

Java

クラスとオブジェクト指向

18 時間目

public変数 vs private変数 vs ローカル変数

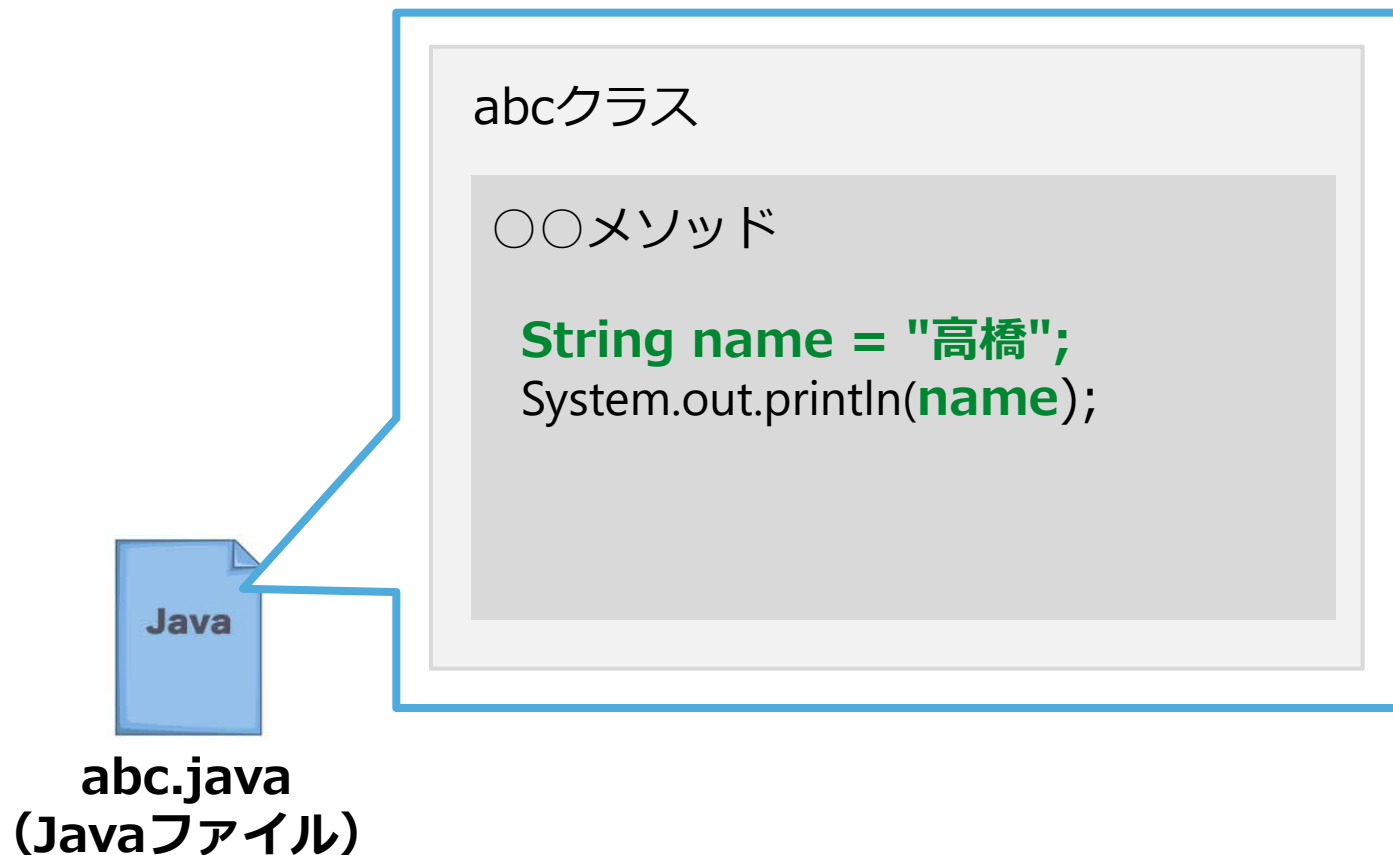
Javaの変数には3タイプの変数がある

- 1 ローカル変数
- 2 private変数(プライベート変数)
- 3 public変数(パブリック変数)

ローカル変数

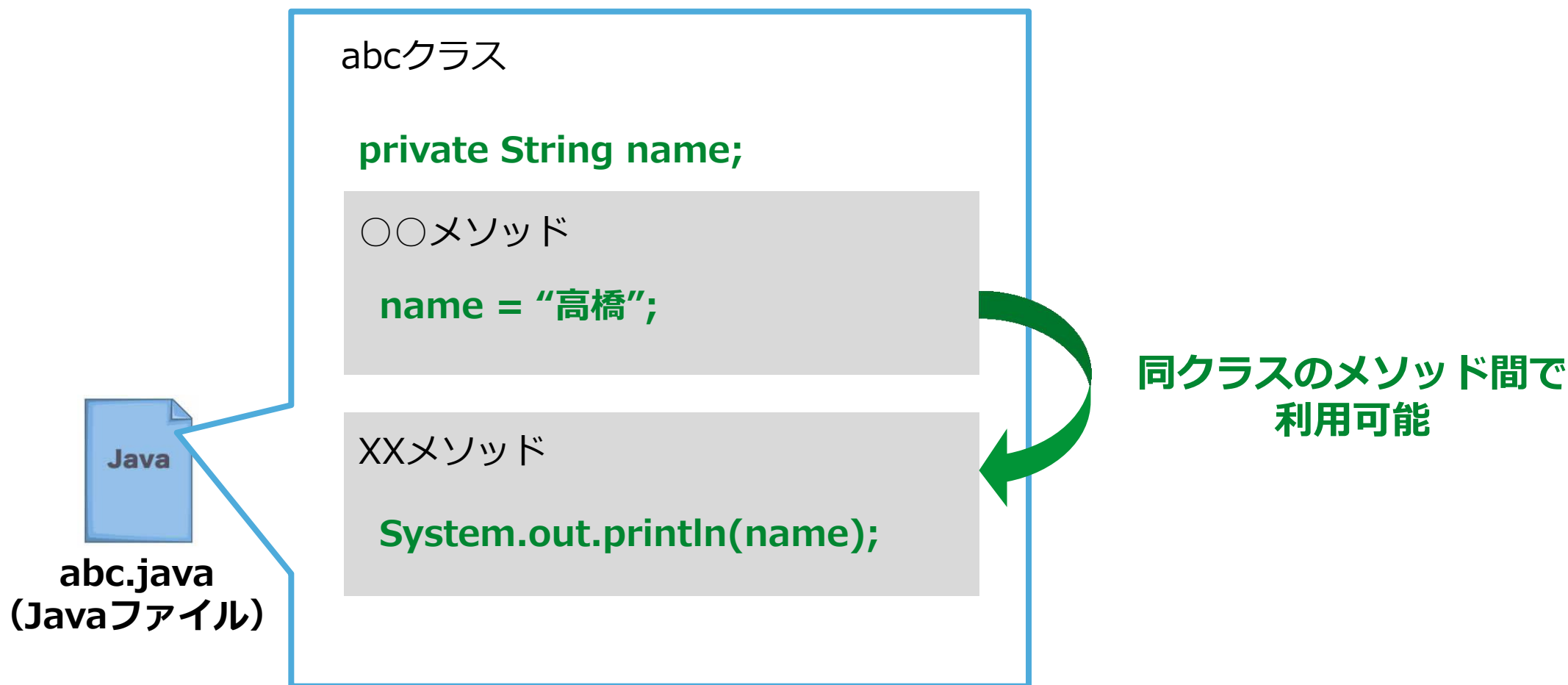
ローカル変数: そのメソッドの中でしか利用できない変数

※メソッド内で宣言した変数のことをローカル変数と言います。



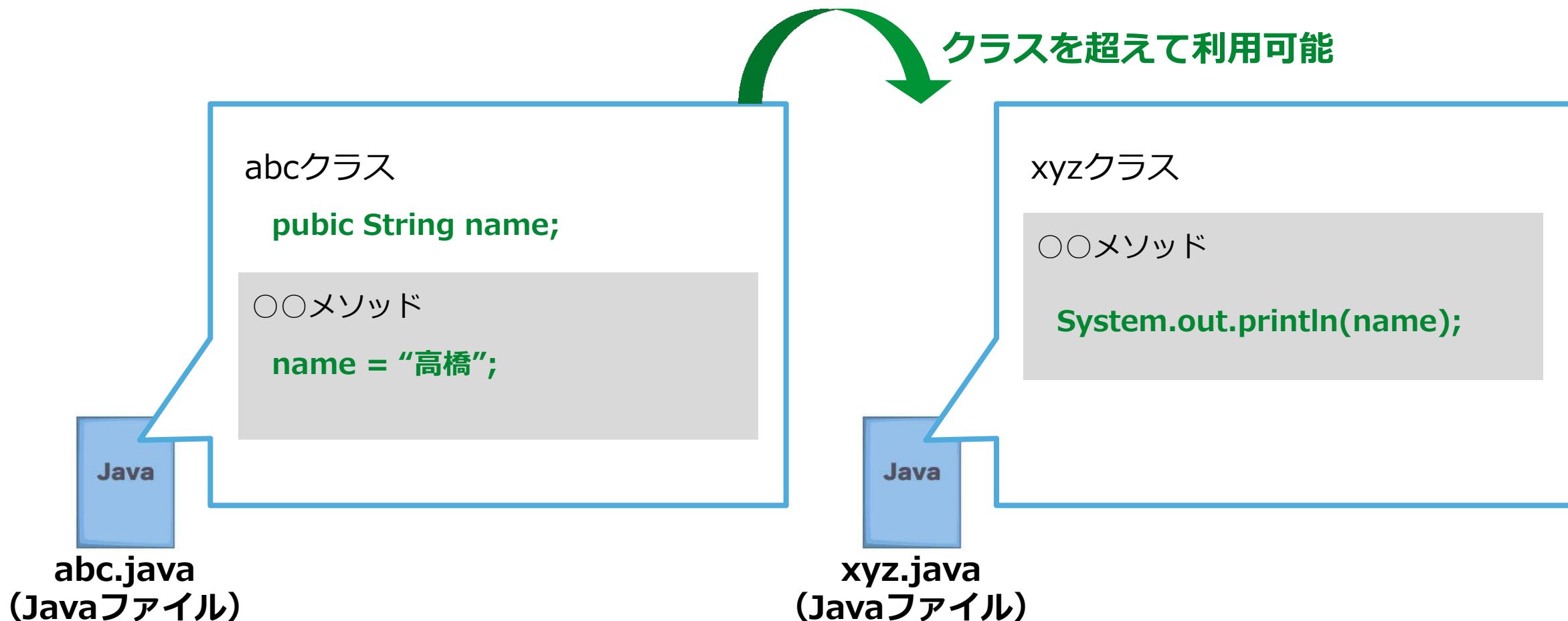
private変数（プライベート変数）

private変数: そのメソッドを超えて（メソッド間で）利用できる変数



public変数（パブリック変数）

public変数: そのクラスを超えて（クラス間で）利用できる変数

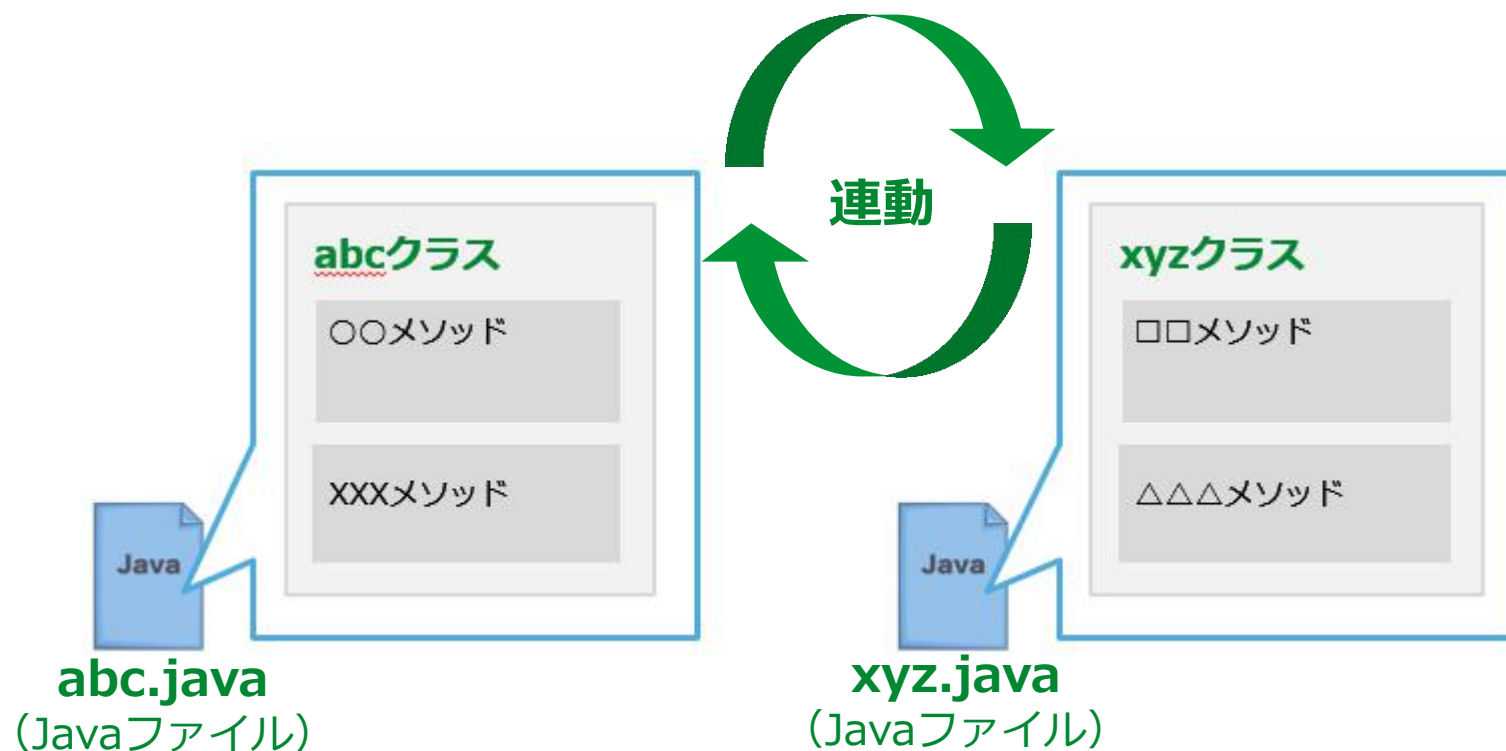


クラスとクラスを連動

解説

前回までは、メソッド（=JavaScriptやPHPの関数に似たモノ）を勉強してきました。
ここからは、クラス(=1つのJavaファイルに原則1つあるモノ) について深く学びます。

