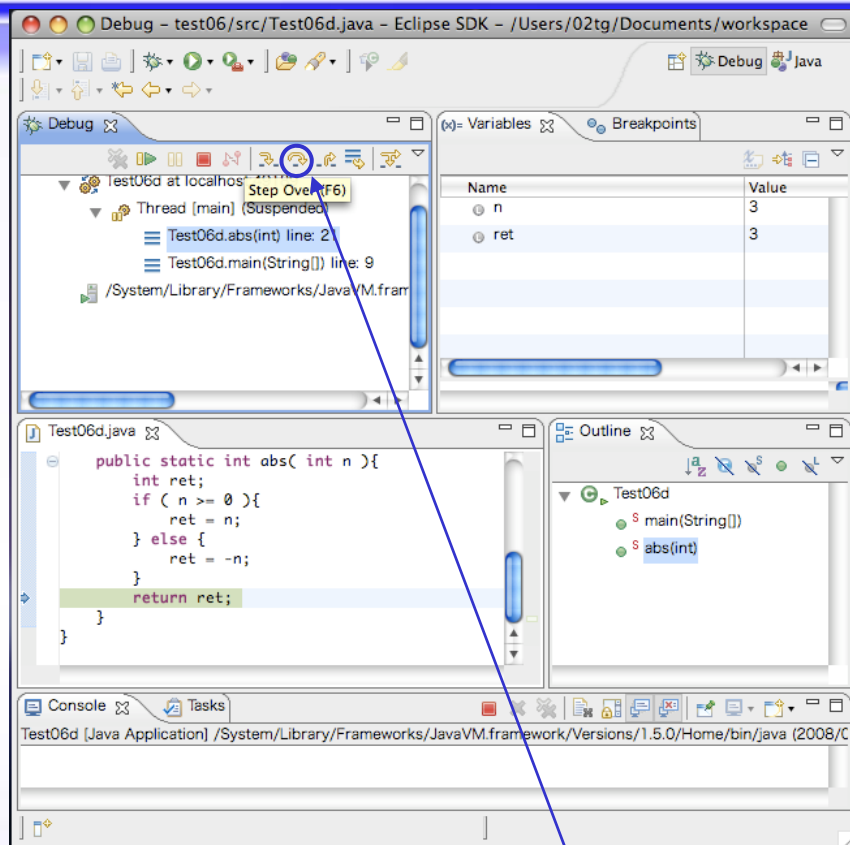
メソッド abs のスタックフレーム

n=3

ret

引数を含む変数が作られる

デバッガ: Step Into ボタンを使うと



変数 ret に n の値が代入された
ステップオーバーをクリック

スタック

メソッド main のスタックフレーム

a=3

b=-1

abs_a

abs_b

sum

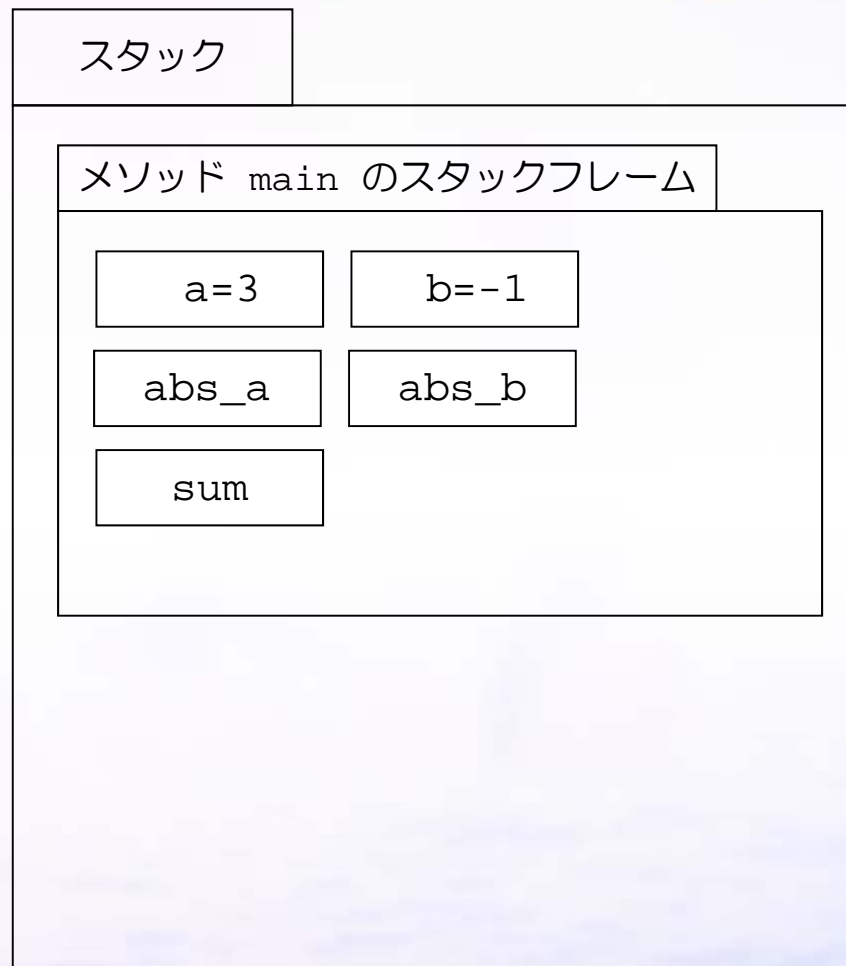
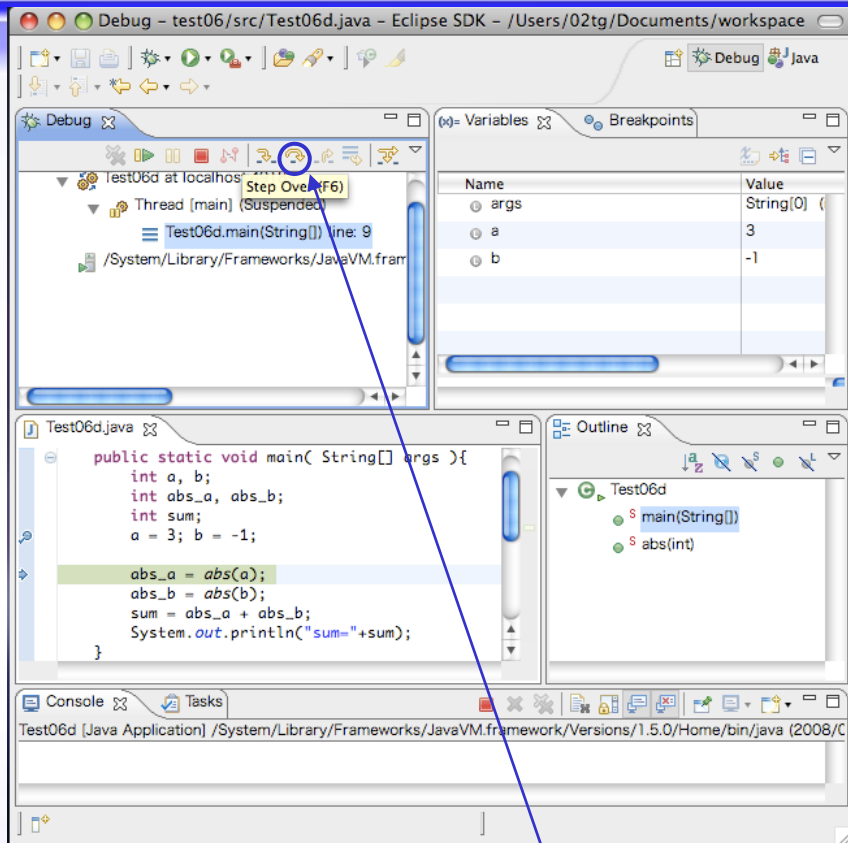
メソッド abs のスタックフレーム

