

# 情報生命科学演習 プログラミングの流れ

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`

# 目標



- ・ プログラミングの流れを理解する
  - 統合開発環境Eclipseの使い方を理解する
  - 簡単なプログラムを作って練習する
  - デバッガの使い方を習得する

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

```
HelloWorld.java  
public class HelloWorld {  
    public static void main( String[] args ){  
        System.out.println("Hello World.");  
    }  
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

HelloWorld.java



バイトコード

HelloWorld.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

Hello World.

計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

```
HelloWorld.java
public class HelloWorld {
    public static void main( String[] args ){
        System.out.println("Hello World.");
    }
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

HelloWorld.java



バイトコード

HelloWorld.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

Hello World.

計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

HelloWorld.java

```
public class HelloWorld {  
    public static void main( String[] args ) {  
        System.out.println("Hello World.");  
    }  
}
```

飾り

「Hello World.」と画面に表示しなさい

ソースプログラムは計算機への命令書

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

```
HelloWorld.java  
public class HelloWorld {  
    public static void main( String[] args ){  
        System.out.println("Hello World.");  
    }  
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

HelloWorld.java



バイトコード

HelloWorld.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

Hello World.

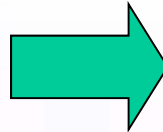
計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# プログラミングから実行までの流れ

## (2) コンパイルする

ソースプログラム

HelloWorld.java



バイトコード

HelloWorld.class

コンパイルとは命令書を計算機が分かる言葉に変換すること  
もう少し正確にいうならば、バイトコードは Java Virtual Machineが理解できる言葉

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

```
HelloWorld.java  
public class HelloWorld {  
    public static void main( String[] args ){  
        System.out.println("Hello World.");  
    }  
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

HelloWorld.java



バイトコード

HelloWorld.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

Hello World.

計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# プログラミングから実行までの流れ



## (3) 実行する

画面(標準出力)

```
Hello World.
```

計算機に分かる言葉に変換された命令書「バイトコード」に従って  
計算機が動作する

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

HelloWorld.java

```
public class HelloWorld {  
    public static void main( String[] args ){  
        System.out.println("Hello World.");  
    }  
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

HelloWorld.java



バイトコード

HelloWorld.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

Hello World.

計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

```
Test01a.java
public class Test01a {
    public static void main( String[] args ){
        int a = 5;
        int b = 3;
        int c = a + b;
        System.out.println("c="+c);
    }
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

Test01a.java



バイトコード

Test01a.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

c=8

計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

```
Test01a.java
public class Test01a {
    public static void main( String[] args ){
        int a = 5;
        int b = 3;
        int c = a + b;
        System.out.println("c="+c);
    }
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

Test01a.java



バイトコード

Test01a.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

c=8

計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# プログラミングから実行までの流れ

## (1) ソースプログラムを書く

Test01a.java

ソースプログラムのファイル名

```
public class Test01a {  
    public static void main( String[] args ) {
```

飾り

```
        int a, b, c;
```

```
        a = 5;    変数 a に値 5 を代入しろ
```

```
        b = 3;    変数 b に値 3 を代入しろ
```

```
        c = a + b; 式 a+b の値を計算して変数 c に代入しろ
```

```
        System.out.println(“c=“+c);
```

```
    }           cの値を画面に表示しろ
```

```
}
```

ソースプログラムは計算機への命令書

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

```
Test01a.java
public class Test01a {
    public static void main( String[] args ){
        int a = 5;
        int b = 3;
        int c = a + b;
        System.out.println("c="+c);
    }
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

Test01a.java



バイトコード

Test01a.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

c=8

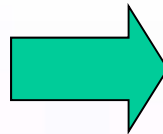
計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# プログラミングから実行までの流れ

## (2) コンパイルする

ソースプログラム

Test01a.java



バイトコード

Test01a.class

コンパイルとは命令書を計算機が分かる言葉に変換すること  
もう少し正確にいうならば、バイトコードは Java Virtual Machineが理解できる言葉

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

```
Test01a.java
public class Test01a {
    public static void main( String[] args ){
        int a = 5;
        int b = 3;
        int c = a + b;
        System.out.println("c="+c);
    }
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

Test01a.java



バイトコード

Test01a.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

c=8

計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# プログラミングから実行までの流れ



## (3) 実行する

画面(標準出力)

```
c=8
```

計算機に分かる言葉に変換された命令書「バイトコード」に従って  
計算機が動作する

# プログラミングから実行までの流れ



## (1) ソースプログラムを書く

```
Test01a.java
public class Test01a {
    public static void main( String[] args ){
        int a = 5;
        int b = 3;
        int c = a + b;
        System.out.println("c="+c);
    }
}
```

ソースプログラムは  
計算機への命令書

## (2) コンパイルする

ソースプログラム

Test01a.java



バイトコード

Test01a.class

コンパイルとは命令書を計算機が  
分かる言葉に変換すること

## (3) 実行する

画面(標準出力)

c=8

計算機に分かる言葉に変換された  
命令書に従って計算機が動作する

# 統合開発環境 Eclipse を使うと



## (1) ソースプログラムを書く

```
HelloWorld.java  
public class HelloWorld {  
    public static void main( String[] args ) {  
        System.out.println("Hello World.");  
    }  
}
```

飾りの部分はEclipseが自動的に生成してくれる

## (2) コンパイルする

ソースプログラム

HelloWorld.java



バイトコード

HelloWorld.class

コンパイルと実行がワンタッチでできる

## (3) 実行する

画面(標準出力)

Hello World.

では、実際に試してみましよう。

# 統合開発環境 Eclipse を使うと



## (1) ソースプログラムを書く

```
HelloWorld.java  
public class HelloWorld {  
    public static void main( String[] args ) {  
        System.out.println("Hello World.");  
    }  
}
```

飾りの部分はEclipseが自動的に生成してくれる

## (2) コンパイルする

ソースプログラム

HelloWorld.java



バイトコード

HelloWorld.class

コンパイルと実行がワンタッチでできる

## (3) 実行する

画面(標準出力)

Hello World.

# 統合開発環境 Eclipse を使うと



## (1) ソースプログラムを書く

HelloWorld.java

```
public class HelloWorld {  
    public static void main( String[] args ) {  
        System.out.println("Hello World.");  
    }  
}
```

飾りの部分はEclipseが自動的に生成してくれる

## (2,3) コンパイル&実行

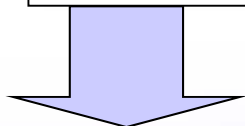
コンパイルと実行がワンタッチでできる

Eclipse内部でいろんなタイミングでコンパイルしているようである

# Eclipseでのプログラミングの流れ



(1) ソースプログラムを書く



(2,3) コンパイル&実行

# Eclipseでのプログラミングの流れ




Eclipseを起動する

プロジェクトを新規作成する

(1) ソースプログラムを書く

(2,3) コンパイル&実行

# HelloWorldプログラムを作ってみよう



プロジェクト名: joseien-h22

HelloWorld.java