Javaプログラミングでは、下記のように複数のクラスを連動させて動かします



今回は『human』というプロジェクトの中に、
『Human』という名前のクラスを作りながら勉強していきます。

新規で『human』プロジェクトを作成

ファイル(F) 編集(E) 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)

① 『ファイル』をクリック

② 『新規』を選択

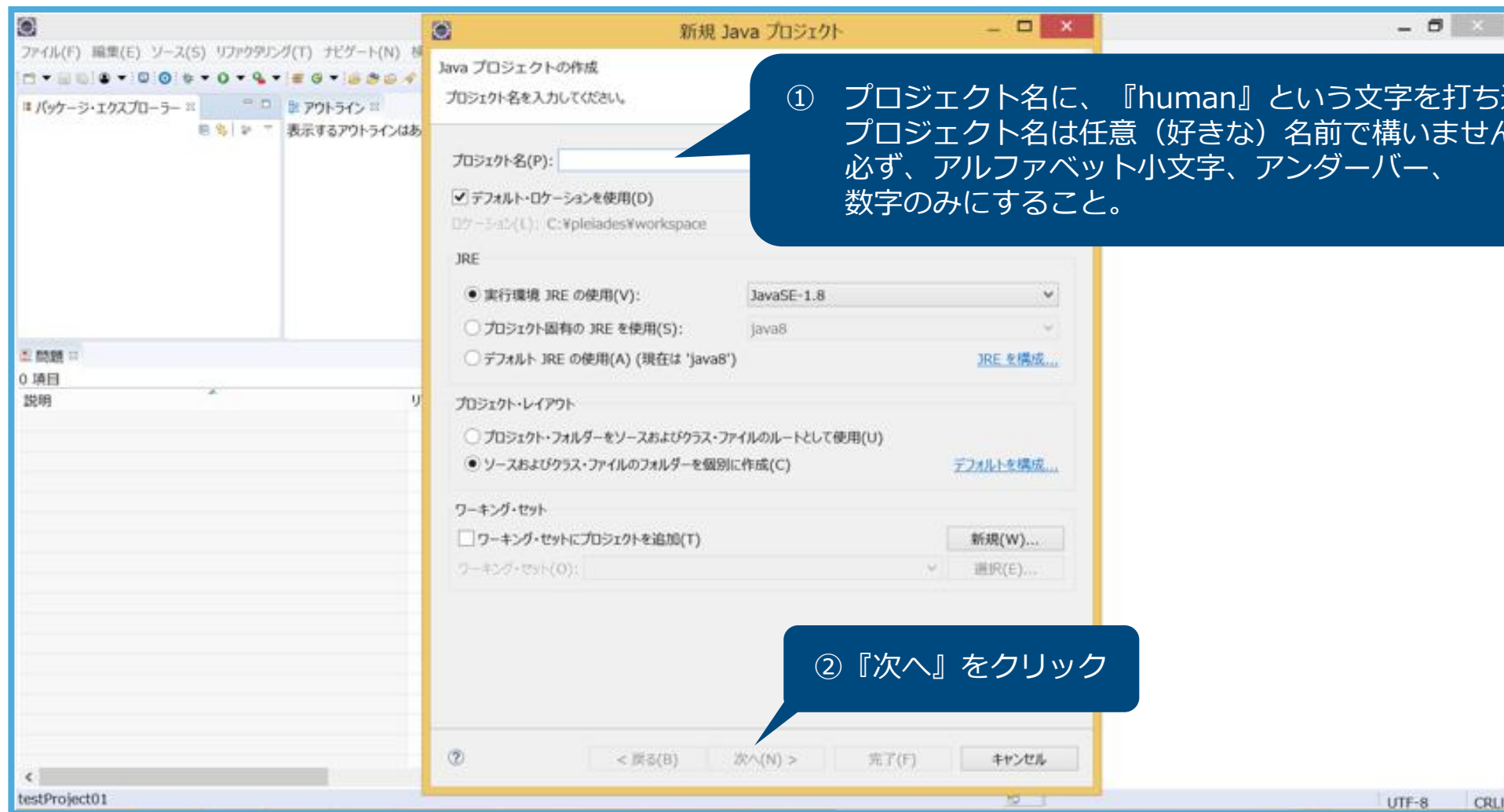
③ 『Javaプロジェクト』をクリック

Javaプロジェクト
Maven プロジェクト
Gradle プロジェクト
プロジェクト(R)...

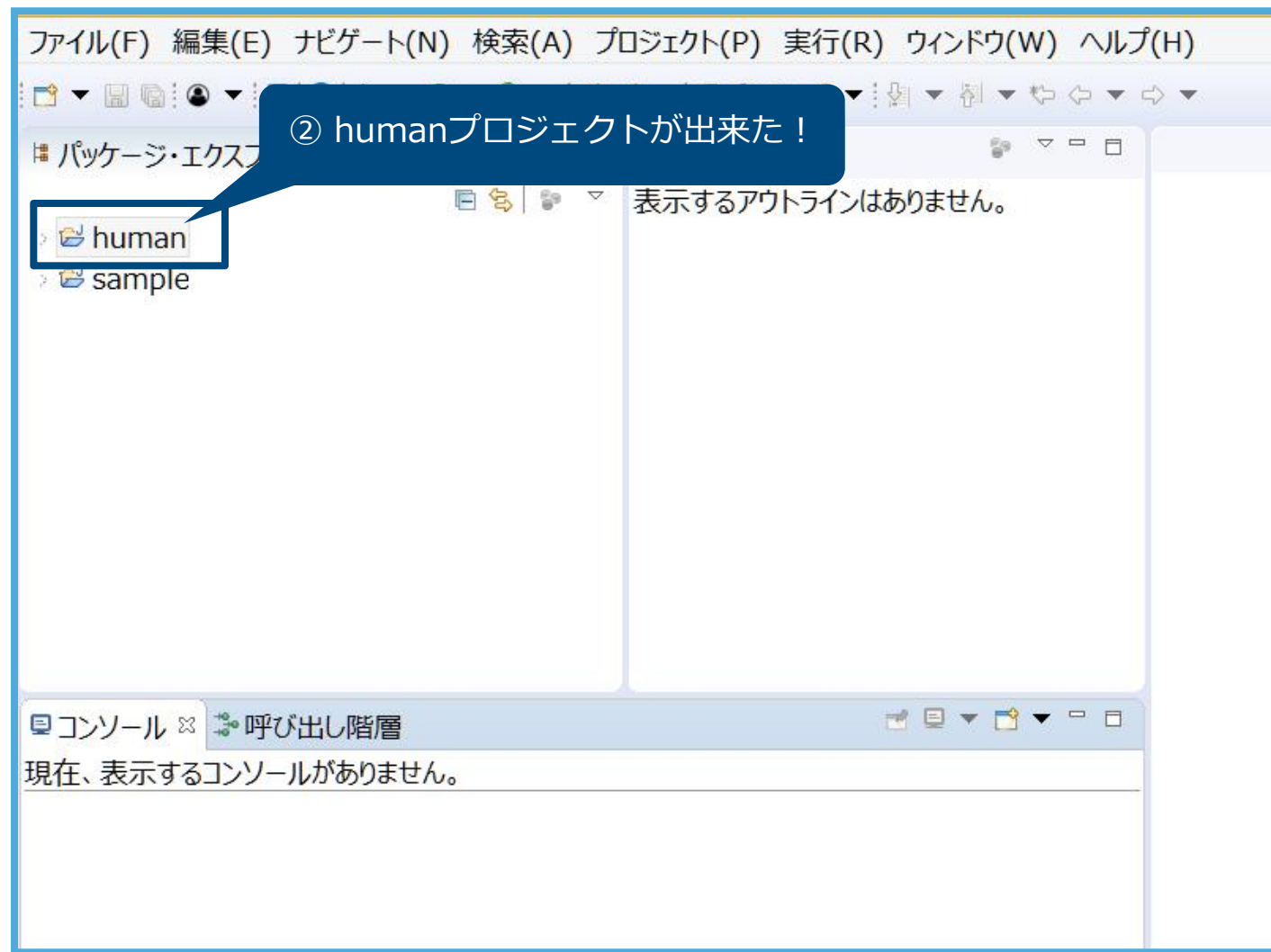
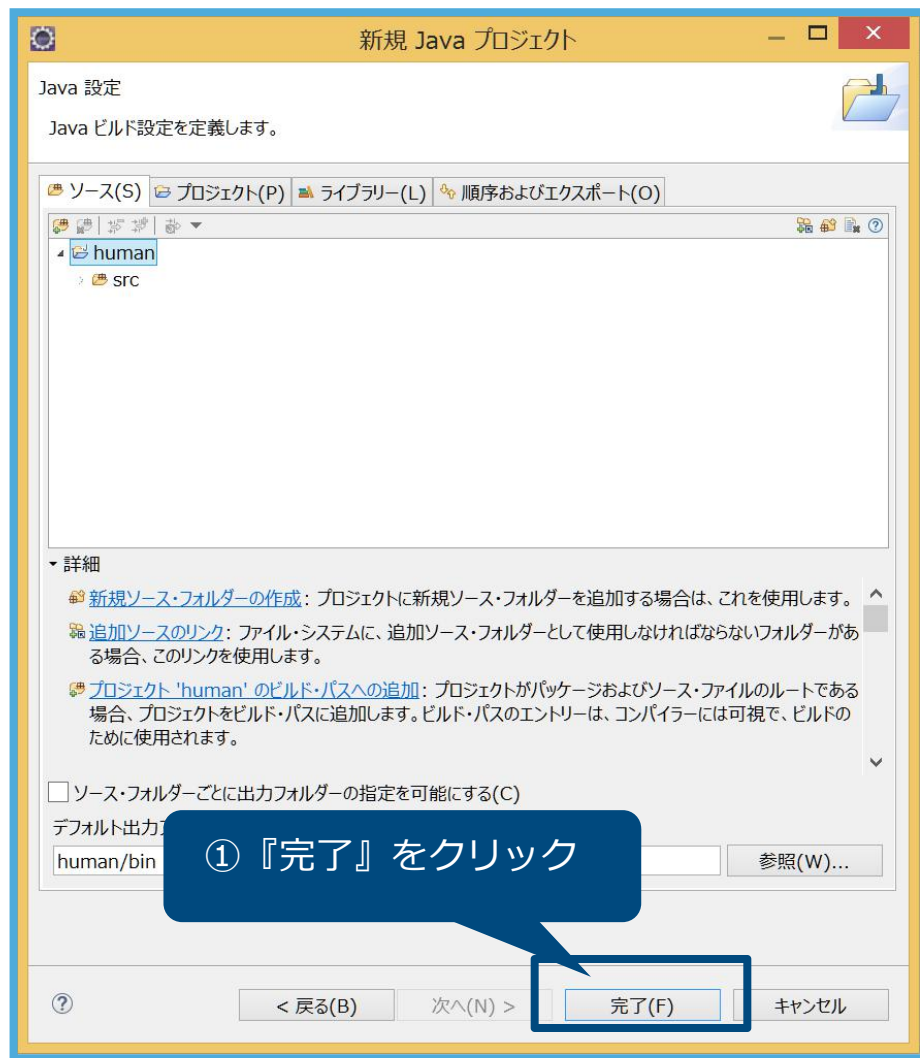
パッケージ・エクスプローラ
sample

コンソール
現在、表示するコンソールがありません。

新規で『human』プロジェクトを作成



新規で『human』プロジェクトを作成



パッケージを作成

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)

① 『src』 を右クリック

② 『新規』 を選択

③ 『パッケージ』 をクリック

現在、表示するコンソールがありません。

パッケージを作成

The screenshot shows the '新規 Java パッケージ' (New Java Package) dialog box. It has a yellow title bar and a light gray body. The text inside says 'Java パッケージ' and '新規 Java パッケージを作成します。' (Create a new Java package). Below that, it says 'パッケージに対応するフォルダーを作成します。' (Create a folder corresponding to the package). There are two input fields: 'ソース・フォルダー(D):' with the value 'human/src' and a '参照(O)...' button; and '名前(M):' with the value 'jp.co.internous.action'. Below the name field is a checkbox labeled 'package-info.java を作成する(C)' which is currently unchecked. At the bottom are two buttons: '完了(F)' (Finish) and 'キャンセル' (Cancel). There is also a help icon (?) in the bottom left corner.