n=3

ret=3

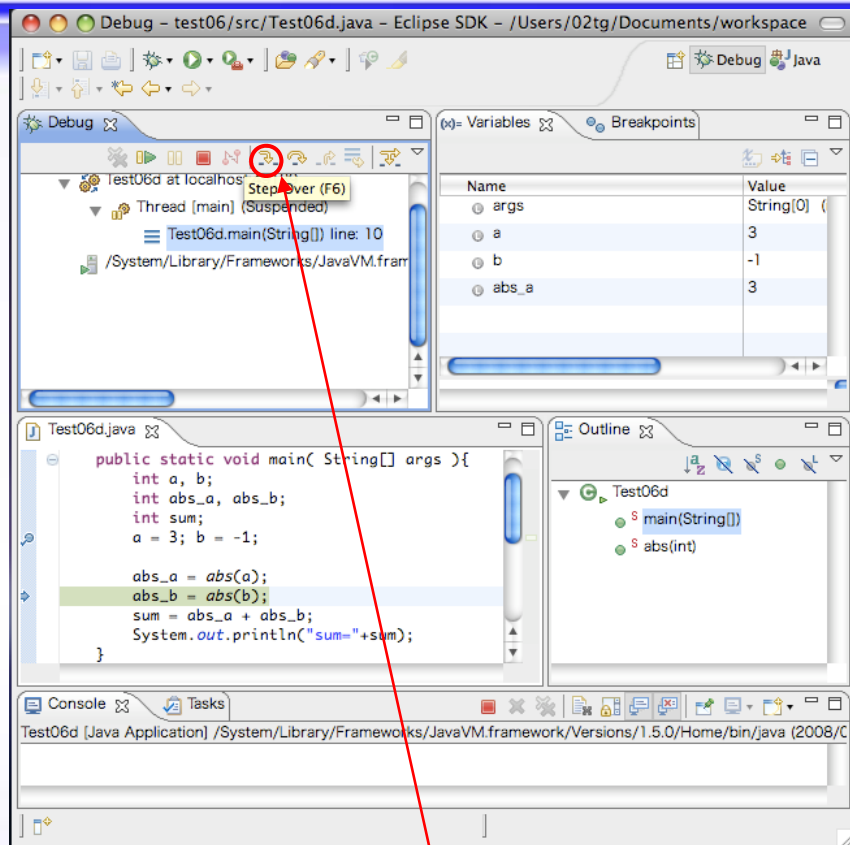
引数を含む変数が作られる

デバッガ: Step Into ボタンを使うと



戻り値を3としてメソッド abs が
終了して、呼び出し元に戻ってきた
ステップオーバーをクリック

デバッガ: Step Into ボタンを使うと



スタック

メソッド main のスタックフレーム

a=3

b=-1

abs_a=3

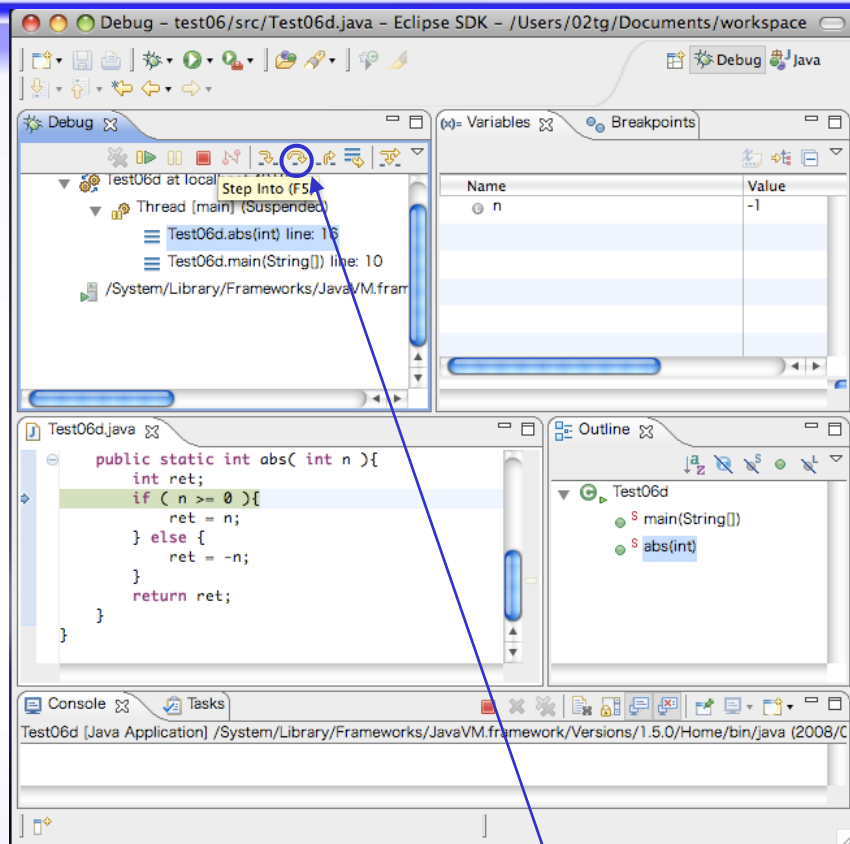