```
public class HelloWorld {  
    public static void main( String[] args ){  
        System.out.println("Hello World.");  
    }  
}
```

# Eclipseでのプログラミングの流れ



Eclipseを起動する

プロジェクトを新規作成する

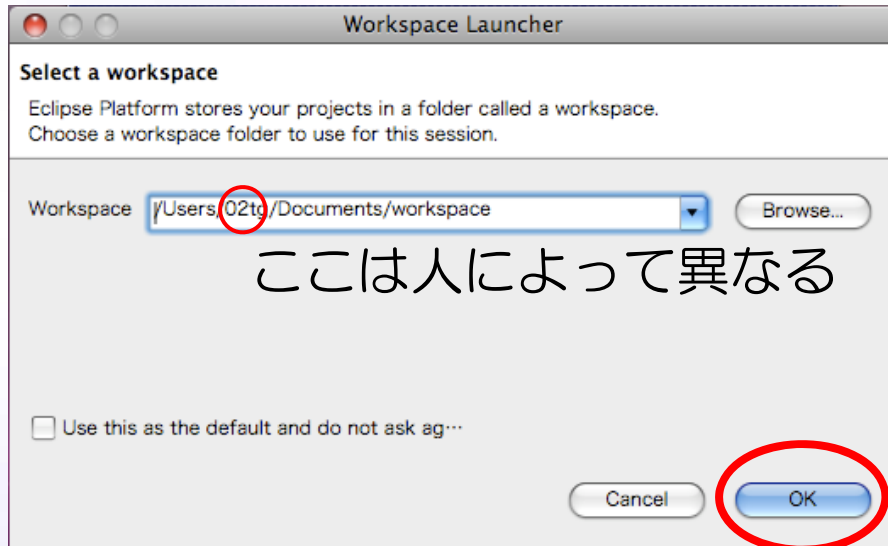
(1) ソースプログラムを書く

(2,3) コンパイル&実行

# Eclipseの起動



↑画面の下にあるこのアイコンをクリックする



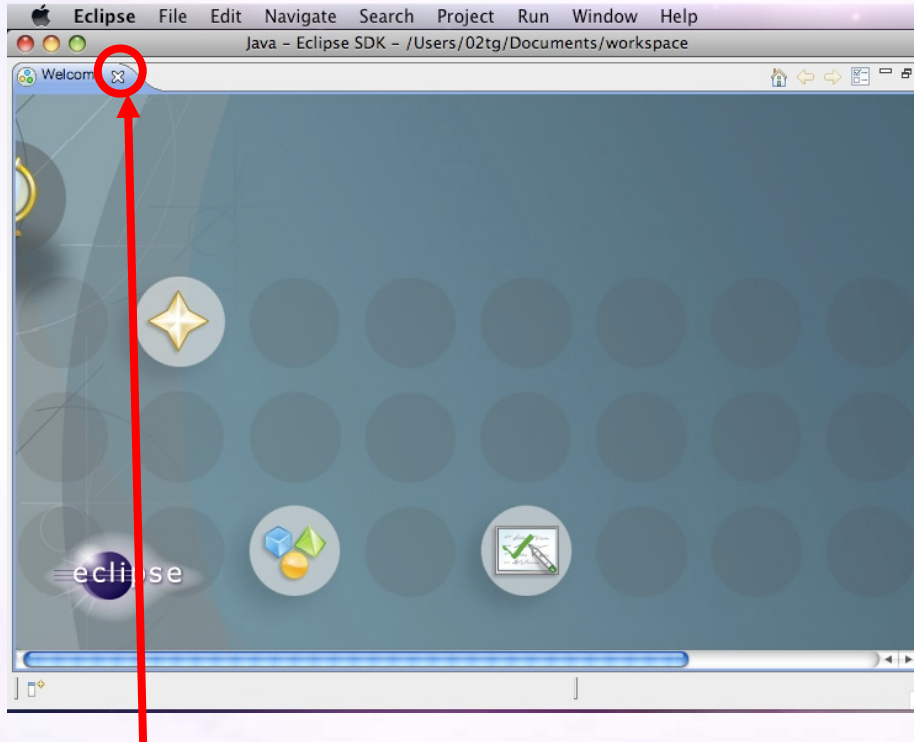
するとこんなダイアログ  
が表示される

そのまま「OK」をクリック

