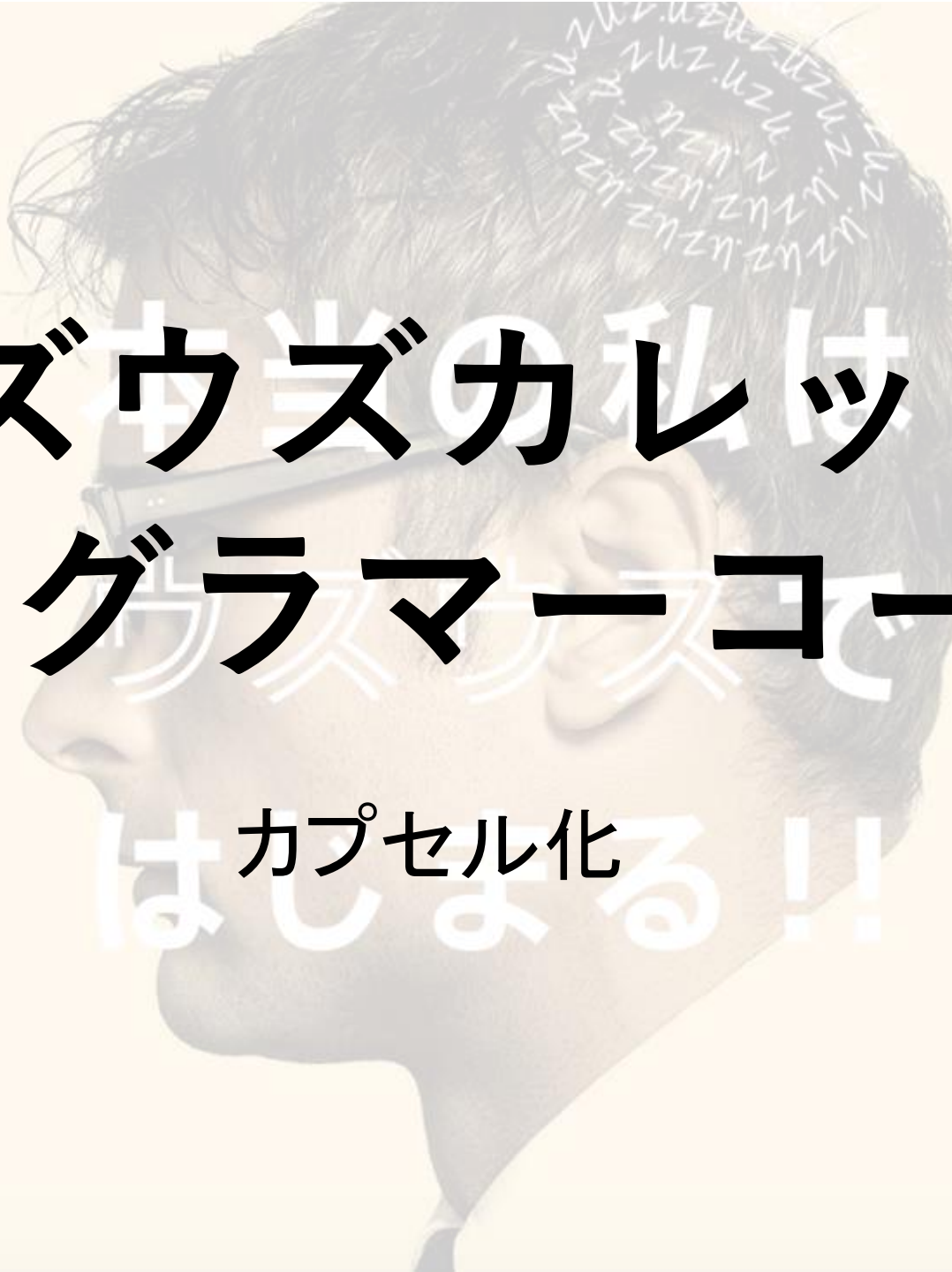