abs_b

sum

変数 abs_a にメソッド abs の
戻り値3が代入された

もう一度Step Intoの動きをみてみましょう

デバッガ: Step Into ボタンを使うと



メソッド abs の最初のステップまで
進んだ
ステップオーバーをクリック

スタック

メソッド main のスタックフレーム

a=3

b=-1

abs_a=3

abs_b

sum

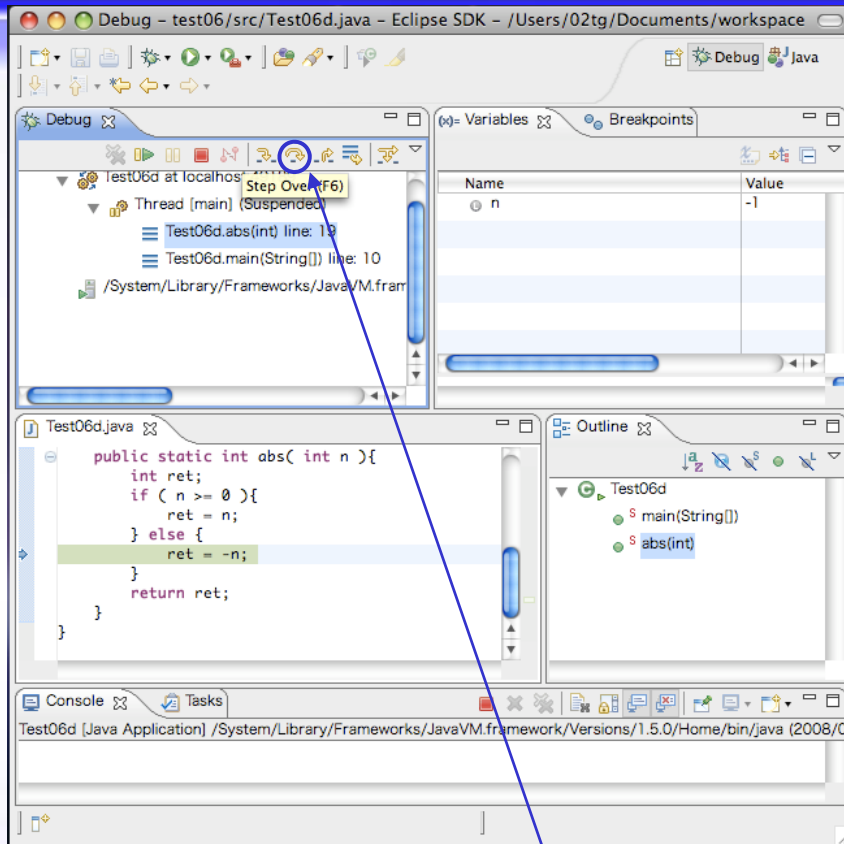