# Eclipseの起動



このウィンドウが現れればEclipse の起動が成功



「Welcome」の画面は使わないので、  
この「X」を押して消しておく

# Eclipseでのプログラミングの流れ



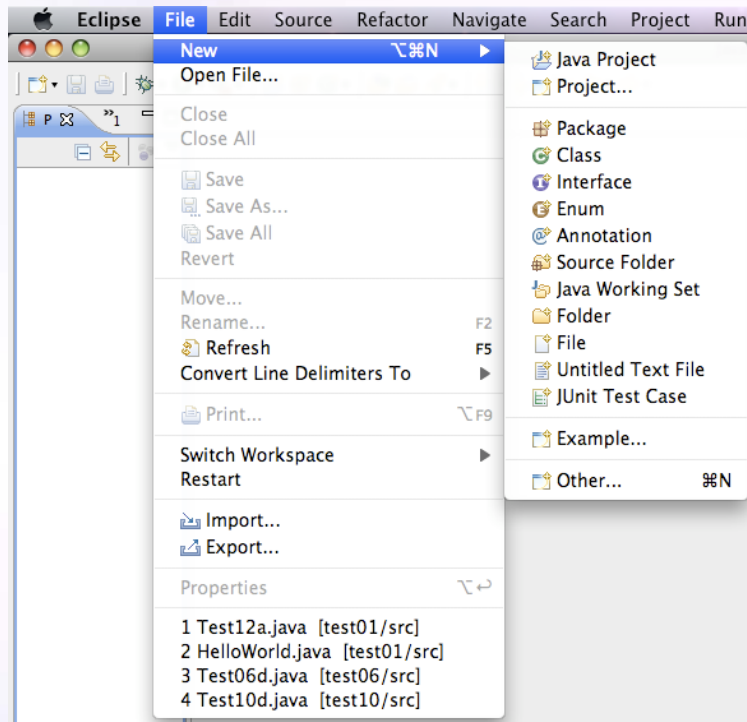
Eclipseを起動する

プロジェクトを新規作成する

(1) ソースプログラムを書く

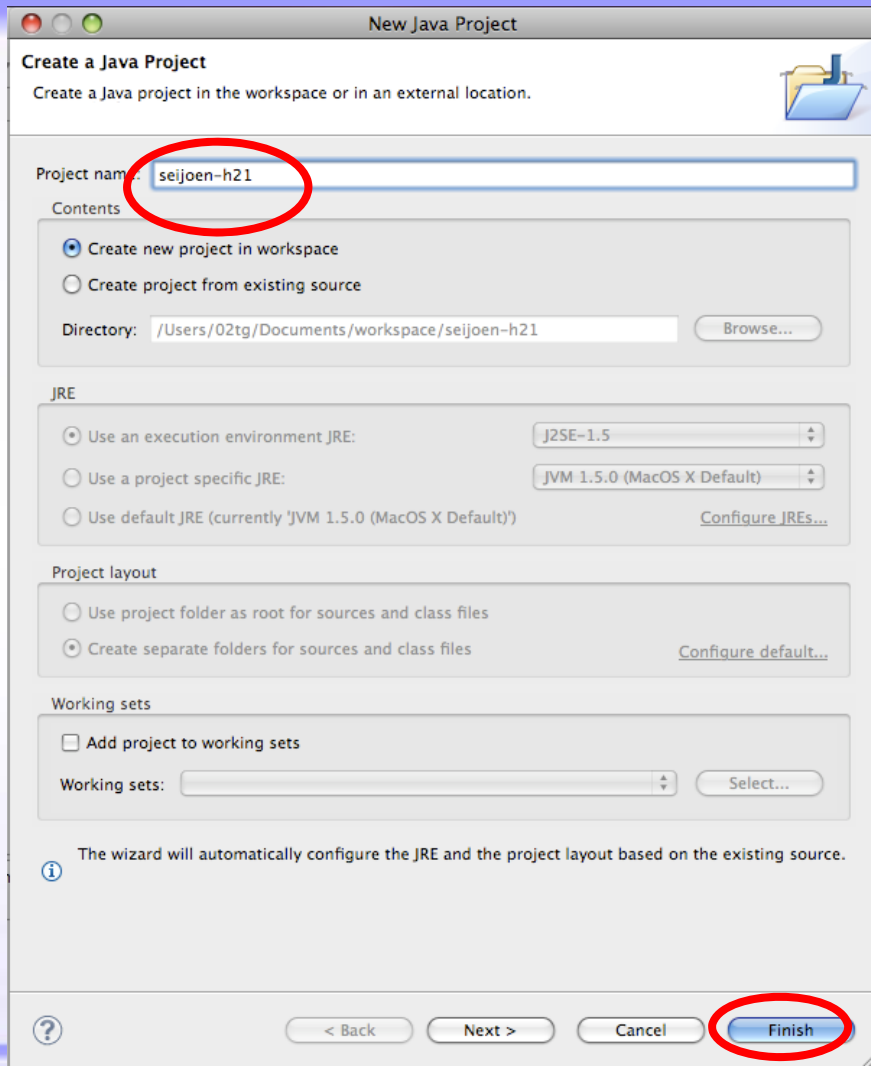
(2,3) コンパイル&実行

# プロジェクトの新規作成



メニューバーから  
「File」「New」  
「Java Project」  
と順にクリック

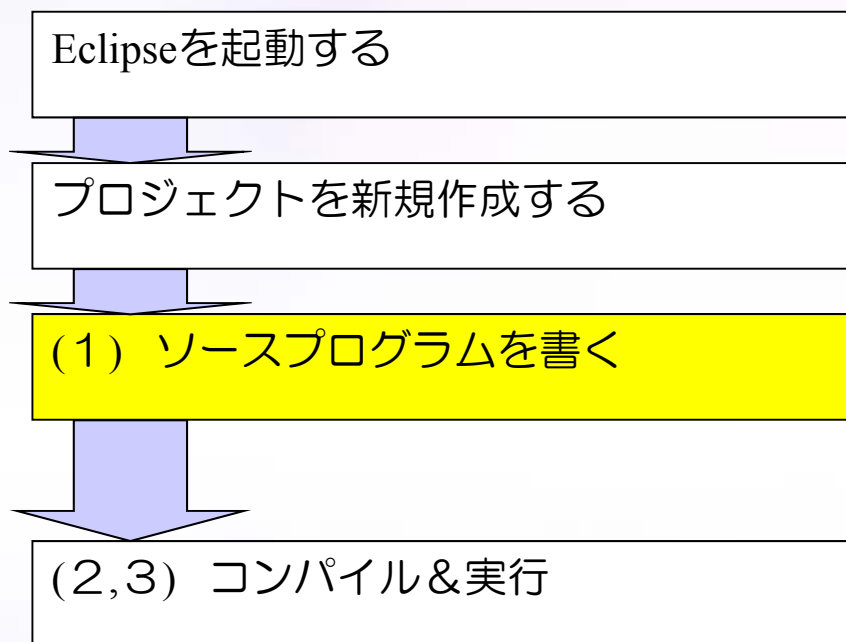
# プロジェクトの新規作成



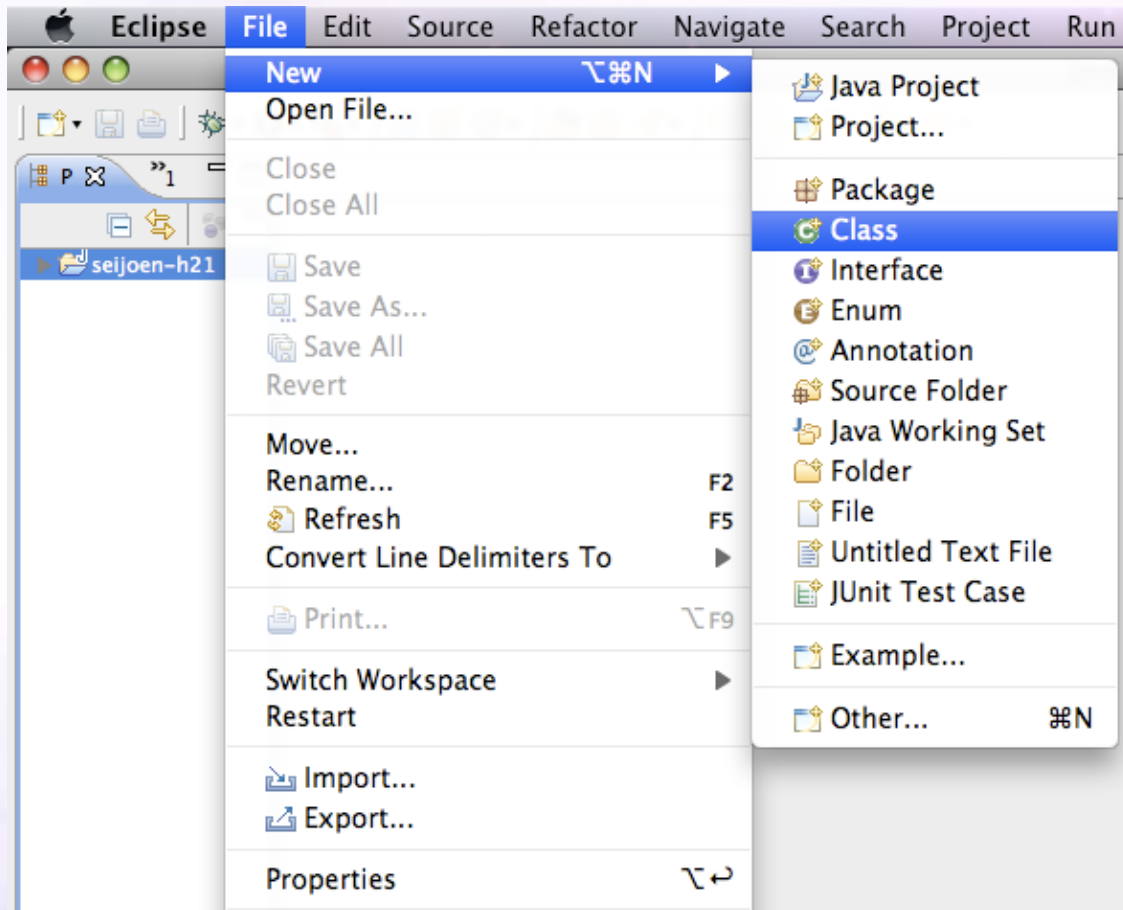