① このままで良い

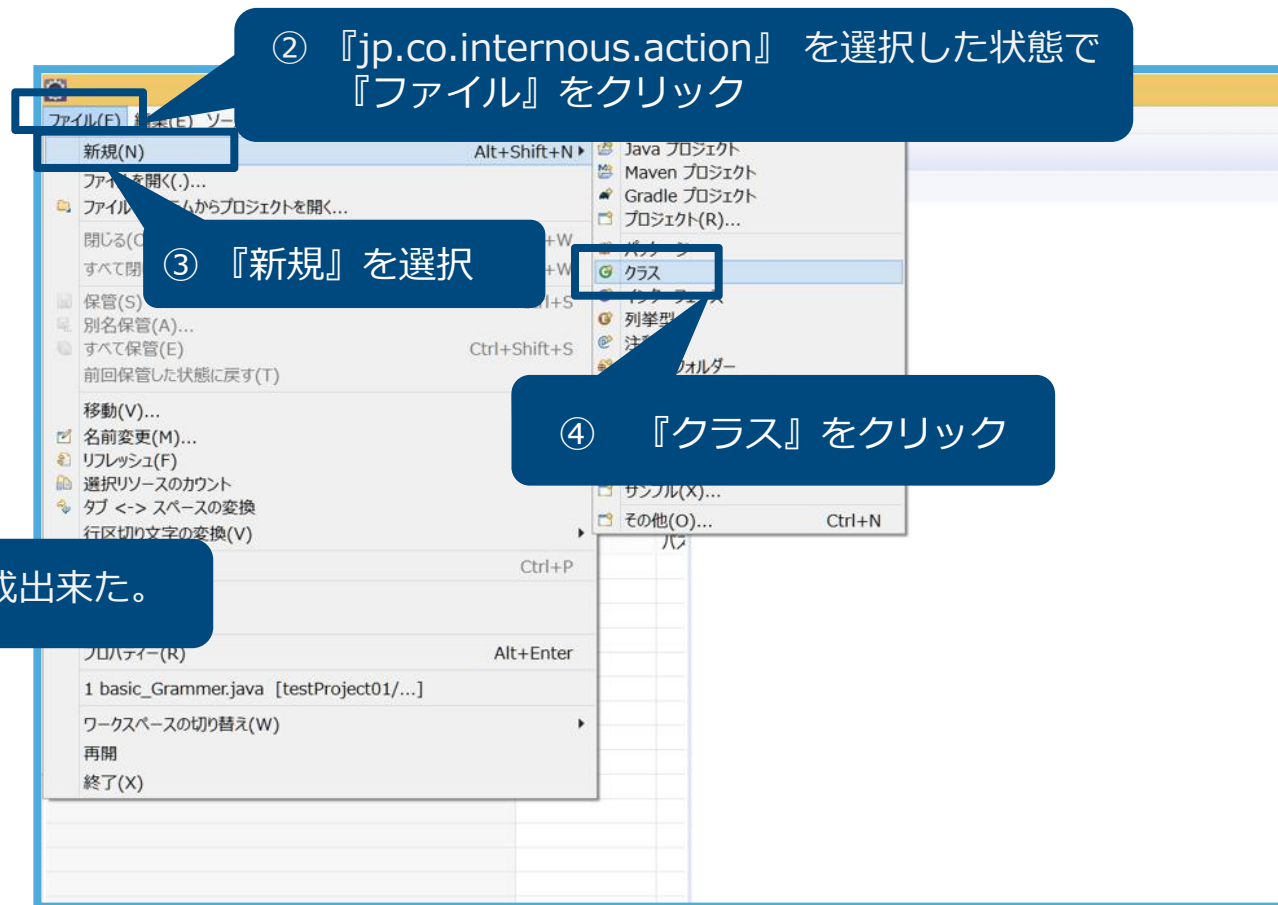
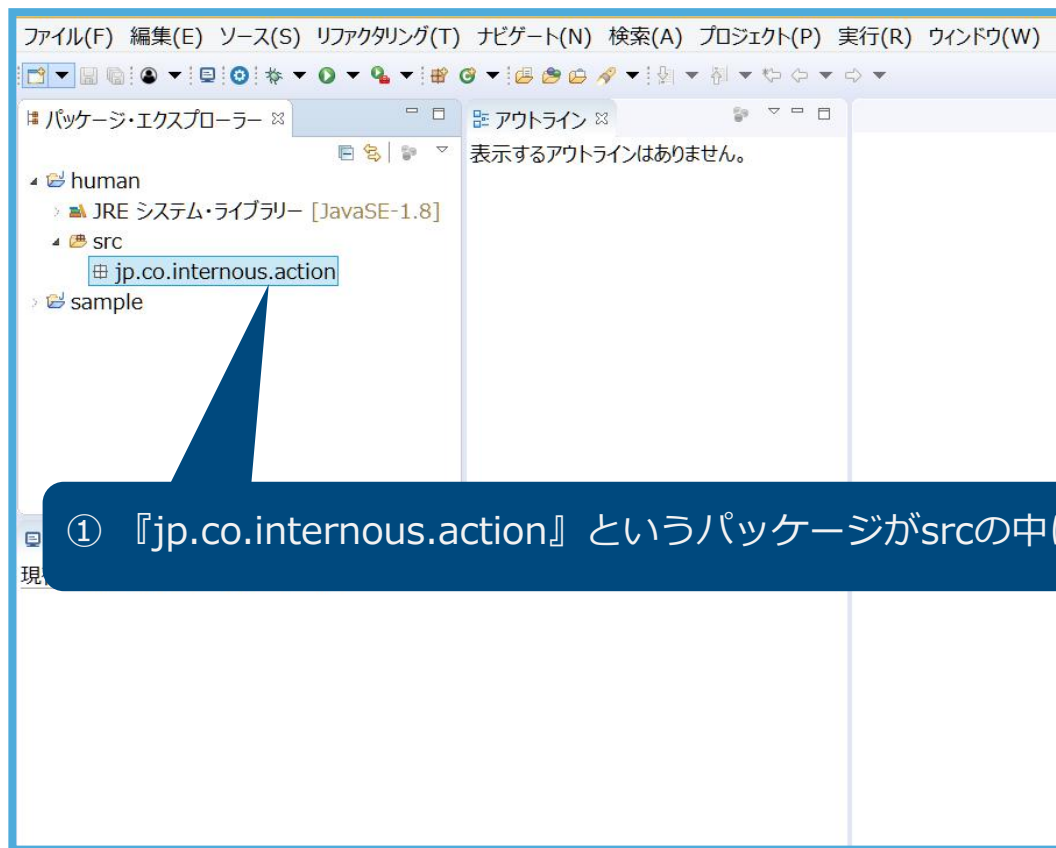
③ チェックを外す

② 『jp.co.internous.action』と書く。
パッケージ名は、IT業界の慣習として、公開したいURLの逆にするのが一般的。
※これやらないと現場で怒られるので注意。

internous.co.jpが、当社のURLなので、ここでは、このURLを逆にして、
『jp.co.internous.action』と入力する。

④ 『完了』をクリック

『Human』 クラスを作成



『Human』 クラスを作成

Java クラス

新規 Java クラスを作成します。

ソース・フォルダー(D): human/src 参照(O)...

パッケージ(K): jp.co.internous.action 参照(W)

☐ エンクロージング型(Y):

名前(M): Human

修飾子: ☒ public(P) ☐ パッケージ(C) ☐ private(V) ☐ protected(T)
☐ abstract(T) ☐ final(L) ☐ static(C)

スーパークラス(S): java.lang.Object

インターフェース(I):

除去(R)

どのメソッド・スタブを作成しますか?

☐ public static void main(String[] args)(V)
☐ スーパークラスからのコンストラクター(U)
☒ 継承された抽象メソッド(H)

コメントを追加しますか? (テンプレートの構成およびデフォルト値については[ここ](#)を参照)
☐ コメントの生成(G)