メソッド abs のスタックフレーム

n=-1

ret

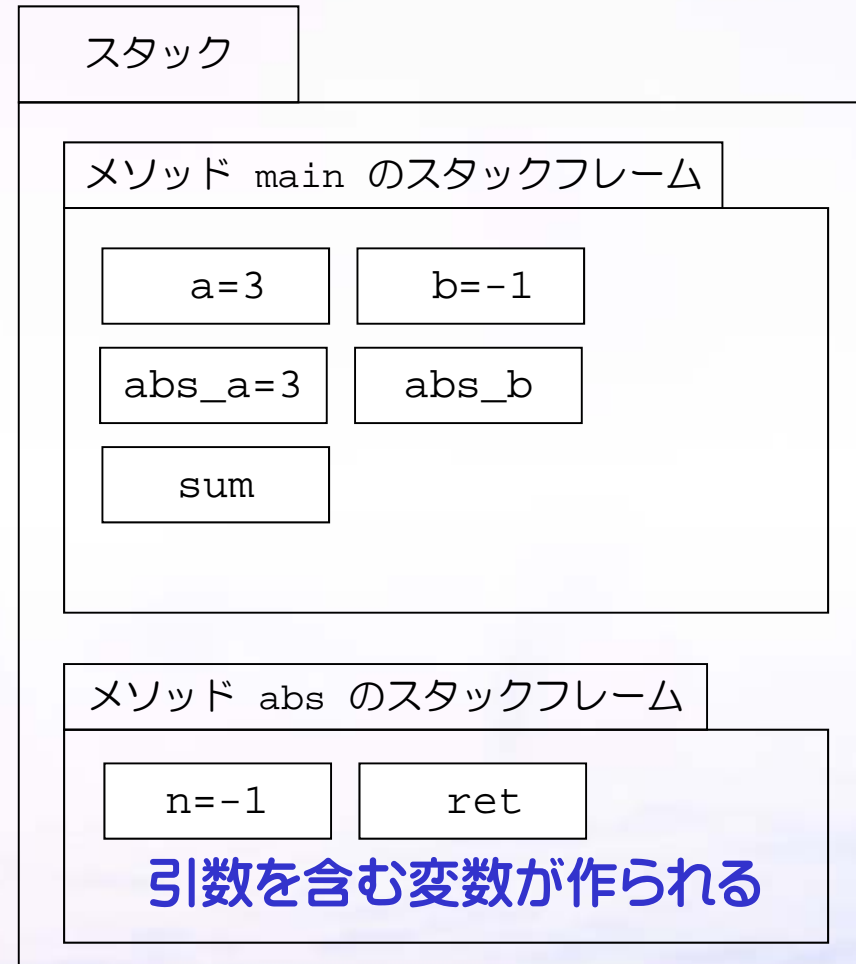
引数を含む変数が作られる

デバッガ: Step Into ボタンを使うと

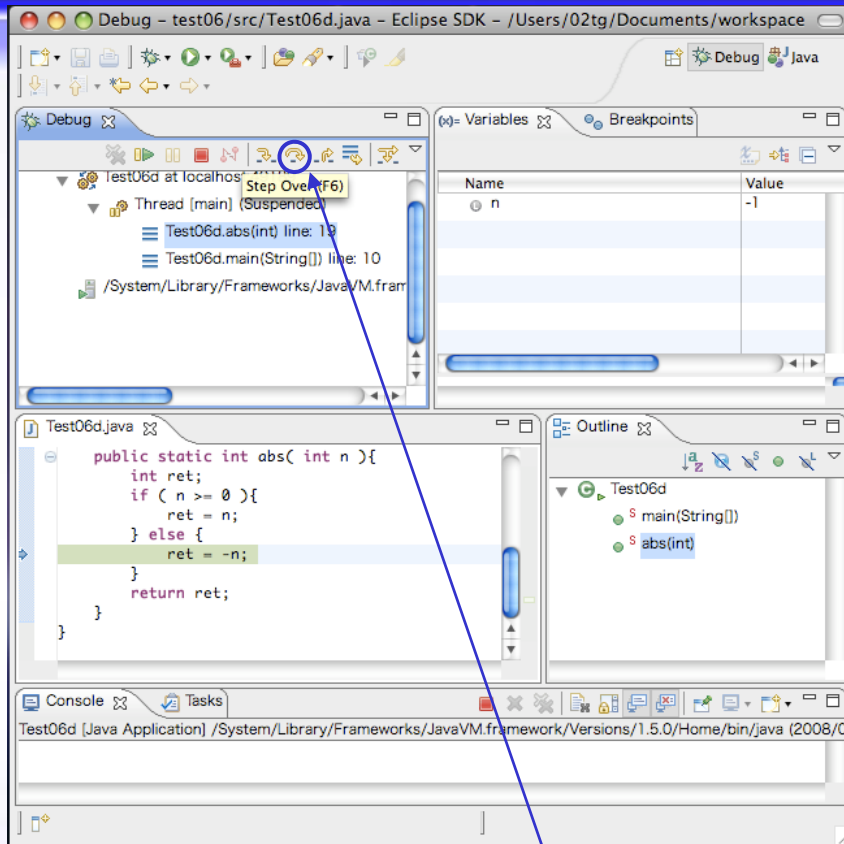


$n \geq 0$ の条件は満たさないので

ステップオーバーをクリック

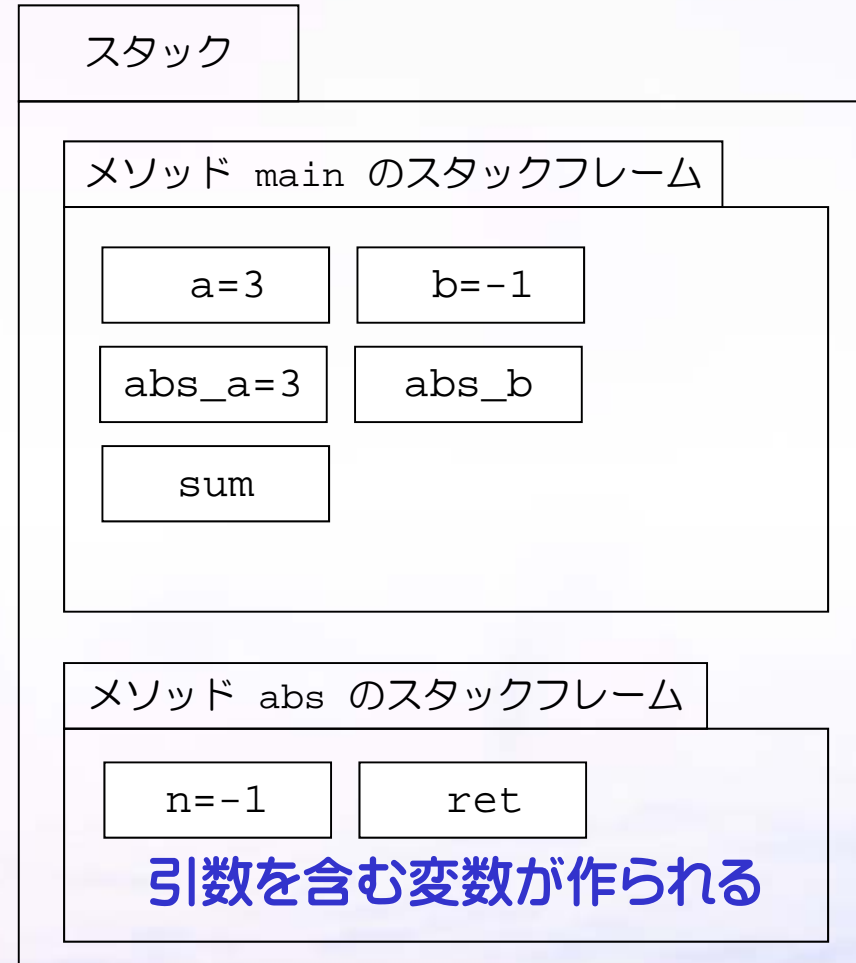


デバッガ: Step Into ボタンを使うと