Project nameを  
「joseien-h22」  
とする

「Finish」をクリック

# Eclipseでのプログラミングの流れ

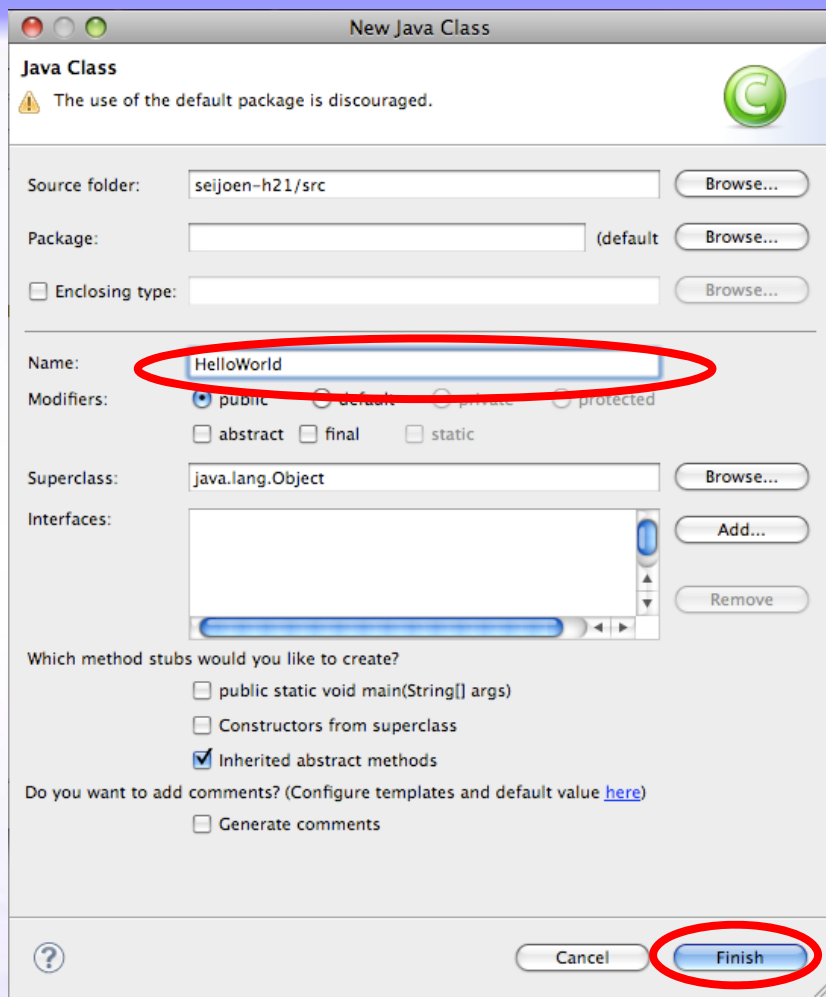


# ソースプログラムを書く



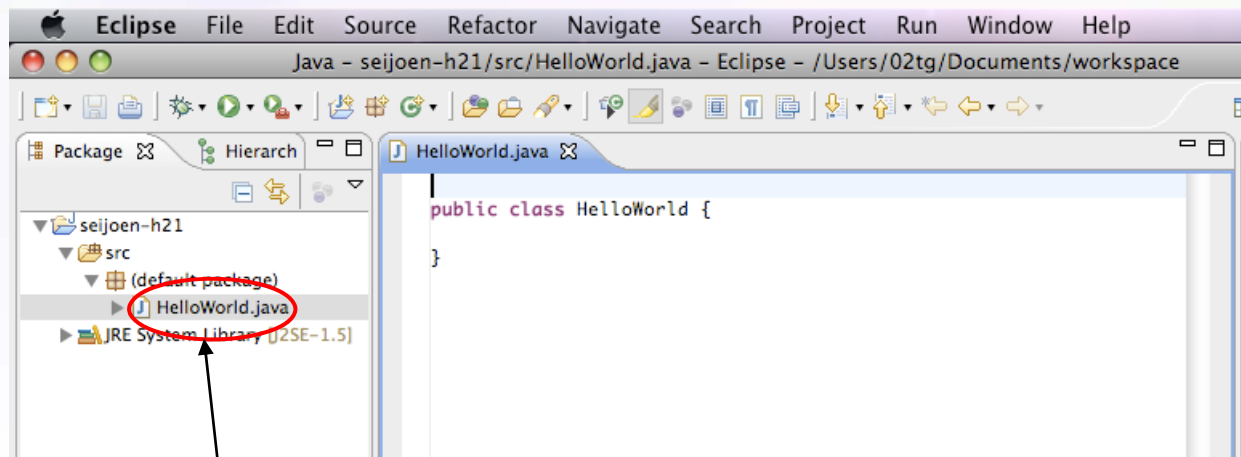
メニューバーから  
「File」 「New」  
「Class」  
と順にクリック

# ソースプログラムを書く



Nameに  
「HelloWorld」  
と入力する

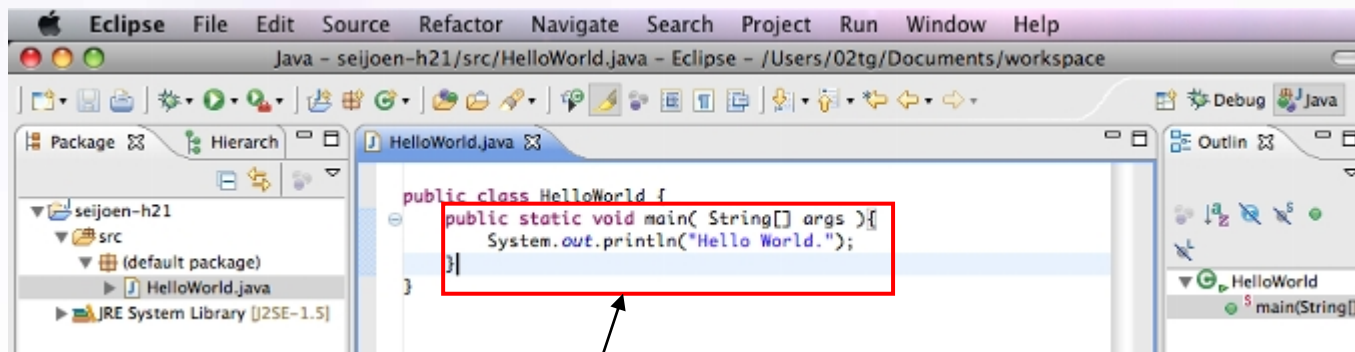
# ソースプログラムを書く



ソースファイル「HelloWorld.java」が新たにできている。

中身はまだほとんど空。

# ソースプログラムを書く



HelloWorld.java