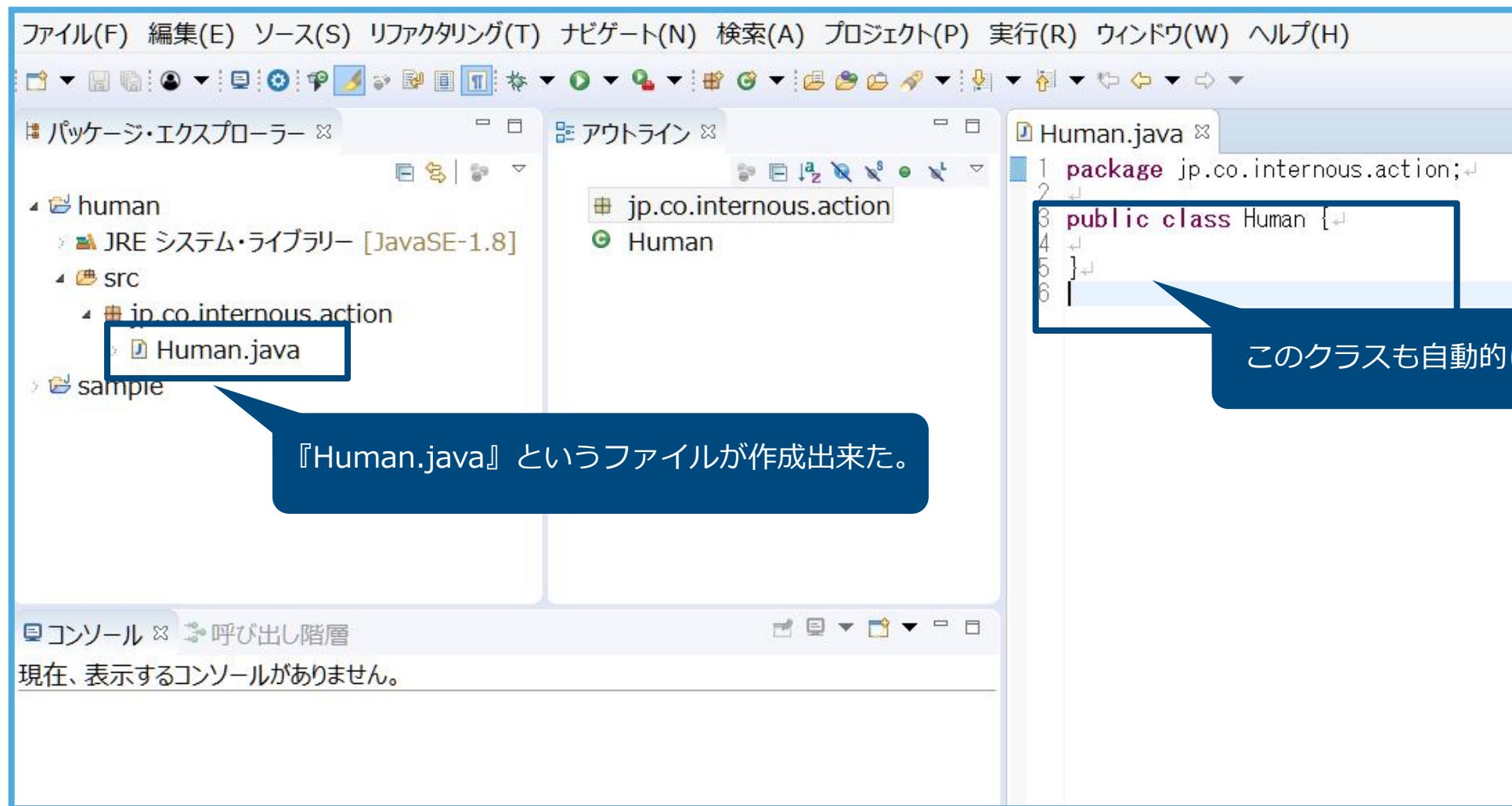
? 完了(F) キャンセル

① 『Human』 と入力。

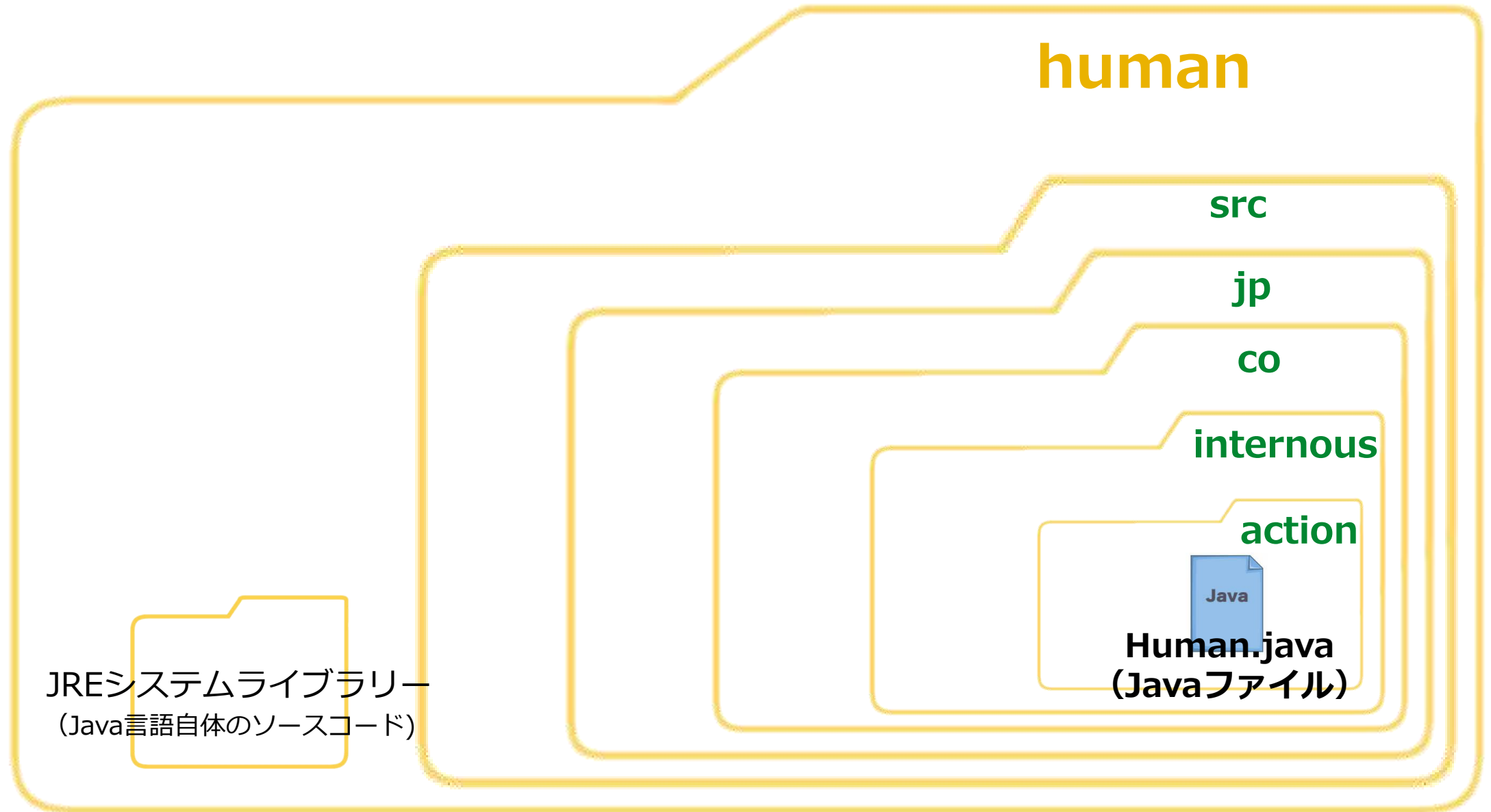
※クラス名の頭文字は必ず大文字にすること。
頭文字を小文字にするとアラートがあがり、さらに現場でも怒られるので注意。

② 『完了』 をクリック

『Human』 クラスを作成



現時点での作成したフォルダ構成（イメージ図）



Humanクラスの中に、mainメソッドを作成

今回は、このHumanクラスの中に、メソッドを記述する事はせず、『mainメソッド』を記述し、その中身は空にしておく。

```
*Human.java
1 package jp.co.internous.action;
2
3 public class Human {
4     public static void main(String[] args) {
5     }
6 }
7
8
9
10
```

mainメソッドの中身は空にしておく。

ここは、常に「String」にする。
intなどにはしない。

『HumanName』 クラスを作成

The image shows a screenshot of an IDE with two windows. The left window is the 'File' menu, and the right window is the 'New Java Class' dialog. Blue callout boxes with numbers 1 through 5 provide step-by-step instructions.

① 『src』 を右クリック

② 『新規』 を選択

③ 『クラス』 をクリック

④ クラス名を『HumanName』 と入力する。

⑤ 『完了』 をクリック

The 'New Java Class' dialog shows the following fields and options:

- Java クラス: 新規 Java クラスを作成します。
- パッケージ(D): human/src
- パッケージ(K): jp.co.internous.action
- 名前(M): HumanName
- 修飾子: ☒ public(P), ☐ private(V), ☐ protected(T), ☐ abstract(T), ☐ final(F), ☐ static(C)
- スーパークラス(S): java.lang.Object
- インターフェース(I):
- どのメソッド・スタブを作成しますか?: ☐ public static void main(String[] args)(V), ☐ スーパークラスからのコンストラクター(U), ☒ 継承された抽象メソッド(H)
- コメントを追加しますか? (テンプレートの構成およびデフォルト値については [ここ](#) を参照) ☐ コメントの生成(G)
- 完了(F) キャンセル

『HumanName』 クラスを作成

The screenshot shows an IDE interface with the following components:

- メニューバー:** 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)
- ツールバー:** Various icons for file operations, editing, and running.
- パッケージ・エクスプローラー:** Displays the project structure. The path `human > src > jp.co.internous.action > HumanName.java` is highlighted with a blue box.
- アウトライン:** Shows the package `jp.co.internous.action` and the class `HumanName`.
- エディタ:** Displays the source code of `HumanName.java`. The code is:

```
1 package jp.co.internous.action;
2
3 public class HumanName {
4     // ...
5 }
```
- コンソール:** At the bottom, it shows the message: 現在、表示するコンソールがありません。

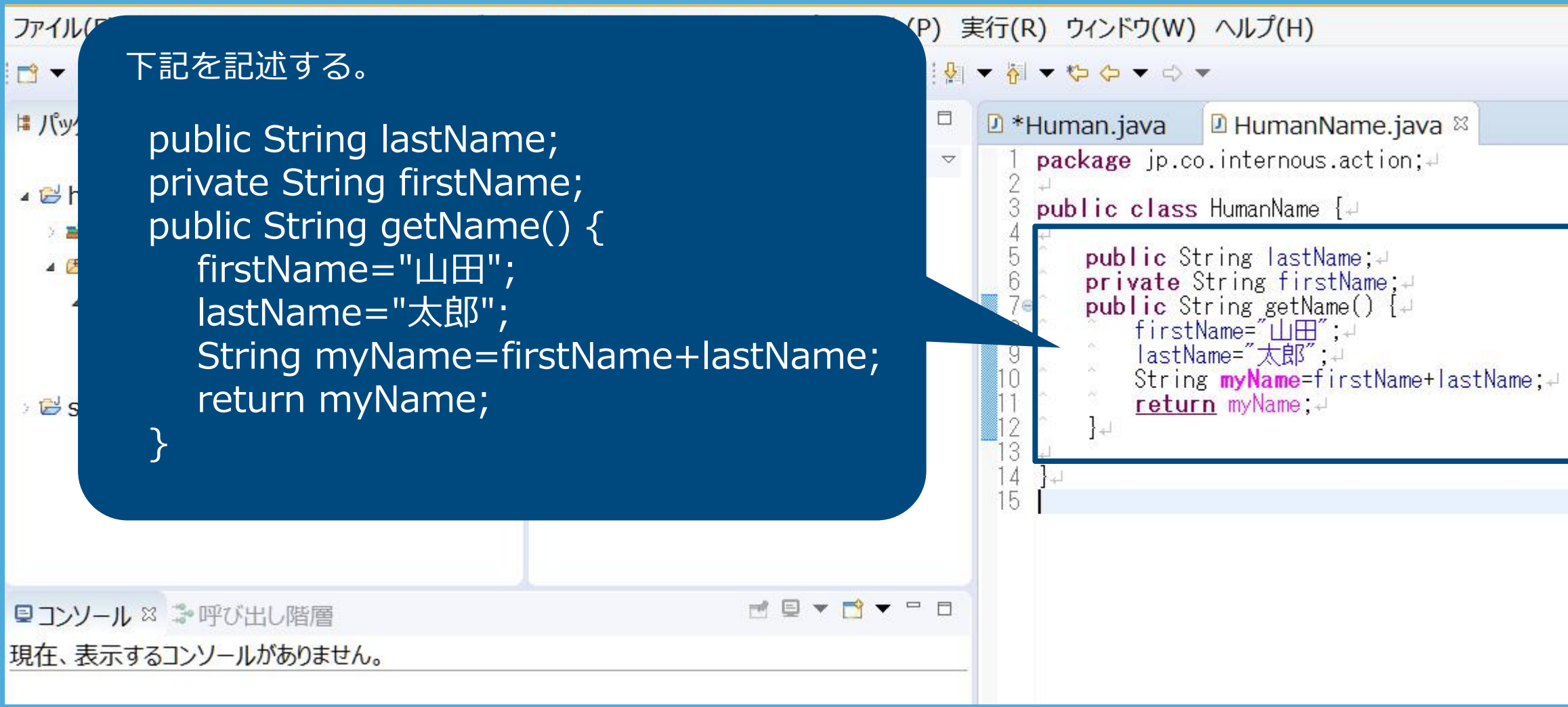
Two callout boxes provide additional information:

- A box pointing to the `HumanName.java` file in the Package Explorer says: 『HumanName.java』 というファイルが作成出来た。
- A box pointing to the `public class HumanName` line in the editor says: このクラスも自動的に作成された。

HumanNameクラスに、getNameメソッドを作成

下記を記述する。

```
public String lastName;  
private String firstName;  
public String getName() {  
    firstName="山田";  
    lastName="太郎";  
    String myName=firstName+lastName;  
    return myName;  
}
```

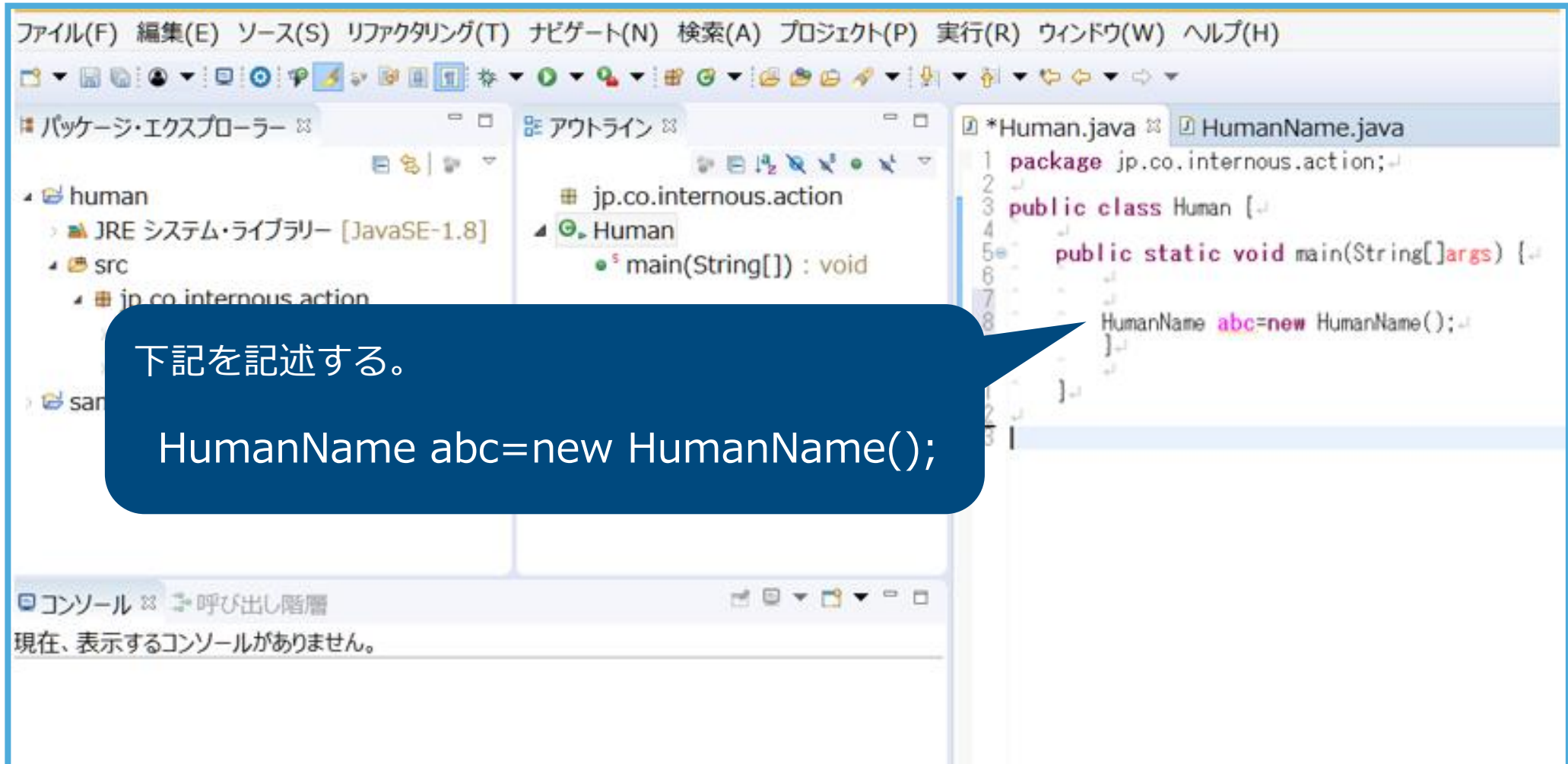


```
1 package jp.co.internous.action;  
2  
3 public class HumanName {  
4  
5     public String lastName;  
6     private String firstName;  
7     public String getName() {  
8         firstName="山田";  
9         lastName="太郎";  
10        String myName=firstName+lastName;  
11        return myName;  
12    }  
13  
14 }  
15
```