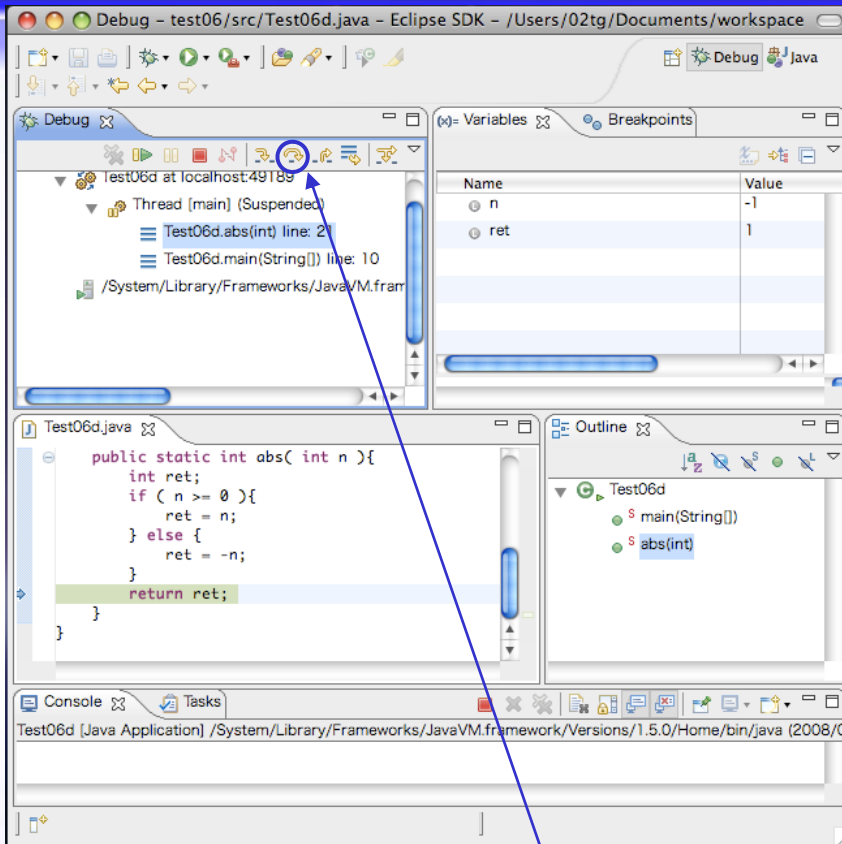


$n \geq 0$ の条件は満たさないので

ステップオーバーをクリック



デバッガ: Step Into ボタンを使うと



変数 ret に $-n$ の値が代入された

ステップオーバーをクリック

スタック

メソッド main のスタックフレーム

a=3

b=-1

abs_a=3

abs_b

sum

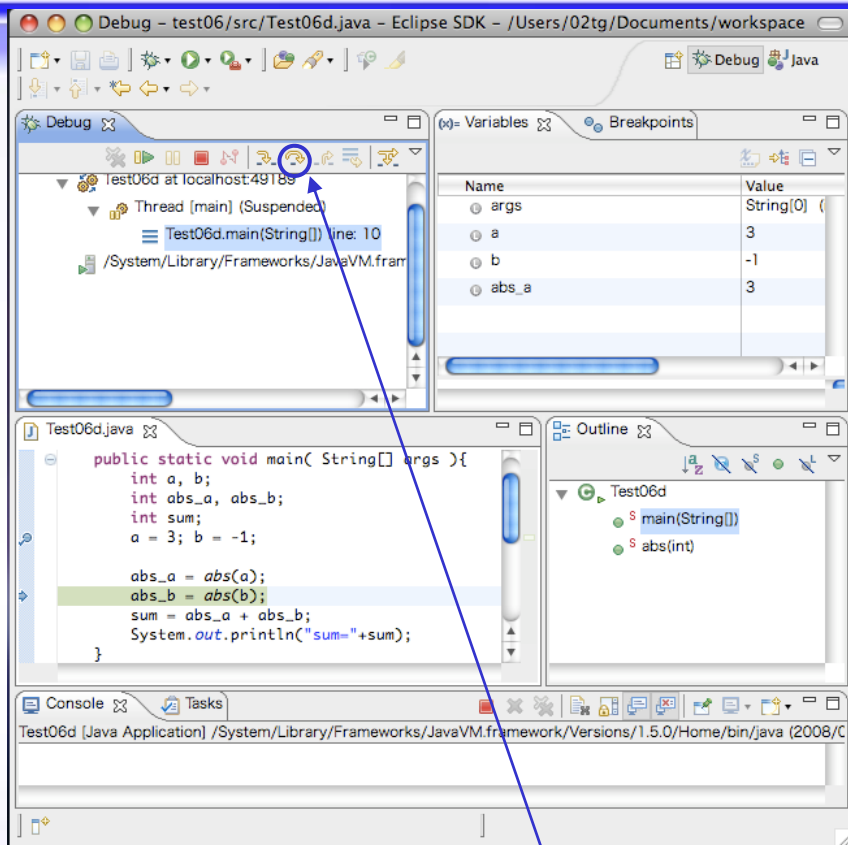