```
public class HelloWorld {  
    public static void main( String[] args ) {  
        System.out.println("Hello World.");  
    }  
}
```

# Eclipseでのプログラミングの流れ



Eclipseを起動する

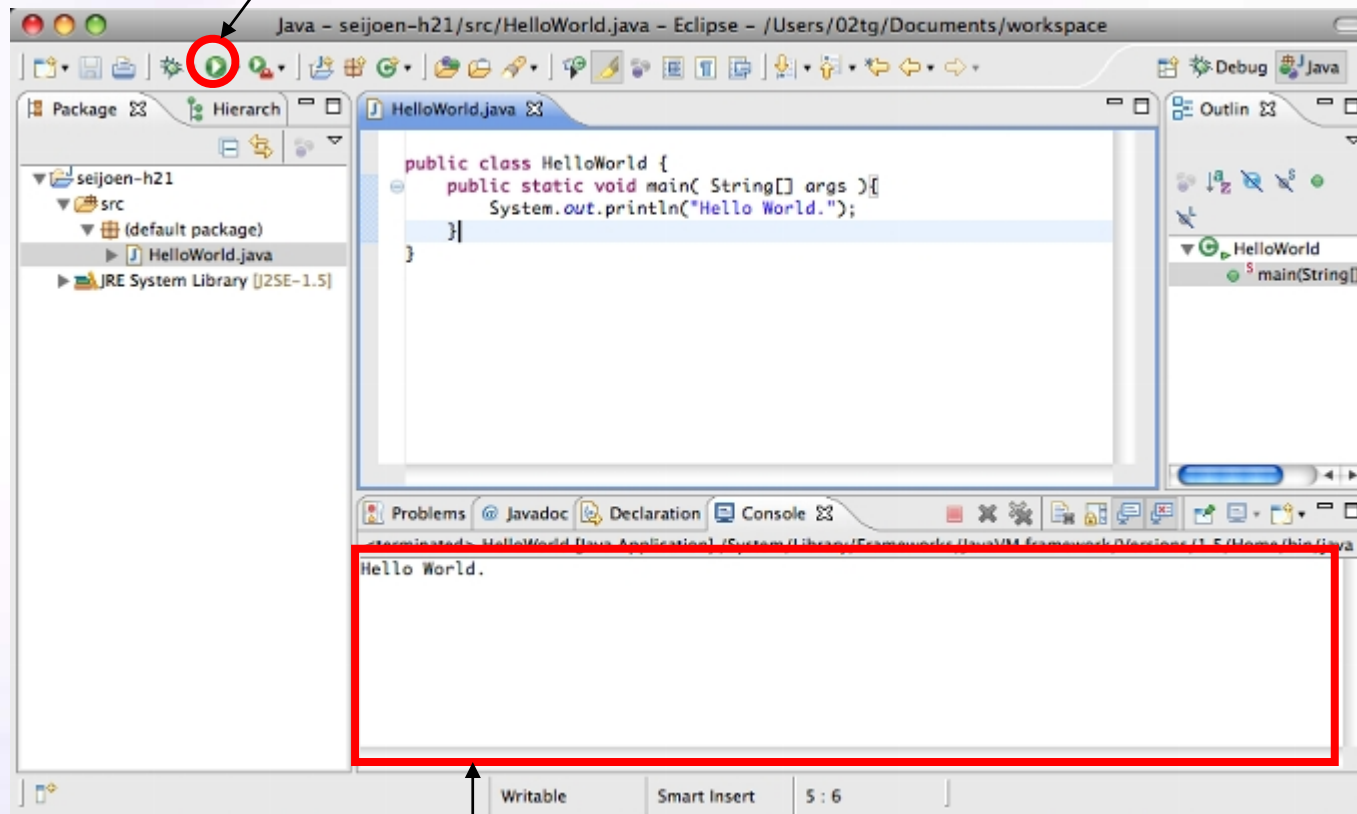
プロジェクトを新規作成する

(1) ソースプログラムを書く

(2,3) コンパイル&実行

# コンパイル & 実行

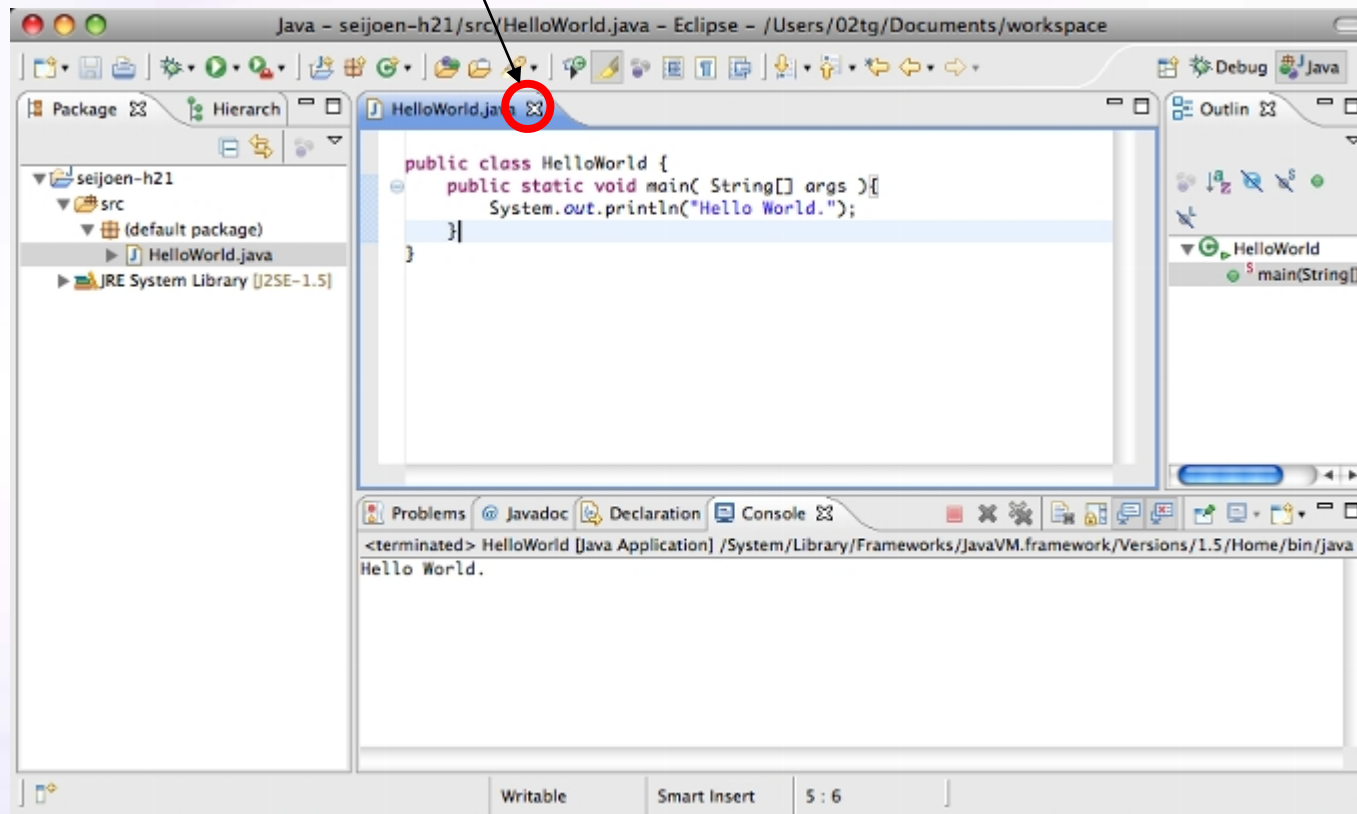
このボタンを押す。するとプログラムが実行される



画面への出力はこの部分に表示される

# 終わったら閉じる

このボタンを押す。するとソースコードが閉じられる





これで、HelloWorld.java のプログラミングは  
完了です

# 練習1-a. 足し算のプログラム



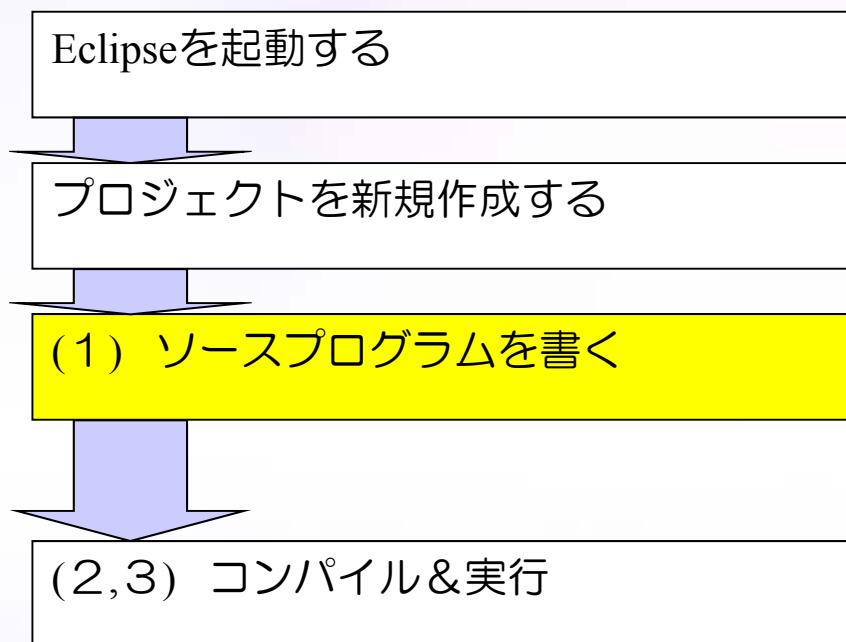
プロジェクト名: test01

Test01a.java