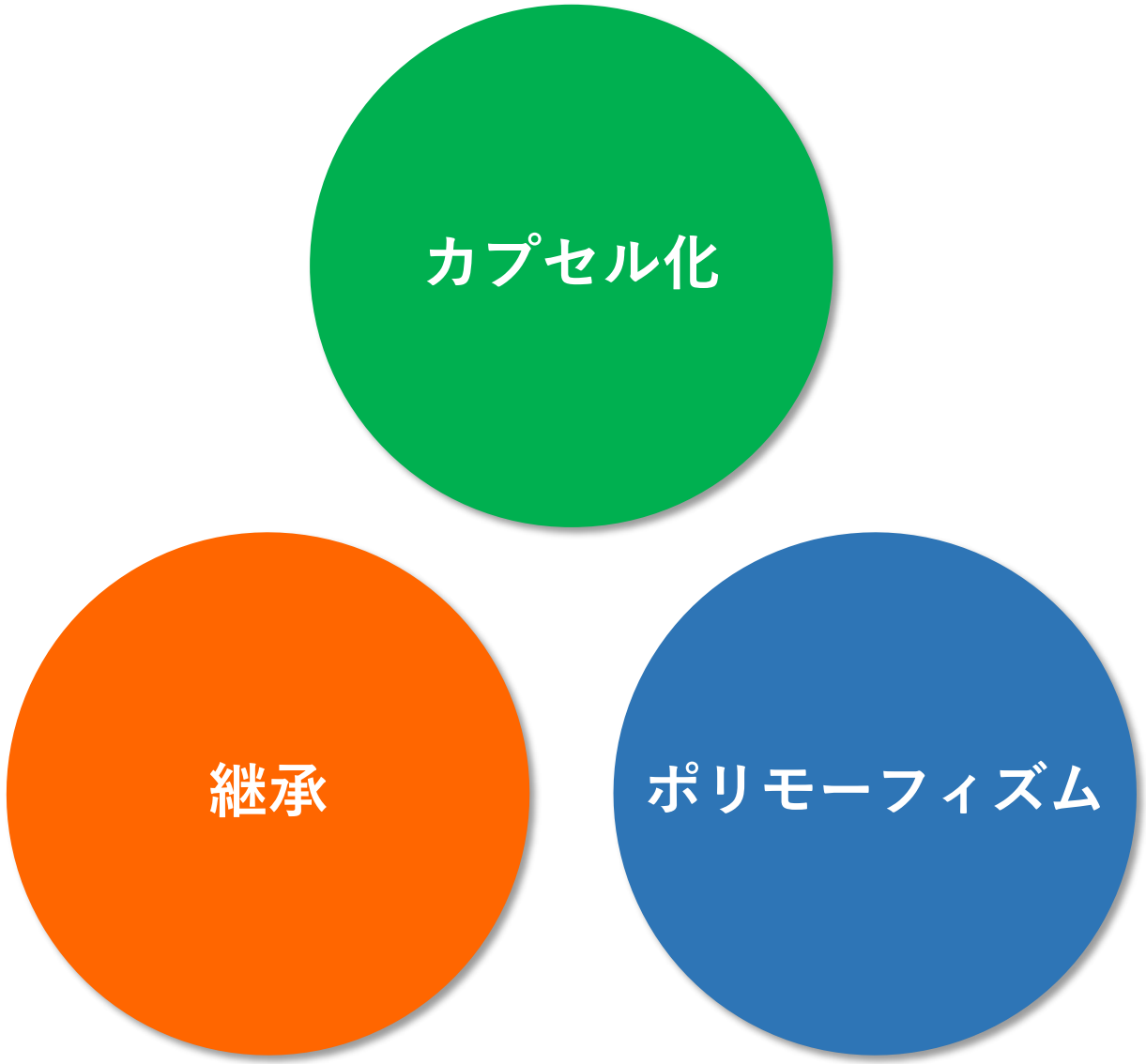