メソッド abs のスタックフレーム

n=-1

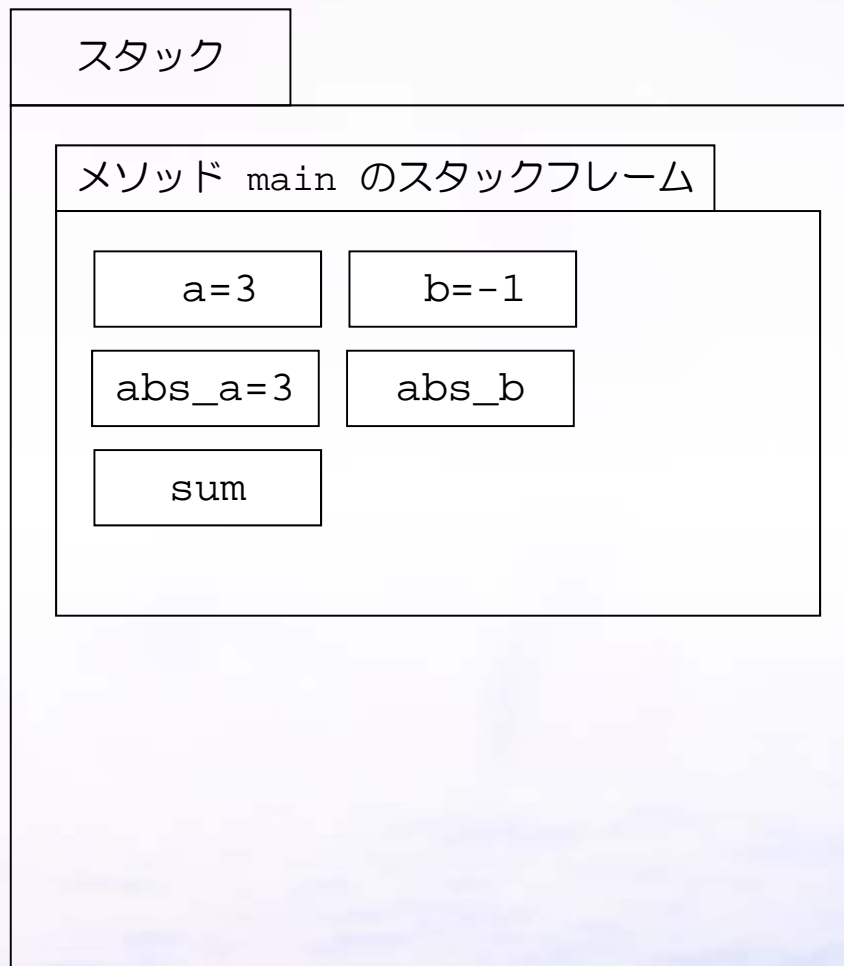
ret=1

引数を含む変数が作られる

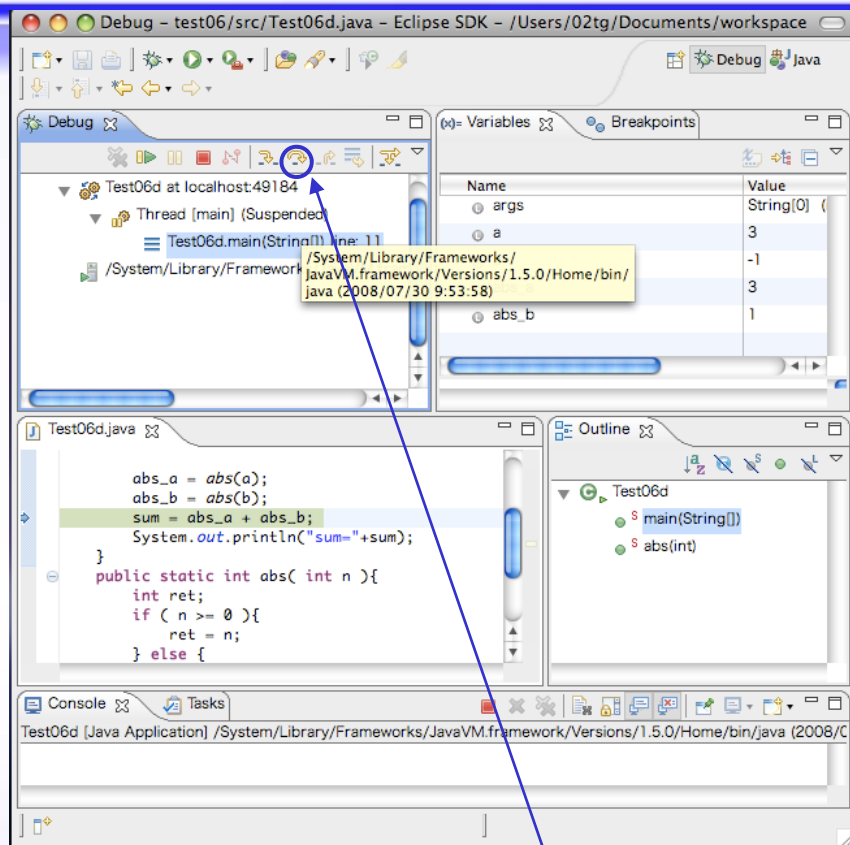
デバッガ: Step Into ボタンを使うと



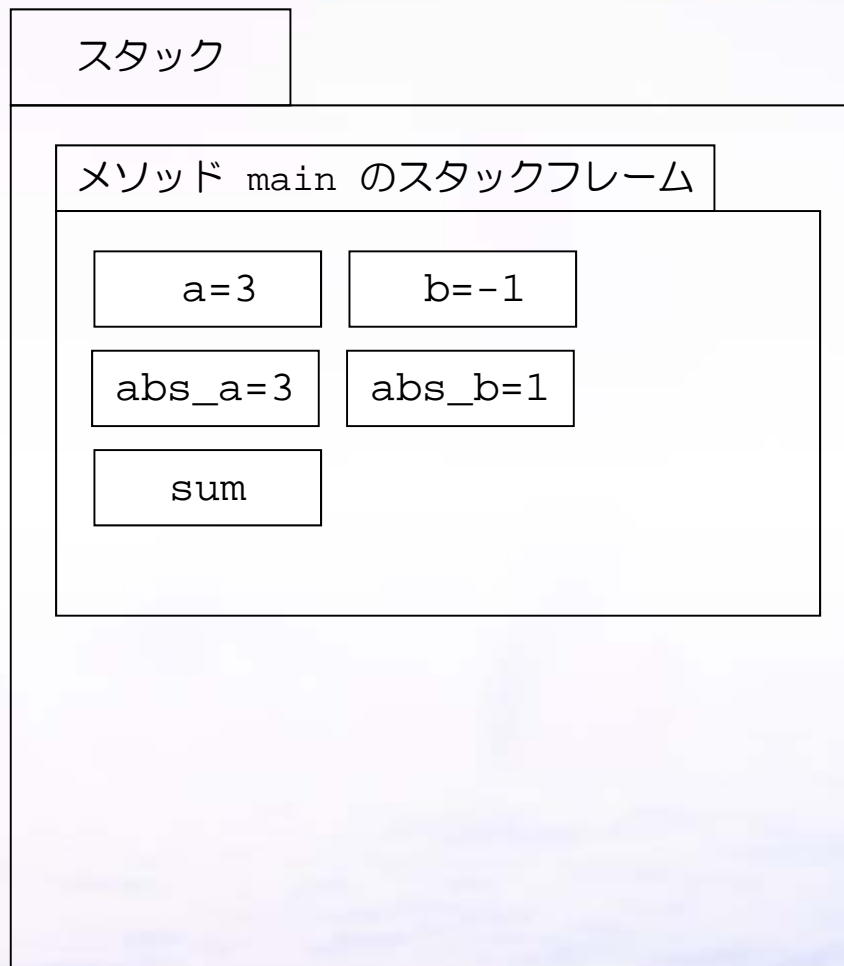
戻り値を1としてメソッド abs が
終了して、呼び出し元に戻ってきた
ステップオーバーをクリック