コンソール 呼び出し階層

現在、表示するコンソールがありません。

Humanクラスと、HumanNameクラスを連動させる



解説

```
HumanName.java *Human.java
1 package human;
2
3 public class Human {
4
5     public static void main(String[] args) {
6
7         変数名 (ここでは理解しやすくするために"abc"にしている)
8         HumanName abc=new HumanName();
9     }
10 }
11
12
```

HumanName abc=new HumanName();

『変数abcの中に、HumanNameクラスの
コピーが代入される。』という意味。

```
*HumanName.java *Human.java
1 package human;
2
3 public class HumanName {
4
5     public String lastName;
6     private String firstName;
7     public String getName() {
8         firstName="山田";
9         lastName="太郎";
10        String myName=firstName+lastName;
11        return myName;
12    }
13 }
14
```

8行目の『**HumanName**』とは、
『**HumanName.java**』と『public class **HumanName**』のHumanName
を指している。

『HumanName.java』と『public class HumanName』は 同じ名前に
しないとダメ (クラス (=ファイル作成) の時に自動的に同じ名前で作られる)

getNameメソッドを呼び出す

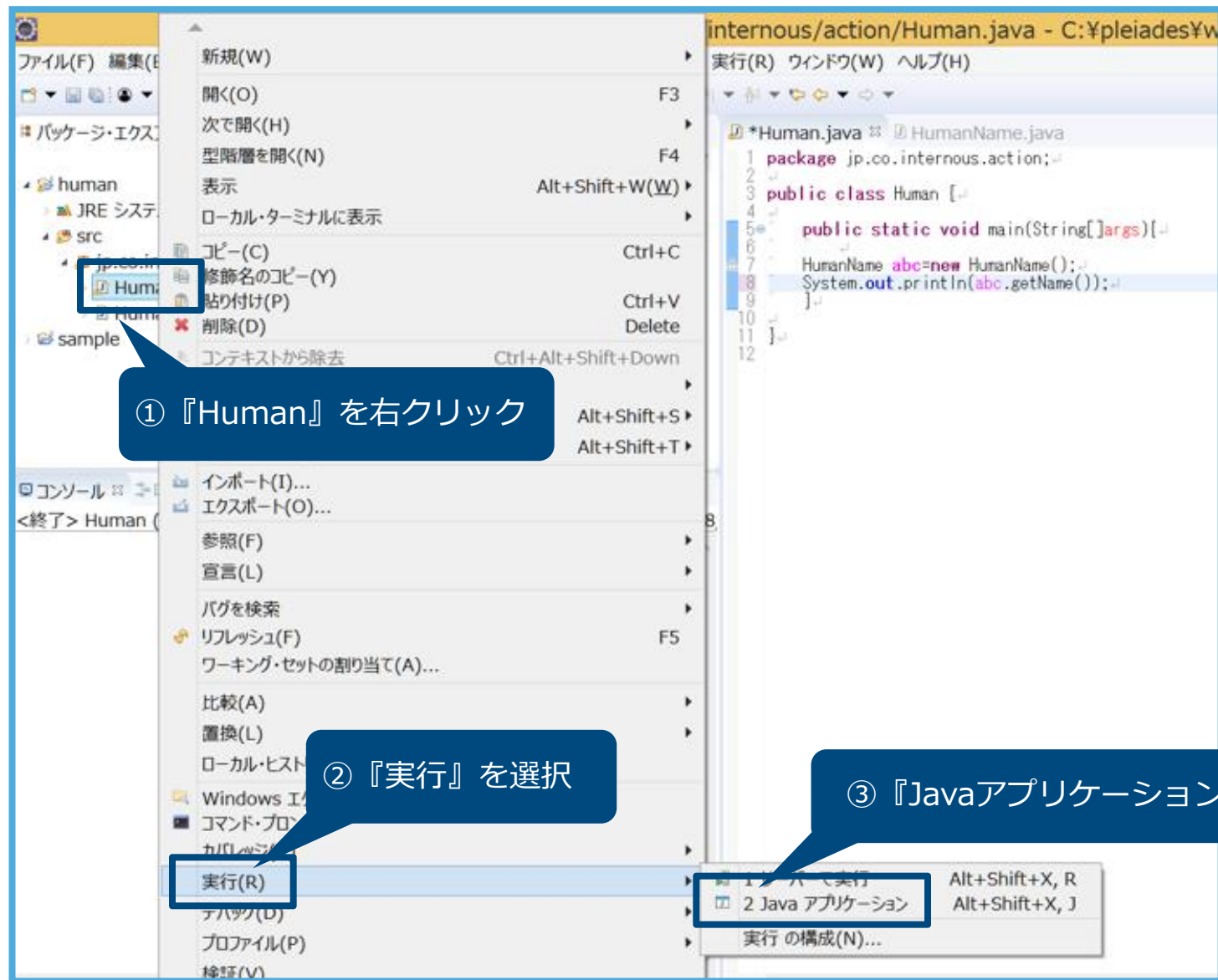
```
*Human.java  HumanName.java
1 package jp.co.internous.action;
2
3 public class Human {
4
5     public static void main(String[] args){
6
7         HumanName abc=new HumanName();
8         System.out.println(abc.getName());
9     }
10
11 }
```

下記を記述する。

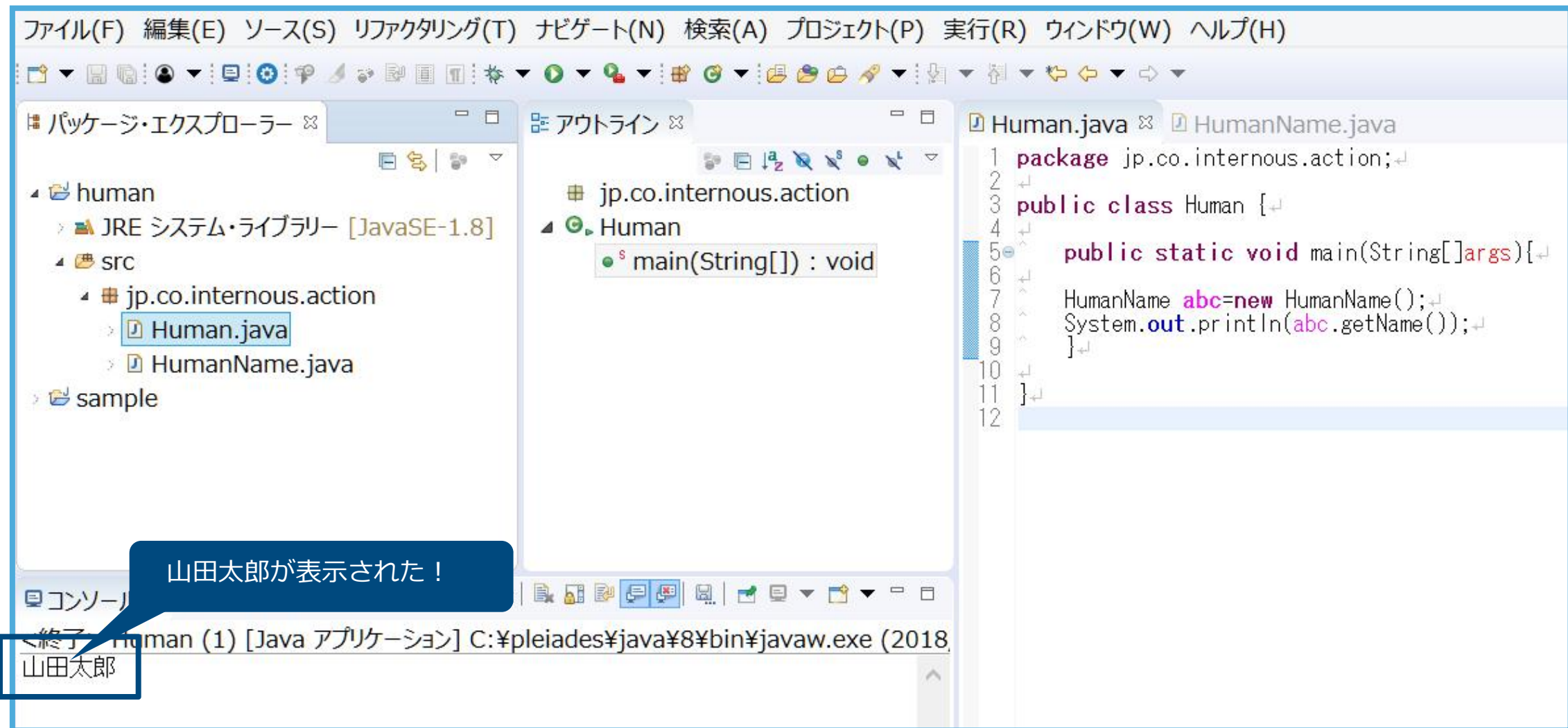
```
System.out.println(abc.getName());
```

```
*HumanName.java  *Human.java
1 package human;
2
3 public class HumanName {
4
5     public String lastName;
6     private String firstName;
7     public String getName() {
8         firstName="山田";
9         lastName="太郎";
10        String myName=firstName+lastName;
11        return myName;
12    }
13
14 }
```