```
public class Test01a {  
    public static void main( String[] args ){  
        int a, b, c;  
        a = 5;    変数 a に値 5 を代入しろ  
        b = 3;    変数 b に値 3 を代入しろ  
        c = a + b; 式 a+b の値を計算して変数 c に代入しろ  
        System.out.println("c="+c);  
    }    cの値を画面に表示しろ  
}
```

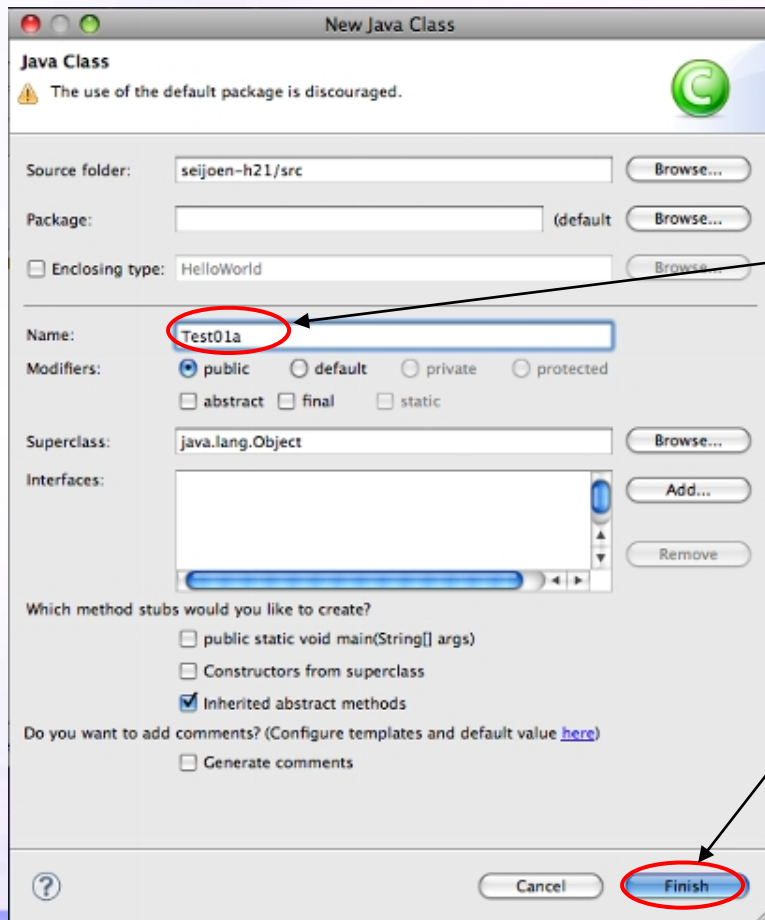
これから上のプログラムを作成していきます

# Eclipseでのプログラミングの流れ



# ソースファイル「Test01a.java」を作成する

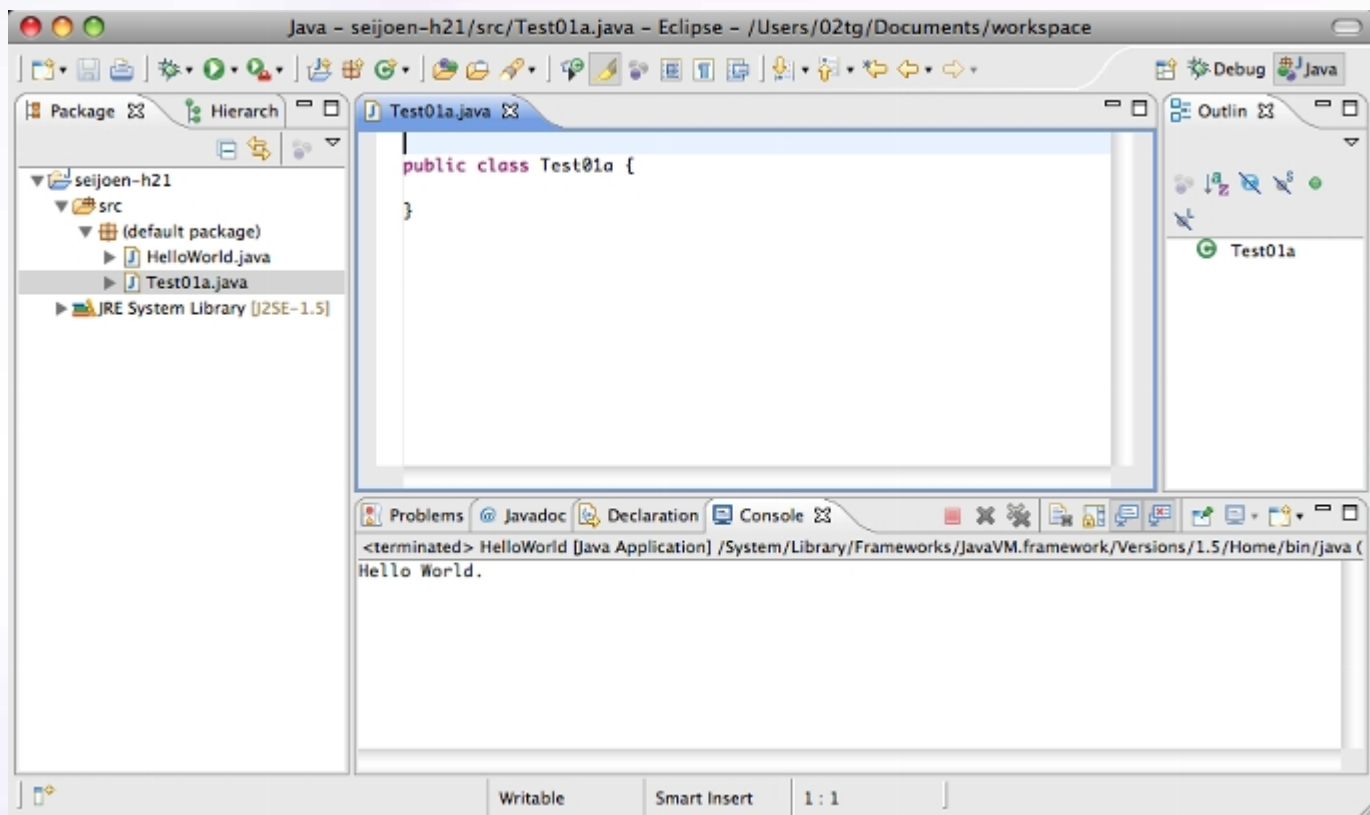
メニューバーから「File」「New」「Class」と順にクリック



今回は  
Nameに「Test01a」と入力

「Finish」をクリック

# ソースファイル「Test01a.java」を作成する



すると  
中身がほとんど空の  
「Test01a.java」が  
作成される

# ソースファイル「Test01a.java」を作成する

ソースファイル「Test01a.java」にソースプログラムを打ち込む

Test01a.java