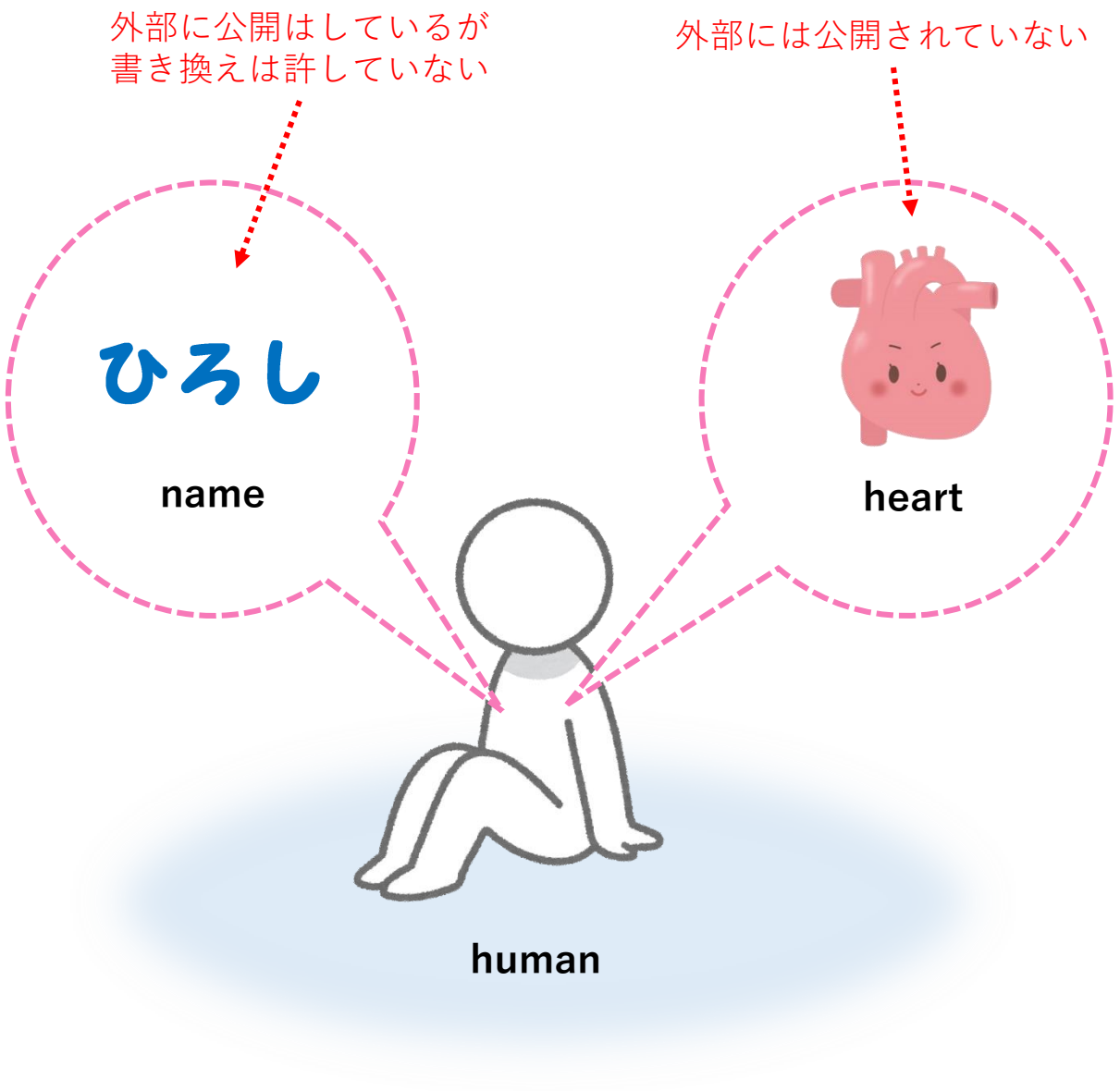