実行

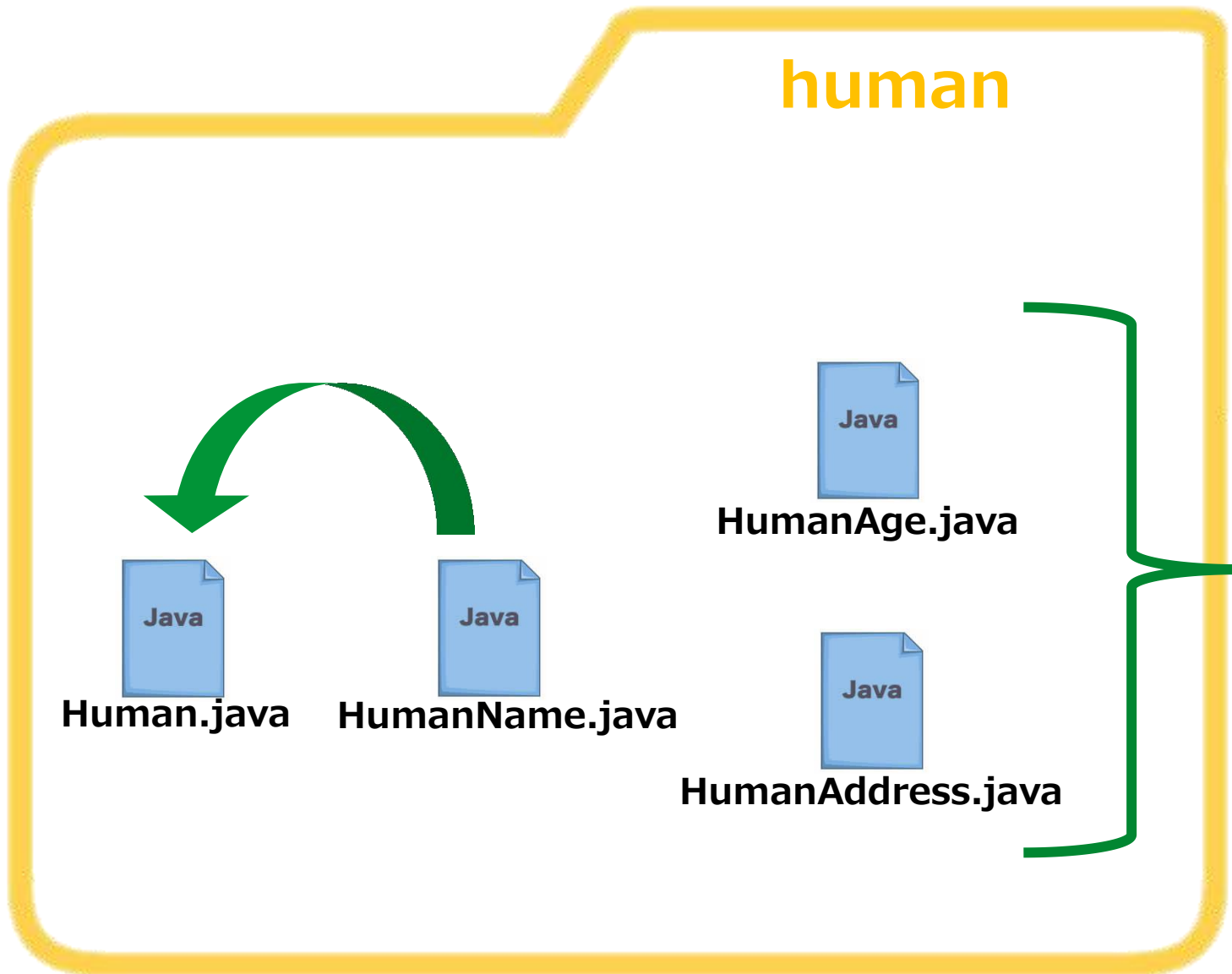


山田太郎が表示



いくつか事例を見てみましょう。

新規で2つのクラスを追加

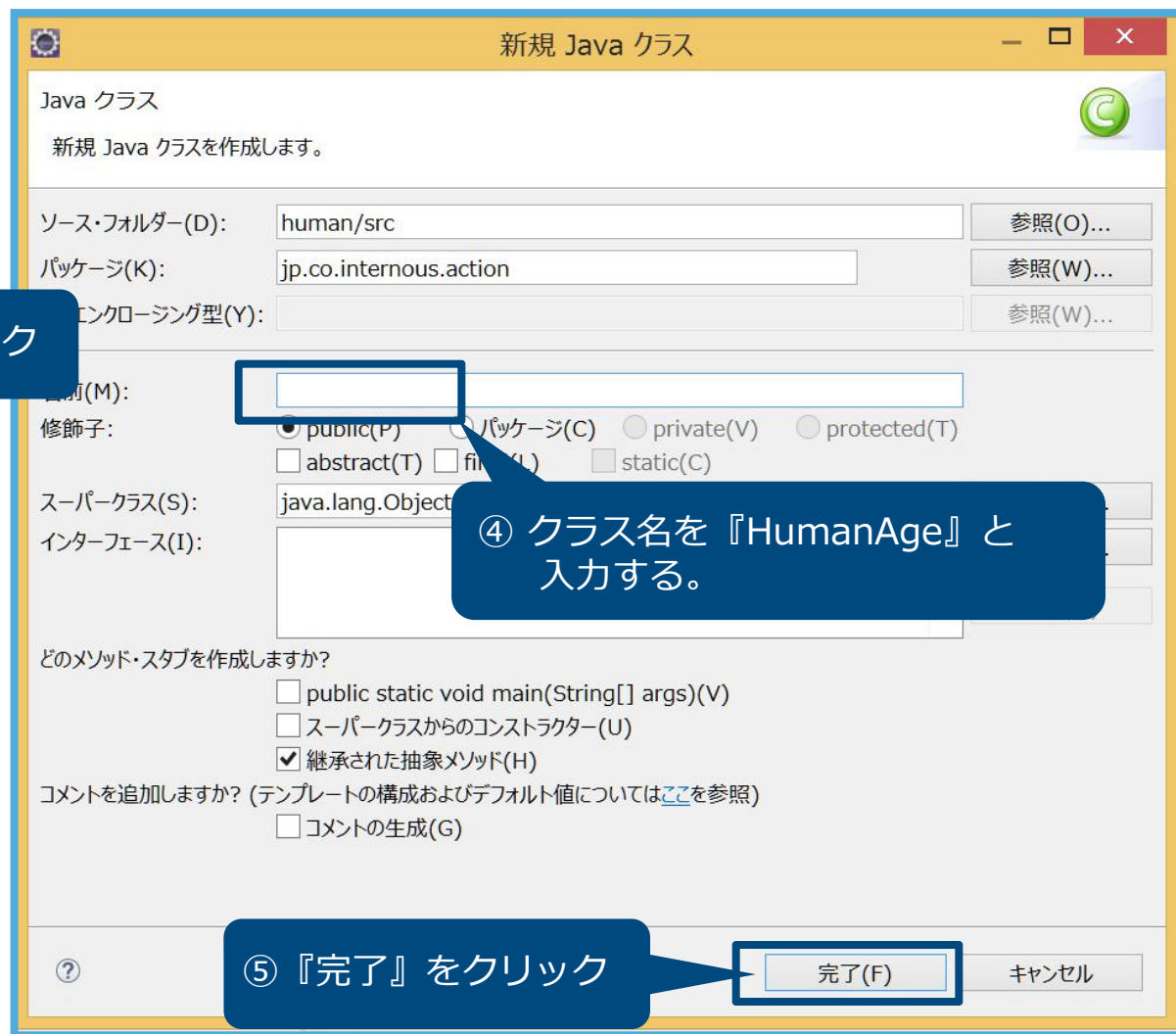
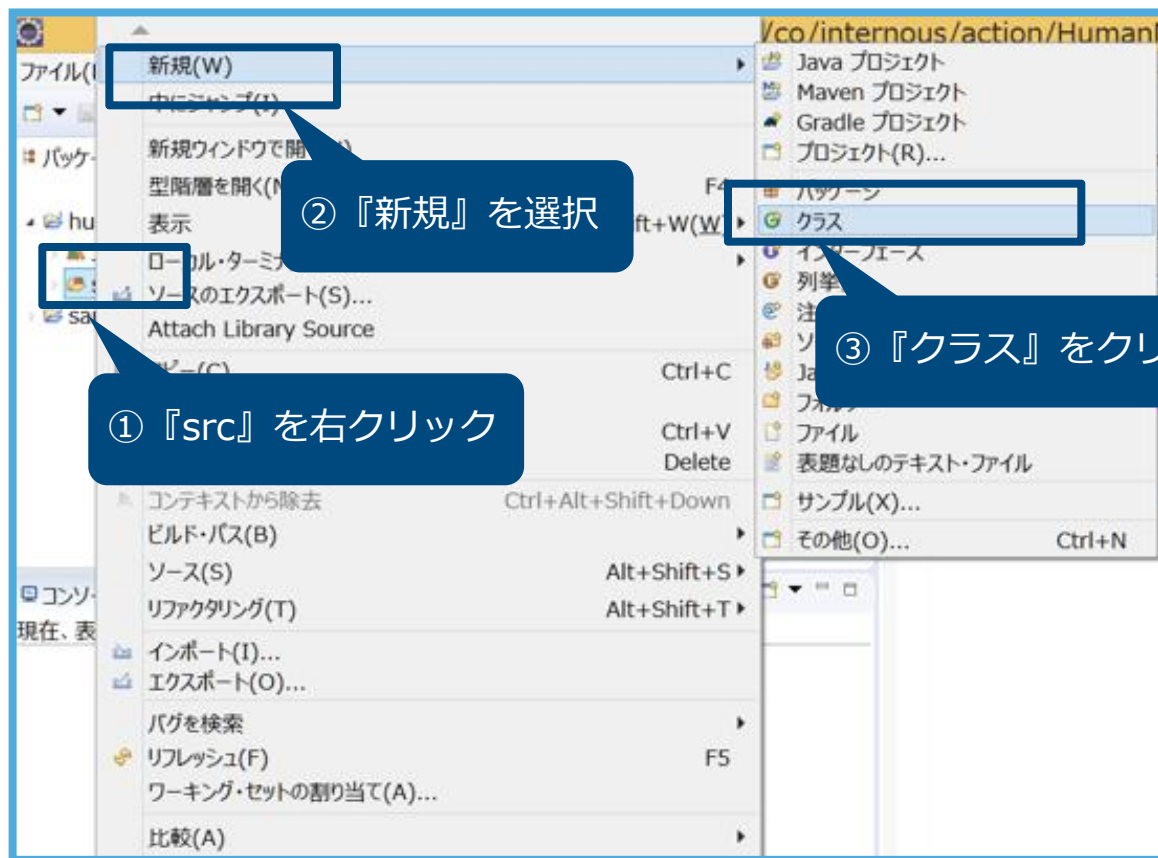


新規でこの2つのクラスを作成して、**Human.java**で、**山田太郎の年齢と住所を読み込んでみましょう。**

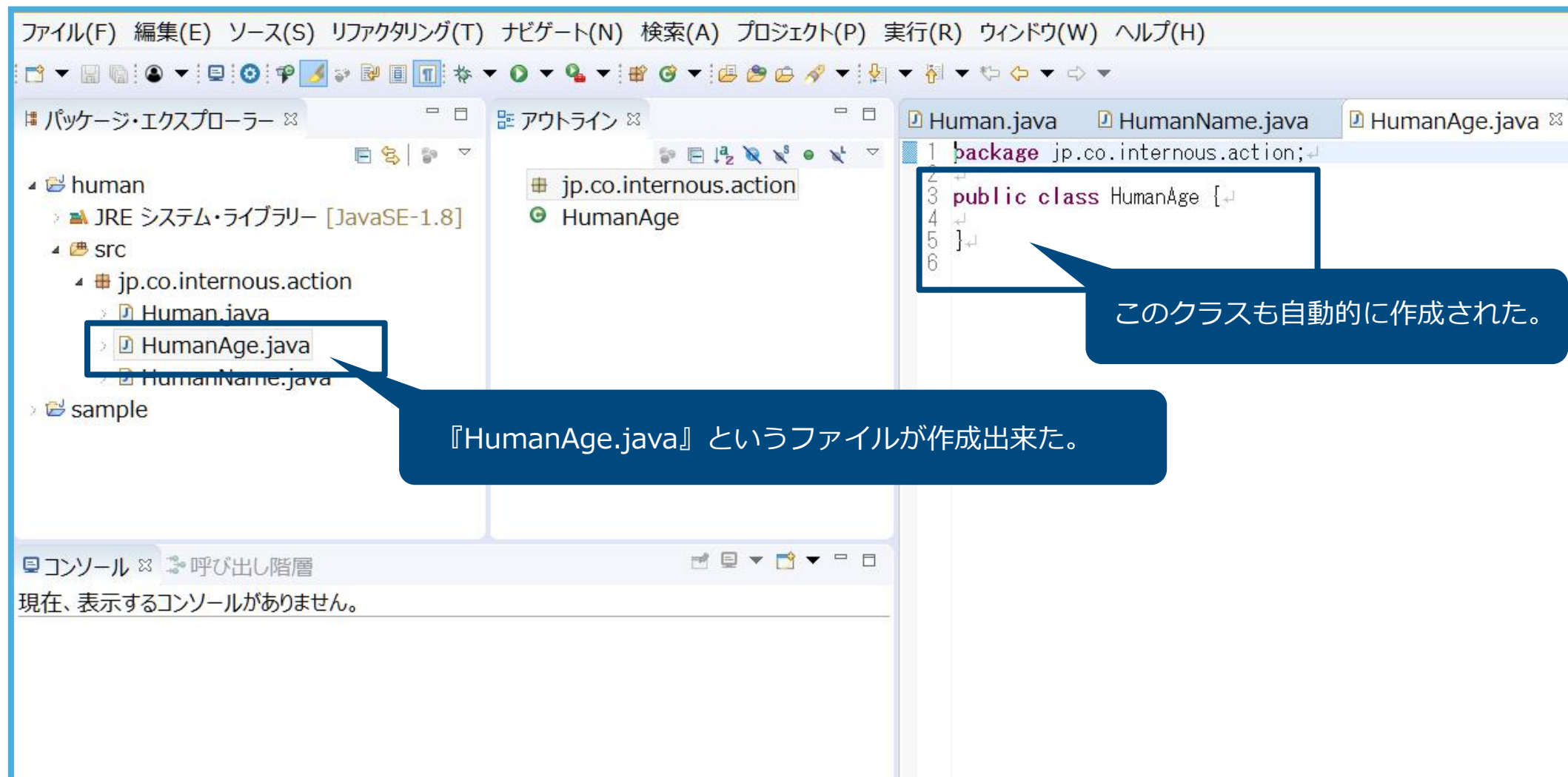
- ・ 年齢は25歳
- ・ 住所は東京

に設定する。

『HumanAge』 クラスを作成



『HumanAge』 クラスを作成



HumanAgeクラスに、getAgeメソッドを作成

The screenshot shows an IDE with the following components:

- メニューバー:** 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)
- ツールバー:** Various icons for file operations, editing, and running.
- パッケージ・エクスプローラー:** Shows the project structure with the following hierarchy:
 - human
 - JRE システム・ライブラリー [JavaSE-1.8]
 - src
 - jp.co.internous.action
 - Human.java
 - HumanAge.java (selected)
 - HumanName.java
 - sample
- アウトライン:** Shows the class structure for `jp.co.internous.action.HumanAge` with a member `age : int`.
- コードエディタ:** Displays the code for `HumanAge.java`. The code is as follows:

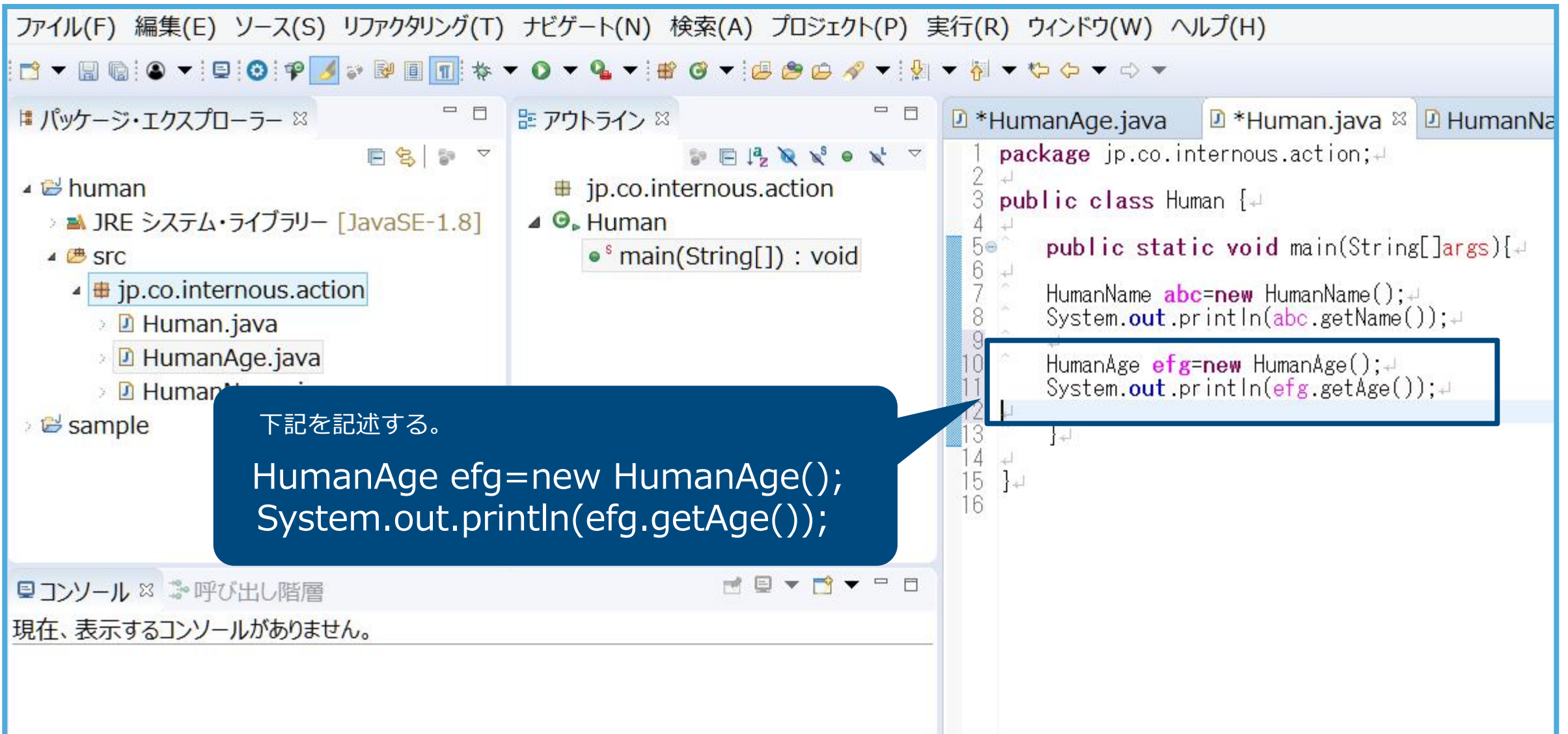
```
1 package jp.co.internous.action;
2
3 public class HumanAge {
4
5     public int age;
6     public int getAge() {
7         age = 25;
8         return age;
9     }
10
11 }
12
13
```
- コンソール:** Shows the message: 現在、表示するコンソールがありません。

A blue callout box points to the `getAge` method in the code editor, containing the following text:

下記を記述する。

```
public int age;
public int getAge() {
    age = 25;
    return age;
}
```

HumanAgeクラスと連動させgetAgeメソッドの呼び出す

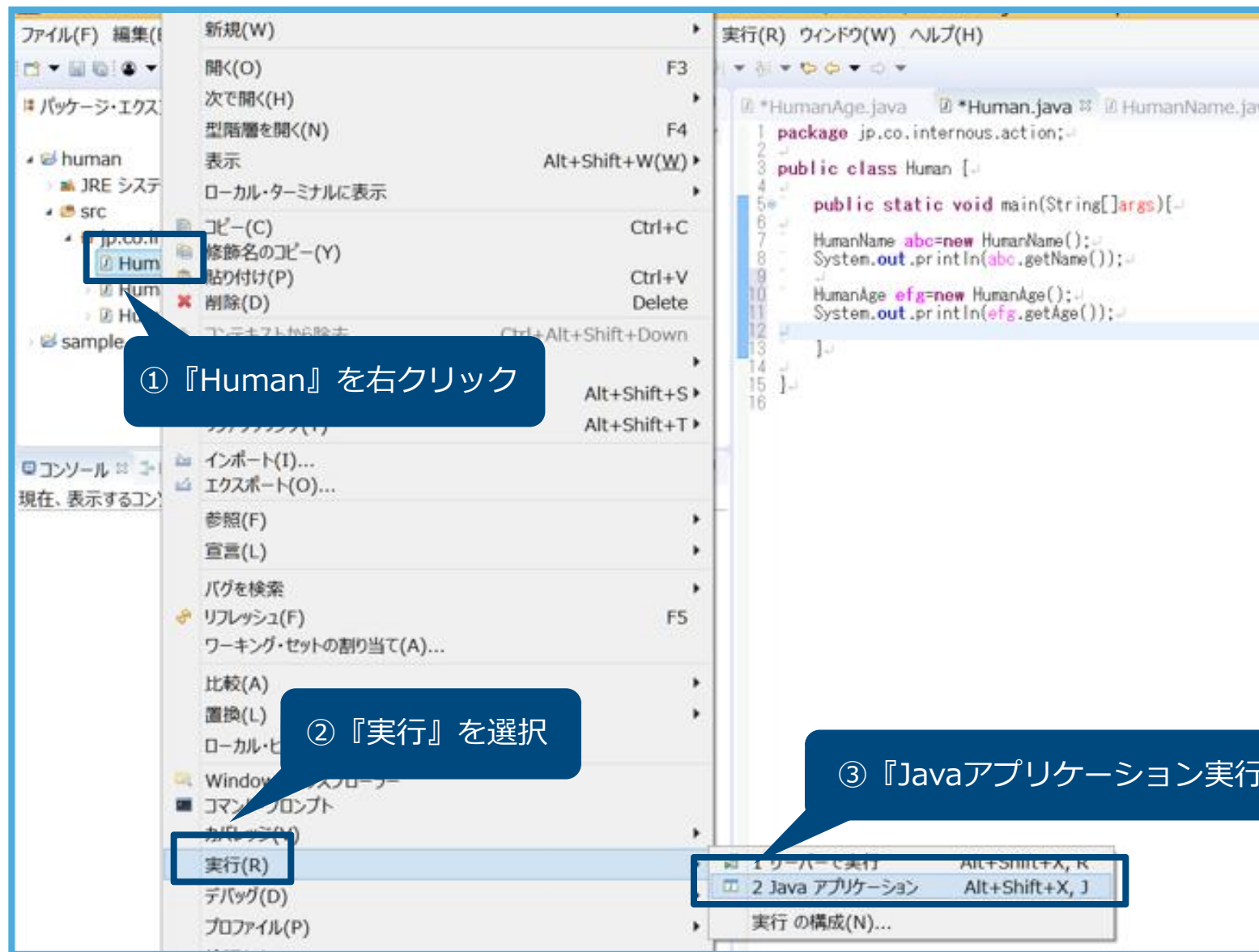


The screenshot shows an IDE with the following components:

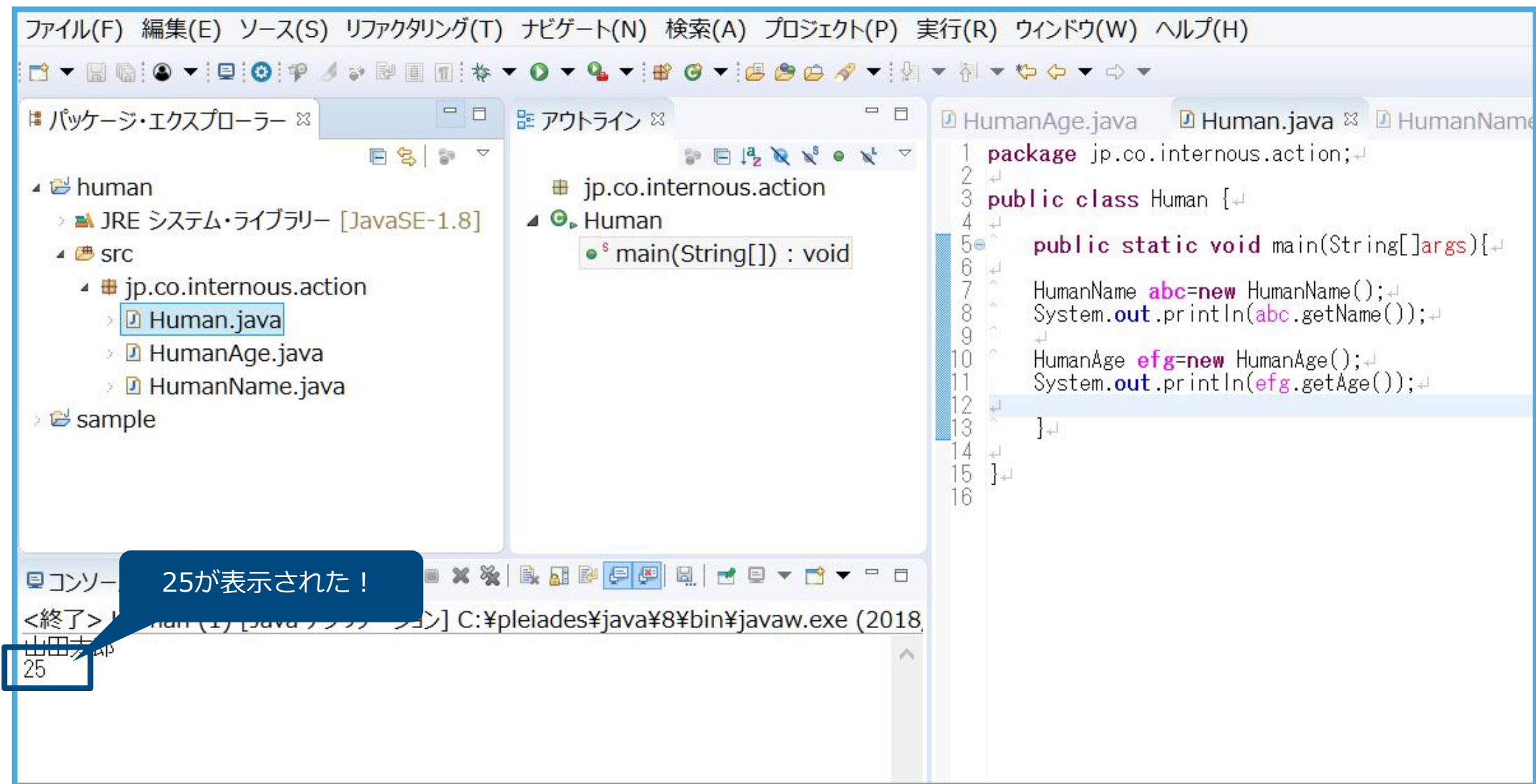
- Package Explorer:** Shows the project structure with 'human' folder containing 'JRE システム・ライブラリー [JavaSE-1.8]', 'src' folder, and 'sample' folder. Under 'src', there is a package 'jp.co.internous.action' containing 'Human.java', 'HumanAge.java', and 'HumanName.java'.
- Outline:** Shows the 'jp.co.internous.action' package with a 'Human' class and a 'main(String[]) : void' method.
- Editor:** Displays the 'HumanAge.java' file with the following code:

```
1 package jp.co.internous.action;
2
3 public class Human {
4
5     public static void main(String[] args){
6
7         HumanName abc=new HumanName();
8         System.out.println(abc.getName());
9
10        HumanAge efg=new HumanAge();
11        System.out.println(efg.getAge());
12    }
13
14 }
15
16
```
- Callout Box:** A blue box with a pointer to line 11 of the code, containing the text: '下記を記述する。' (Describe the following.) followed by the code: 'HumanAge efg=new HumanAge();' and 'System.out.println(efg.getAge());'.
- Console:** Shows the message '現在、表示するコンソールがありません。' (No console to display is currently available.)

実行



25が表示



『HumanAddress』 クラスを作成

The screenshot illustrates the process of creating a new Java class in an IDE. The main window shows a project structure with a file explorer on the left and a code editor on the right. A context menu is open over the 'src' directory, and a 'New Java Class' dialog is displayed.