```
public class Test01a {  
    public static void main( String[] args ) {  
        int a, b, c;  
        a = 5;  
        b = 3;  
        c = a + b;  
        System.out.println("c="+c);  
    }  
}
```

セミコロンを忘れず

# Eclipseでのプログラミングの流れ



Eclipseを起動する

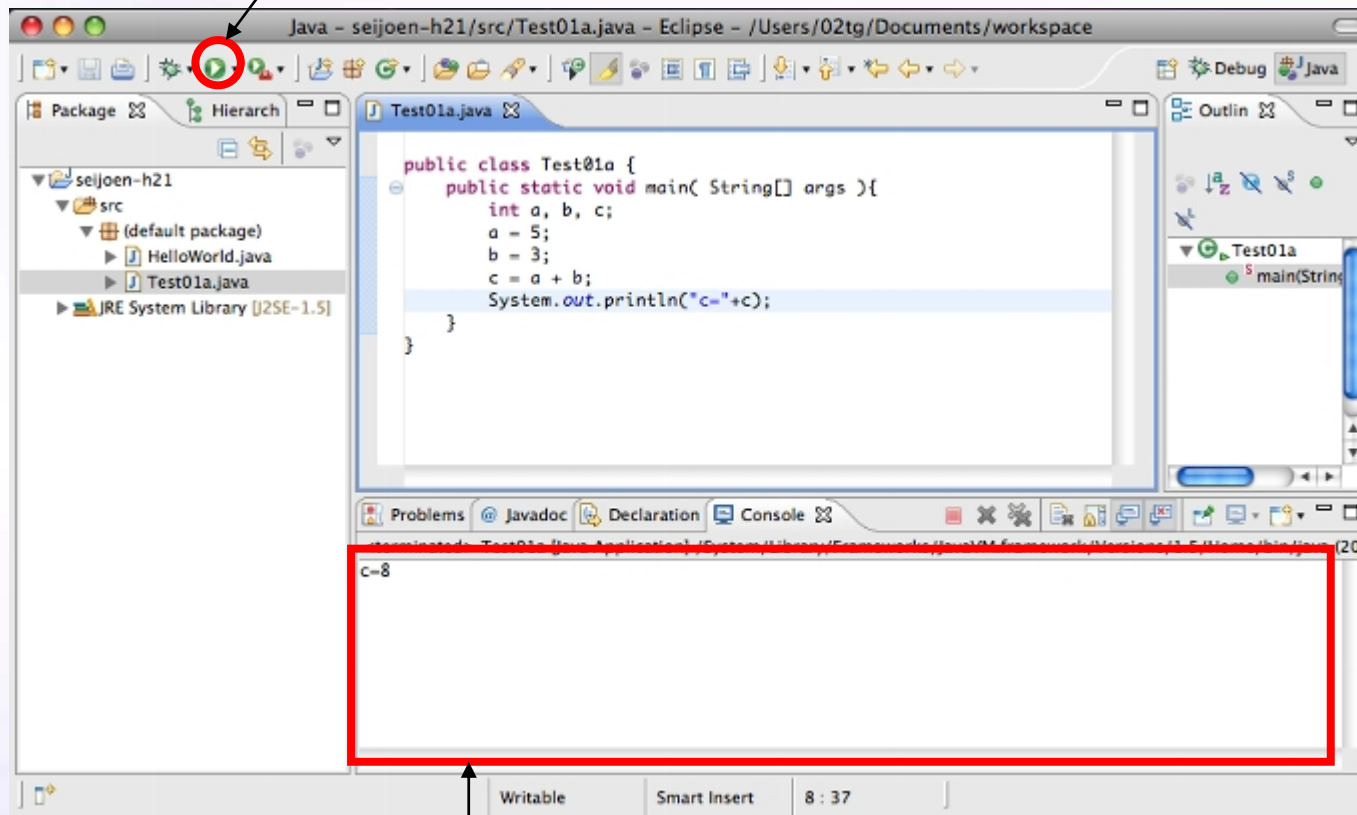
プロジェクトを新規作成する

(1) ソースプログラムを書く

(2,3) コンパイル&実行

# プログラム「Test01a.java」を実行する

このボタンを押す. するとプログラムが実行される



画面への出力はこの部分に表示される

# 練習1-b. 掛け算のプログラム

プロジェクト名: joseien-h22 練習1-aと同じプロジェクトに作る  
新たなファイル Test01b.java を新たに作る

Test01b.java