ウズウズカレッジ
プログラマーコース

カプセル化

～オブジェクト指向の3大要素～





≪カプセル化≫

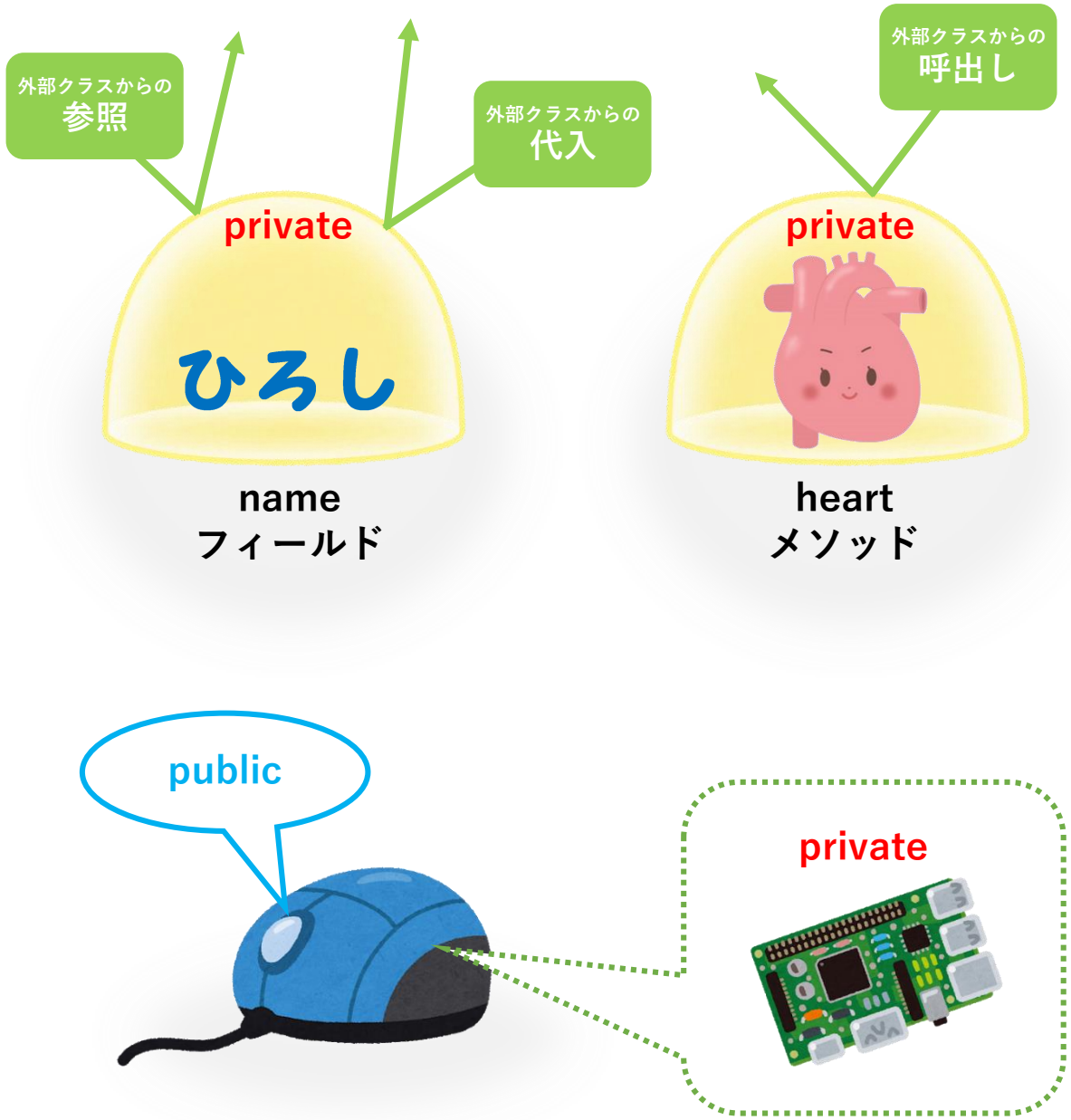
□現実世界のモノで想像してほしいのですが、モノというのは本来外部からアクセスできる情報や機能はそこまで多くありません。
例えば人間だと急所である心臓という機能を肉体で隠していますし、名前も誰かに変えられるようなことはないよう法律で守られています。

オブジェクト指向で表現するモノも同じく、フィールドやメソッドを無制限に外部に公開していると、ミスや悪意ある攻撃などによるトラブルが起きやすくなってしまいます。

□**privateやpublicといったアクセス修飾子を用い、外部クラスからのフィールドやメソッドへのアクセスを制御する**ことを**カプセル化**と言います。