① 『src』 を右クリック

② 『新規』 を選択

③ 『クラス』 をクリック

④ クラス名を『HumanAddress』と入力する。

⑤ 『完了』 をクリック

The 'New Java Class' dialog shows the following fields and options:

- Java クラス: 新規 Java クラスを作成します。
- パッケージ(D): human/src
- パッケージ(K): jp.co.internous.action
- エンクロージング型(Y):
- 名前(M):
- 修飾子: ☒ public(P) ☐ パッケージ(C) ☐ private(V) ☐ protected(T) ☐ abstract(T) ☐ final(F) ☐ static(C)
- スーパークラス(S): java.lang.Object
- インターフェース(I):
- どのメソッド・スタブを作成しますか?: ☐ public static void main(String[] args)(V) ☐ スーパークラスからのコンストラクター(U) ☒ 継承された抽象メソッド(H)
- コメントを追加しますか? (テンプレートの構成およびデフォルト値については[ここ](#)を参照) ☐ コメントの生成(G)
- 完了(F) キャンセル

『HumanAddress』 クラスを作成

The screenshot shows an IDE with the following components:

- メニューバー:** 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)
- ツールバー:** Various icons for file operations, editing, and running.
- パッケージ・エクスプローラー:** Displays the project structure. The package `jp.co.internous.action` is expanded, showing files `Human.java`, `HumanAddress.java`, `HumanAge.java`, and `HumanName.java`. `HumanAddress.java` is highlighted with a blue box.
- アウトライン:** Shows the package `jp.co.internous.action` and the class `HumanAddress`.
- エディタ:** Displays the source code of `HumanAddress.java`. The code is:

```
1 package jp.co.internous.action;
2
3 public class HumanAddress {
4     //
5 }
6
```
- コンソール:** Shows the output of the Java application. The text is: `<終了> Human (1) [Java アプリケーション] C:\pleiades\java\8\bin\javaw.exe (2018, 山田太郎 25`

Two callouts provide additional information:

- A callout pointing to `HumanAddress.java` in the Package Explorer says: 『HumanAge.java』 というファイルが作成出来た。
- A callout pointing to the `HumanAddress` class in the Outline says: このクラスも自動的に作成された。

getAdressメソッドを作成

The screenshot shows an IDE with the following components:

- メニューバー:** 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)
- ツールバー:** Various icons for file operations, editing, and running.
- パッケージ・エクスプローラー:** Shows the project structure with 'human' > 'src' > 'jp.co.internous.action' > 'HumanAddress.java' selected.
- アウトライン:** Shows the class 'HumanAddress' with attributes 'address : String' and method 'getAddress() : String'.
- エディタ:** Displays the code for 'HumanAddress.java'. The following code is highlighted with a blue box:

```
1 package jp.co.internous.action;
2
3 public class HumanAddress {
4
5     public String address;
6     public String getAddress() {
7         address="東京";
8         return address;
9     }
10
11
12 }
13
```
- コンソール:** Shows the output of a program run:

```
<終了> Human (1)
山田太郎
25
```

A blue callout box points to the highlighted code in the editor, containing the following text:

下記を記述する。

```
public String address;
public String getAddress() {
    address="東京";
    return address;
}
```

HumanAgeクラスと連動させ、getAddressメソッドを呼び出す

The screenshot shows an IDE with the following components:

- Package Explorer:** Shows the project structure with packages `human`, `JRE システム・ライブラリー [JavaSE-1.8]`, `src`, and `sample`. Under `src`, the package `jp.co.internous.action` contains `Human.java`, `HumanAddress.java`, `HumanAge.java`, and `HumanName.java`.
- Outline:** Shows the `Human` class with a `main(String[]) : void` method.
- Editor:** Displays the `Human.java` file. The code is as follows:

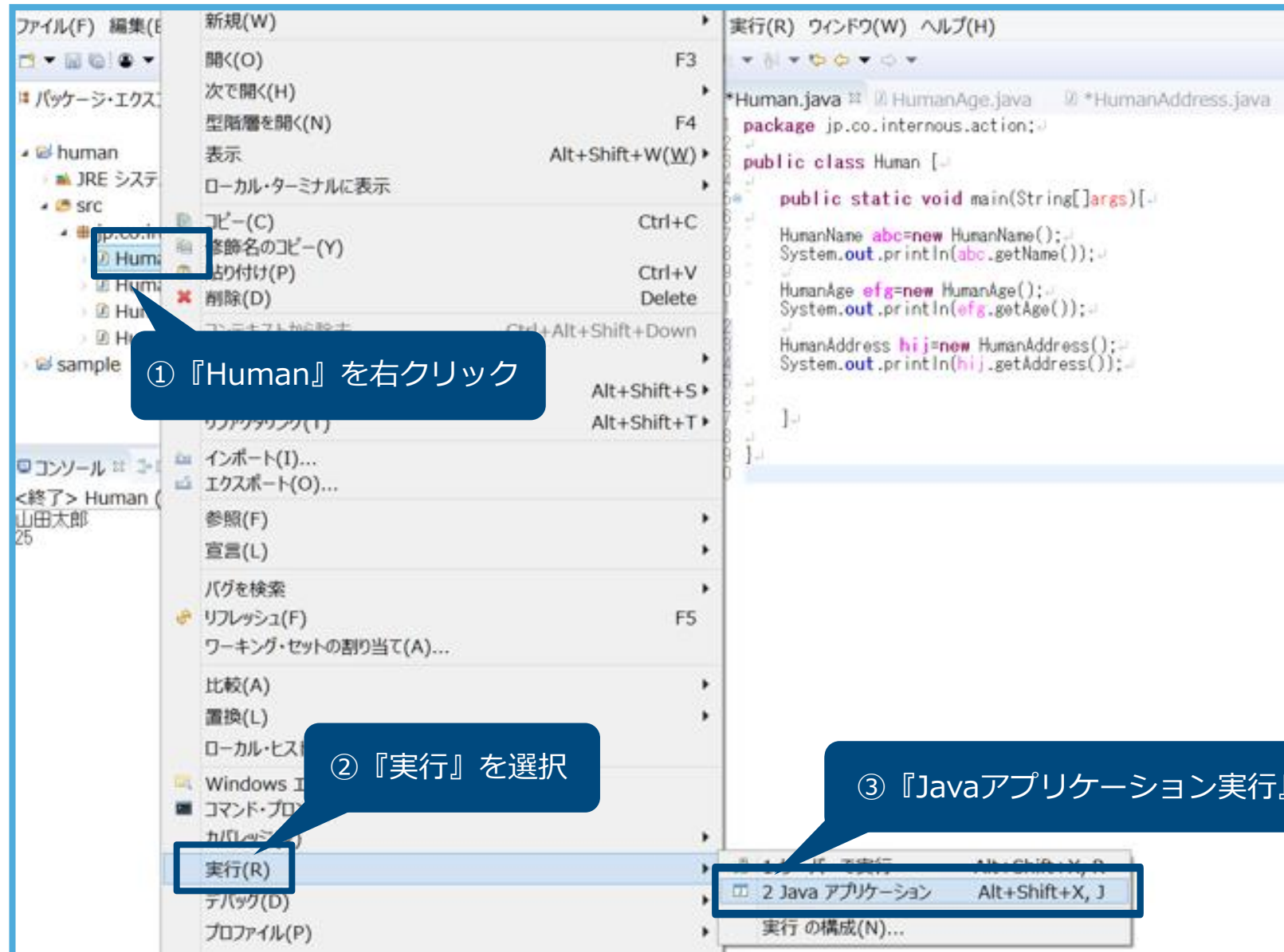
```
1 package jp.co.internous.action;
2
3 public class Human {
4
5     public static void main(String[] args) {
6
7         HumanName abc = new HumanName();
8         System.out.println(abc.getName());
9
10        HumanAge efg = new HumanAge();
11        System.out.println(efg.getAge());
12
13        HumanAddress hij = new HumanAddress();
14        System.out.println(hij.getAddress());
15    }
16 }
```

A callout box points to the `HumanAddress hij = new HumanAddress();` and `System.out.println(hij.getAddress());` lines in the code, indicating the focus of the task.

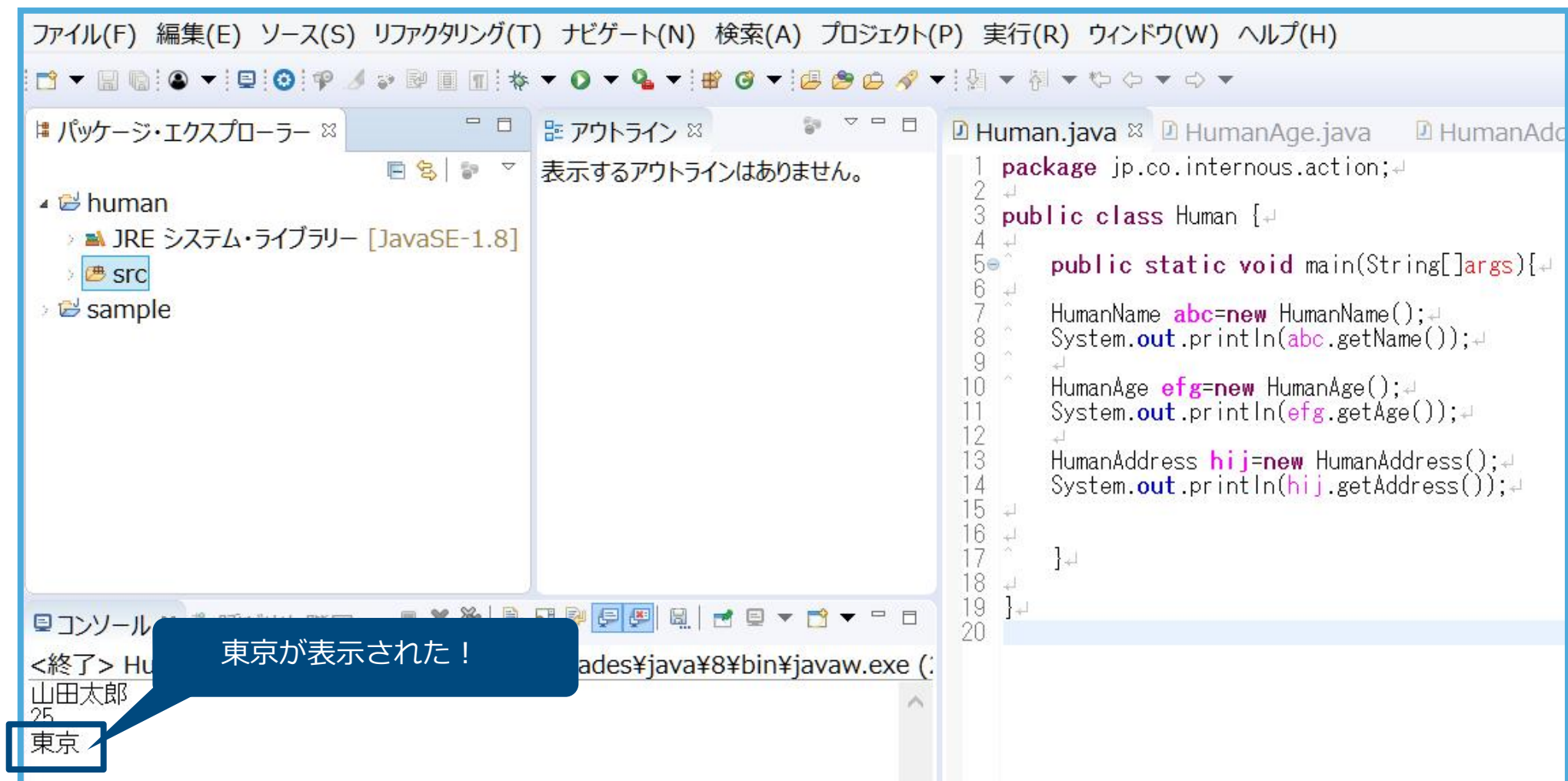
下記を記述する。

```
HumanAddress hij=new HumanAddress();
System.out.println(hij.getAddress());
```

実行



東京が表示



作成済みのクラスの比較

作成した下記3つのクラスとメソッドを比較してみよう

HumanName.java

```
public class HumanName {  
  
    public String lastName;  
    private String firstName;  
  
    public String getName() {  
        firstName="山田";  
        lastName="太郎";  
        String myName=firstName+lastName;  
        return myName;  
    }  
}
```

HumanAge.java

```
public class HumanAge {  
  
    public int age;  
  
    public int getAge() {  
        age = 25;  
        return age;  
    }  
}
```

HumanAddress.java

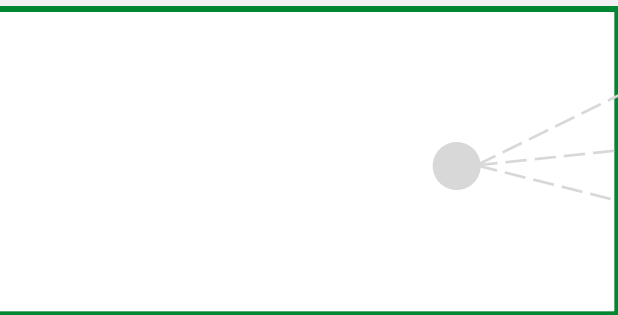
```
public class HumanAddress {  
  
    public String address;  
  
    public String getAddress() {  
        address="東京";  
        return address;  
    }  
}
```


•HumanGenderクラスの正しいメソッドを選んでください

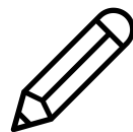
•たくさんあったので
•復習しましょう

Gender.java

```
public class HumanGender {
```



```
}
```



```
public int gender;
```

```
public int getGender() {  
    gender = "男性";  
    return gender;  
}
```

```
String gender;
```

```
String getGender() {  
    gender = "男性";  
    return gender;  
}
```

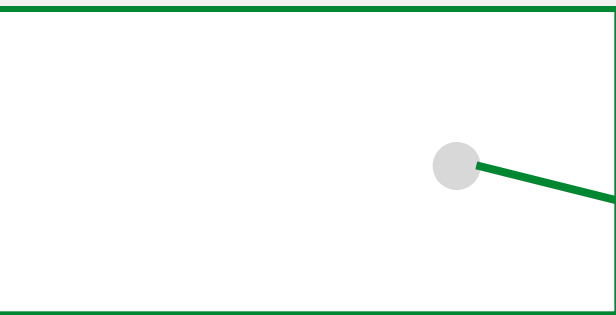
```
public String gender;
```

```
public String getGender() {  
    gender = "男性";  
    return gender;  
}
```

•たくさんあったので
•復習しましょう

Gender.java

```
public class HumanGender {
```



```
}
```

```
public int gender;
```

```
public int getGender() {  
    gender = "男性";  
    return gender;  
}
```

```
String gender;
```

```
String getGender() {  
    gender = "男性";  
    return gender;  
}
```

```
public String gender;
```

```
public String getGender() {  
    gender = "男性";  
    return gender;  
}
```