```
public class Test01b {  
    public static void main( String[] args ){  
        int a, b, c; 変数 a, b, c を使うと宣言  
        a = 5;      変数 a に値 5 を代入しろ  
        b = 3;      変数 b に値 3 を代入しろ  
        c = a * b;  積 a*b の値を計算して変数 c に代入しろ  
        System.out.println(“c=“+c);  
    }                cの値を画面に表示しろ  
}
```

# Test01b.java を Eclipseで作ってみよう

Eclipseを起動する

プロジェクトを新規作成する

(1) ソースプログラムを書く

(2,3) コンパイル&実行

このステップからはじめる。  
つまりメニューバーから  
File-New-Class  
とクリックして...

# 練習1-b. 掛け算のプログラム



Test01b.java

```
public class Test01b {  
    public static void main( String[] args ){  
        int a, b, c;  
        a = 5;  
        b = 3;  
        c = a * b;  
        System.out.println("c="+c);  
    }  
}
```

画面

c=15

このように表示されれば成功

# プログラムの構造



クラス名.java

```
public class クラス名 {  
    public static void main( String[] args ) {  
        変数宣言;  
        命令1;  
        命令2;  
        . . . ;  
    }  
}
```

実際には、変数はメソッドの最初で宣言しなくても  
使う直前までに宣言すればよいことになっている

# プログラムの構造



クラス名.java

```
public class クラス名 {
    public static void main( String[] args ){
        変数宣言;
        命令1;
        命令2;
        . . . ;
    }
}
```

例

Test01b.java

```
public class Test01b {
    public static void main( String[] args ){
        int a, b, c; 変数 a, b, c を使うと宣言
        a = 5;       変数 a に値 5 を代入しろ
        b = 3;       変数 b に値 3 を代入しろ
        c = a * b;   積 a*b の値を計算して変数 c に代入しろ
        System.out.println("c="+c);
    }               cの値を画面に表示しろ
}
```

# 練習1-c. 合計のプログラム



Test01c.java

改行はしてもしなくてもよい

```
public class Test01c {  
    public static void main( String[] args ){  
        int a, b, c, d, e, f, sum;  
        a = 3; b = 1; c = 5; d = 4; e = 7; f = 6;  
        sum = a + b + c ここを埋める  
        System.out.println("sum="+sum);  
    }  
}
```

3, 1, 5, 4, 7, 6 の合計を表示するように、プログラムを完成させよ

# 練習1-d. べき乗のプログラム



Test01d.java

a の 5 乗を計算する

(1) 実際に作成して実行結果が正しく出力されることを確認せよ

```
public class Test01d {  
    public static void main( String[] args ) {  
        int a, pow_a;  
        a = 2;  
        pow_a = 1;  
        pow_a = pow_a*a;  
        pow_a = pow_a*a;  
        pow_a = pow_a*a;  
        pow_a = pow_a*a;  
        pow_a = pow_a*a;  
        System.out.println( "pow_a="+pow_a );  
    }  
}
```

(2) a = 2; のところ値を変更してみよ. 変更しても正しく計算されることを確認せよ

式  $pow\_a * a$  の値を計算して変数  $pow\_a$  に代入  
式  $pow\_a * a$  の値を計算して変数  $pow\_a$  に代入  
式  $pow\_a * a$  の値を計算して変数  $pow\_a$  に代入  
式  $pow\_a * a$  の値を計算して変数  $pow\_a$  に代入  
式  $pow\_a * a$  の値を計算して変数  $pow\_a$  に代入

aの値を画面に表示しろ

# 練習1-e. 階乗のプログラム



Test01e.java

6! を計算する

```
public class Test01e {  
    public static void main( String[] args ){  
        int a;  
        a = 6;      変数 a に値 6 を代入しろ  
        a = a*5;    式 a*5 の値を計算して変数 a に代入しろ  
        a = a*4;    式 a*4 の値を計算して変数 a に代入しろ  
        a = a*3;    式 a*3 の値を計算して変数 a に代入しろ  
        a = a*2;    式 a*2 の値を計算して変数 a に代入しろ  
        a = a*1;    式 a*1 の値を計算して変数 a に代入しろ  
        System.out.println(“a="+a);  
    }              aの値を画面に表示しろ  
}
```

(1) 実際に作成して実行結果が正しく出力されることを確認せよ

# まとめ



- ・ プログラミングの流れを学習した
  - 統合開発環境Eclipseの使い方を学習した
  - 簡単なプログラムを作って練習した

## プログラムの構造

クラス名.java

```
public class クラス名 {  
    public static void main( String[] args ) {  
        変数宣言;  
        命令1;  
        命令2;  
        . . . ;  
    }  
}
```