カプセル化のポイントは以下の通りです。

- ミス・悪意によるフィールドの書き換えや意図せぬ用途でメソッドが起動されるなどの外部アクセスによるトラブルを防ぐため、**外部に公開する必要のないフィールドやメソッドはすべて隠す**。
- 外部にメンバ（特にフィールド）へのアクセスを許す場合も、そのメンバにアクセスするための**正式な手続き方法（アクセス用メソッド）のみを公開する**。



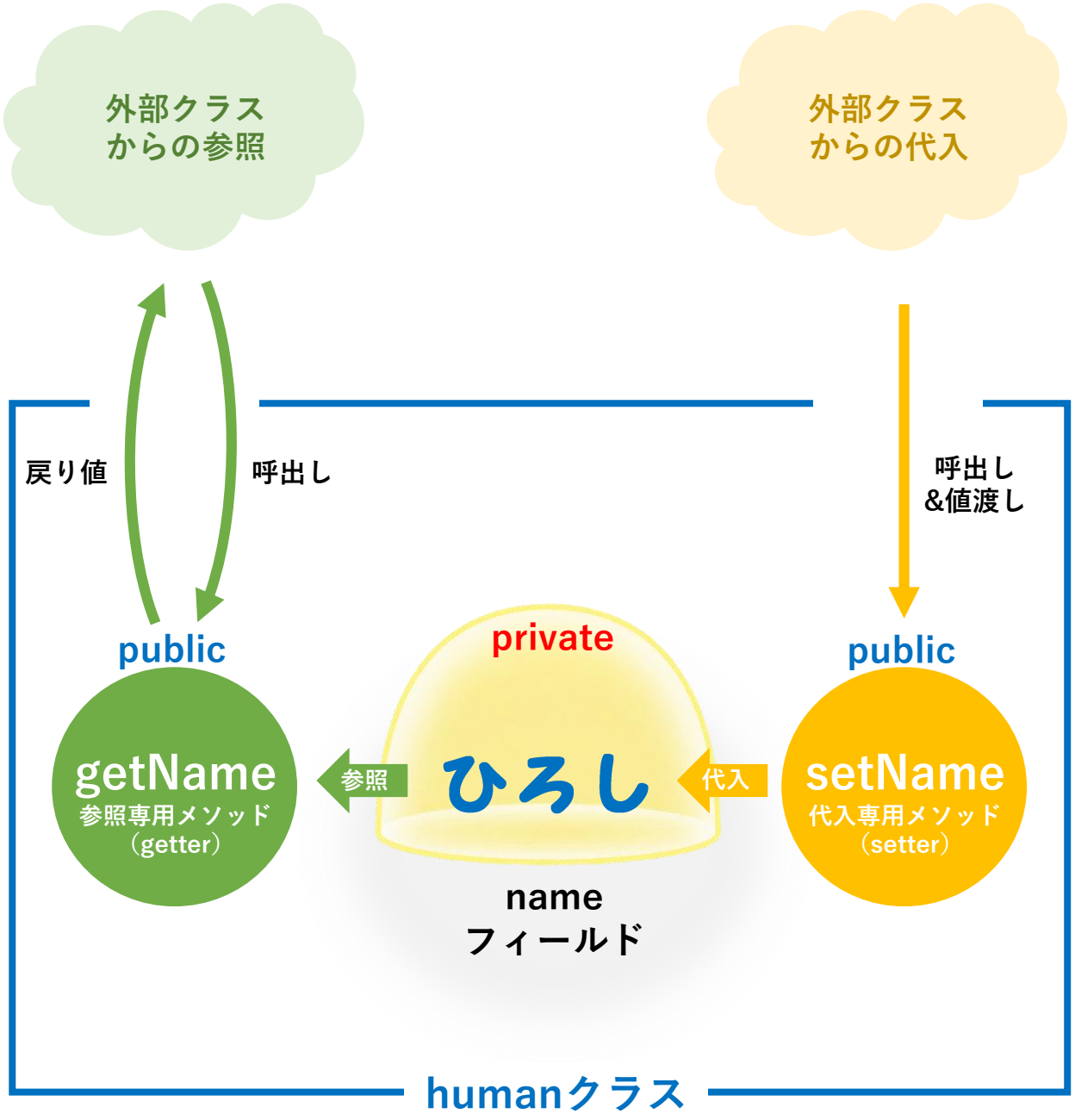
≪アクセス制御（privateとpublic）≫

□**privateは自クラス内からのアクセスのみを許す修飾子**です。
カプセル化のキモはこのprivateをフィールドやメソッドにつけることで、他クラスからその存在を隠すことにあります。