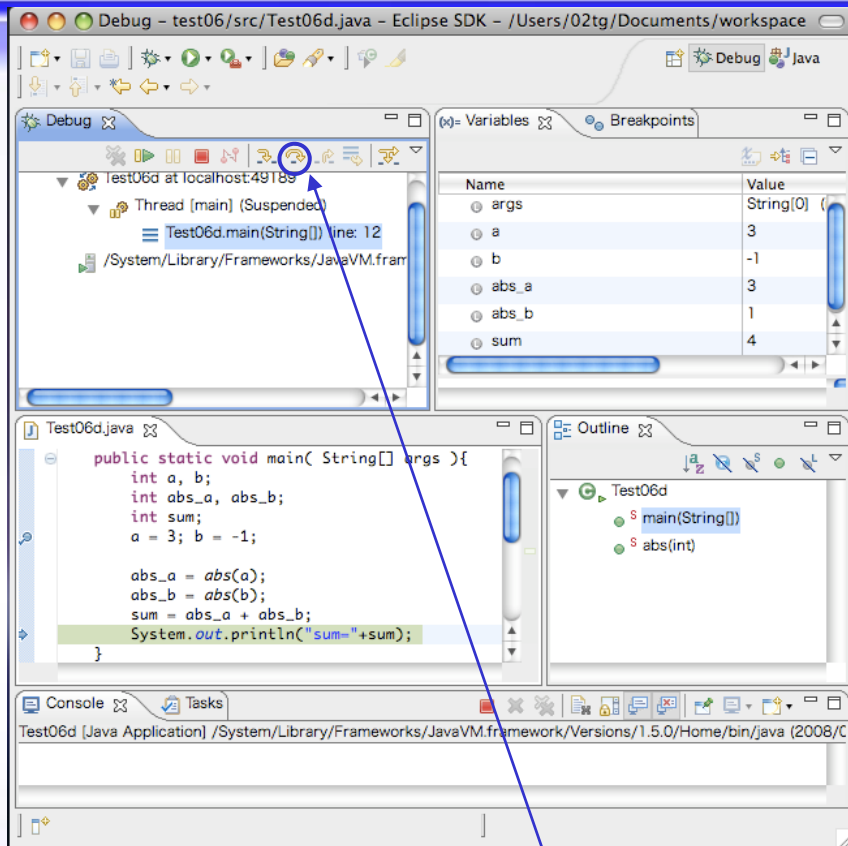
デバッガ: Step Into ボタンを使うと



変数 `abs_b` にメソッド `abs` の
戻り値1が代入された
ステップオーバーをクリック



デバッガ: Step Into ボタンを使うと



スタック

メソッド main のスタックフレーム

a=3

b=-1

abs_a=3

abs_b=1

sum=4

変数 sum に abs_a+abs_b の
値が代入された
ステップオーバーをクリック

練習6-b. つづき



Test06b.java

```
public class Test06b {
    public static void main( String[] args ){
        int a, b;
        int abs_a, abs_b;
        int sum;
        a = 3; b = -1;

        abs_a = abs(a);
        abs_b = abs(b);
        sum = abs_a+abs_b;
        System.out.println("sum="+sum);
    }
    public static int abs( int n ){
        int ret;
        if ( n >= 0 ){
            ret = n;
        } else {
            ret = -n;
        }
        return ret;
    }
}
```

- (1) デバッガを使ってこの2つの呼び出しに対して, Step Into を使って n と ret の値がそれぞれどうなっているのか調べよ
- (2) 変数 c と abs_c を追加し, a=-3; b=7; c=-5; としたときに, n と ret の値がどうなるか, デバッガを使って調べよ

練習6-c. べき乗の合計を計算せよ



Test06c.java

```
public class Test06c {
    public static void main( String[] args ){
        int a, b, c, n;
        int pow_a, pow_b, pow_c;
        int sum;
        a = 3; b = 7; c = 4; n = 3;

        pow_a = power(a,n);
        pow_b = power(b,n);
        pow_c = power(c,n);
        sum = pow_a+pow_b+pow_c;
        System.out.println("sum="+sum);
    }
    public static int power( int x, int n ){
        int i, ret;

        return ret;
    }
}
```

xのn乗を計算して ret に代入せよ

べき乗の合計を
計算するプログラム
を完成させよ

練習6-c. 解答



Test06c.java

```
public class Test06c {
    public static void main( String[] args ){
        int a, b, c, n;
        int pow_a, pow_b, pow_c;
        int sum;
        a = 3; b = 7; c = 4; n = 3;

        pow_a = power(a,n);
        pow_b = power(b,n);
        pow_c = power(c,n);
        sum = pow_a+pow_b+pow_c;
        System.out.println("sum="+sum);
    }
    public static int power( int x, int n ){
        int i, ret;
        ret = 1;
        for ( i = 0; i < n; i++ ){
            ret = ret * x;
        }
        return ret;
    }
}
```

べき乗の合計を
計算するプログラム
を完成させよ

練習6-d. 階乗の合計を計算せよ



Test06d.java

```
public class Test06d {
    public static void main( String[] args ){
        int a, b, c;
        int facto_a, facto_b, facto_c;
        int sum;
        a = 3; b = 7; c = 4;

        facto_a = factorial(a);
        facto_b = factorial(b);
        facto_c = factorial(c);
        sum = facto_a+facto_b+facto_c;
        System.out.println("sum="+sum);
    }
    public static int factorial( int n ){
        int ret;
        ret = 1;
        return ret;
    }
}
```

階乗 $n!$ を計算しretに代入せよ

階乗の合計を
計算するプログラム
を完成させよ

練習6-d. 解答



Test06d.java

```
public class Test06d {
    public static void main( String[] args ){
        int a, b, c;
        int facto_a, facto_b, facto_c;
        int sum;
        a = 3; b = 7; c = 4;

        facto_a = factorial(a);
        facto_b = factorial(b);
        facto_c = factorial(c);
        sum = facto_a+facto_b+facto_c;
        System.out.println("sum="+sum);
    }
    public static int factorial( int n ){
        int ret;
        ret = 1;
        while ( n > 0 ){
            ret = ret * n;
            n = n - 1;
        }
        return ret;
    }
}
```

階乗の合計を
計算するプログラム
を完成させよ

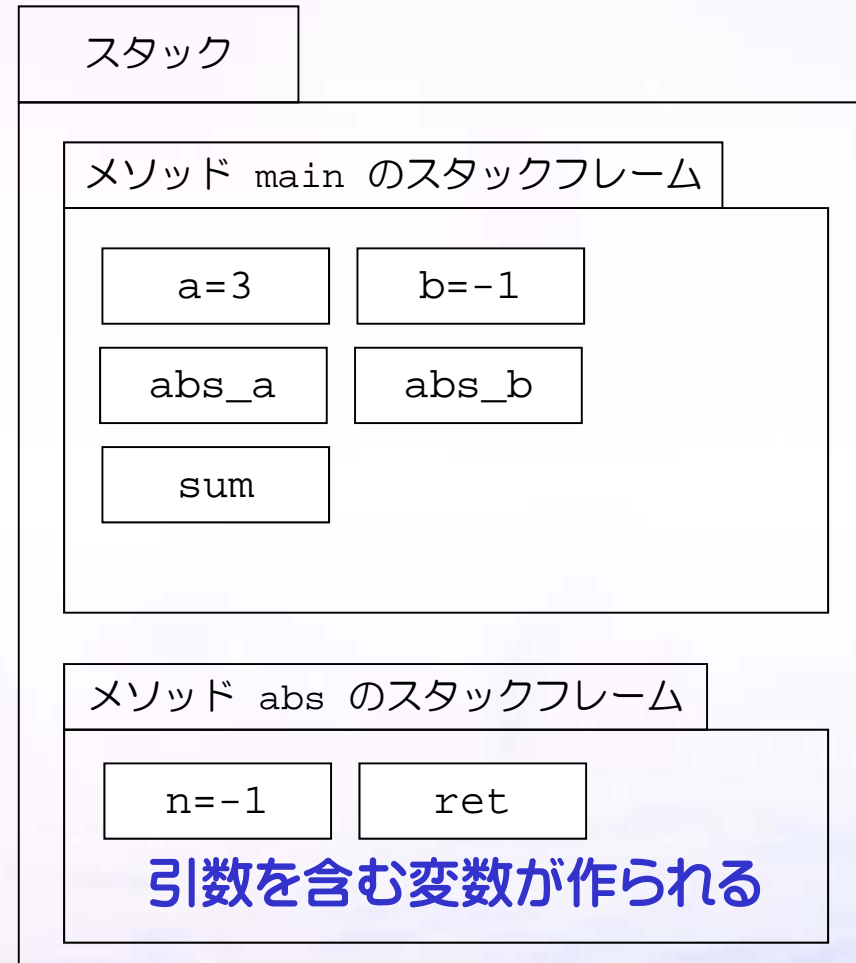
まとめ



Java文法の基本, メソッド

```
public static 戻り値の型 メソッド名(引数){  
    文;  
    return 式;  
}
```

- mainメソッド内の変数はスタックにあるmainメソッドのフレーム内に格納されている
- メソッドを呼び出すと新たなフレームができてスタックに積まれる
- メソッドを抜けるときスタックからそのフレームが取り除かれる



補足: Java の命名規則



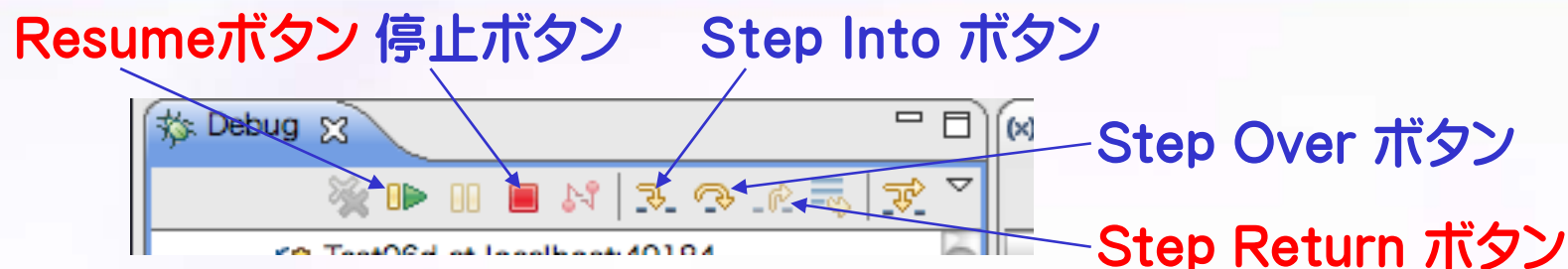
よく使われている命名規則

識別子のタイプ		
クラス名	1文字目大文字, 2文字目以降は小文字, 単語の区切り目は大文字で	<code>SplitSiteClassifier</code>
変数名, メソッド名	1文字目小文字, 2文字目以降は小文字, 単語の区切り目は大文字で	<code>setVisible(boolean),</code> <code>readSeqs(),</code> <code>dotMat</code>

<http://java.sun.com/docs/codeconv/html/CodeConventions.doc8.html>

本講義で示す例は, 別の規則にしたがっている

補足：デバツガの使い方



Resumeボタン	次のブレークポイントまで進める
停止ボタン	プログラムを終了させる
Step Into ボタン	メソッドの先頭コードの手前まで進める
Step Over ボタン	1ステップ進める
Step Return ボタン	現在のメソッドが終了して呼び出し元に戻る