□privateとは逆に**外部クラスに公開したいフィールドやメソッドには public修飾子**をつけます。

□オブジェクト指向では**基本的にフィールドにはprivateをつけます。**
フィールドにpublicをつけるケースは極めて稀だと思ってください。
フィールドにprivateをつけると、そのフィールドは外部クラスから**変更はおろか参照すらできなくなります。**

□クラスで内部的に使うことを想定したメソッドには必ずprivateをつけましょう。外部に公開するメソッドは基本的にpublicをつけます。



≪アクセス用メソッド（getterとsetter）≫

□**privateをつけたフィールドには外部から参照・変更ができるような専用のメソッド**を必要に応じて提供します。
これを**アクセス用メソッド**と言い、参照用メソッドを**getter**、代入用メソッドを**setter**と呼びます。

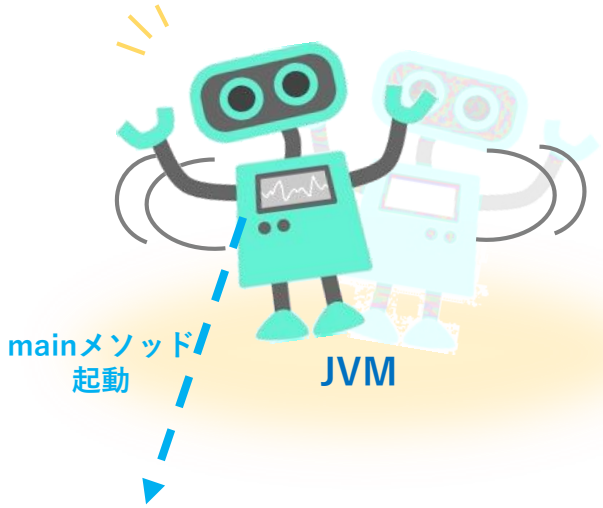
□getterは引数なしで呼び出されると、対応するprivateなフィールドの値を参照し、戻り値として返すだけのメソッドです。
以下のように記述します。（フィールド名を「abc」とする）

(public) フィールドの型 getAbc() { return abc; }

□setterは引数を対応するprivateなフィールドに代入するだけのメソッドです。
以下のように記述します。（フィールド名を「abc」とする）

(public) void setAbc(フィールドの型 引数名) { abc = 引数名 ; }

～mainメソッドにpublicが用いられる理由～




```
public static void main(String[] args) {↵
^    ↵
^    //メソッドのしくみ↵
^    ↵
^    int print = sum( 2 , 3 ) ;↵
^    ↵
^    System.out.println("print : " + print);↵
^    ↵
}↵
```

public static void main (String[] args) {　}

mainメソッドはJVMから呼び出されますが、
JVM自体がクラスの外にあるためpublicが必要になります。

クラスに使用可能なアクセス修飾子

ｱｸｾｽ修飾子	説明
public	どこからでもアクセス可能
無指定	同パッケージ内からのアクセスのみ可能

メンバに使用可能なアクセス修飾子

ｱｸｾｽ修飾子	説明
public	（publicクラスの場合）どこからでもアクセス可能
protected	同パッケージ内とサブクラスからアクセス可能
無指定	同パッケージ内からのアクセスのみ可能
private	同クラス内からのアクセスのみ可能

≪その他のアクセス制御≫

□**クラス**にもアクセス修飾子「**public**」「**無指定**」が存在します。
アクセス修飾子を何もつけない（無指定）場合、パッケージ外からクラスへのアクセスはできません。
パッケージ外からのアクセスを許す場合にのみpublicをつけます。

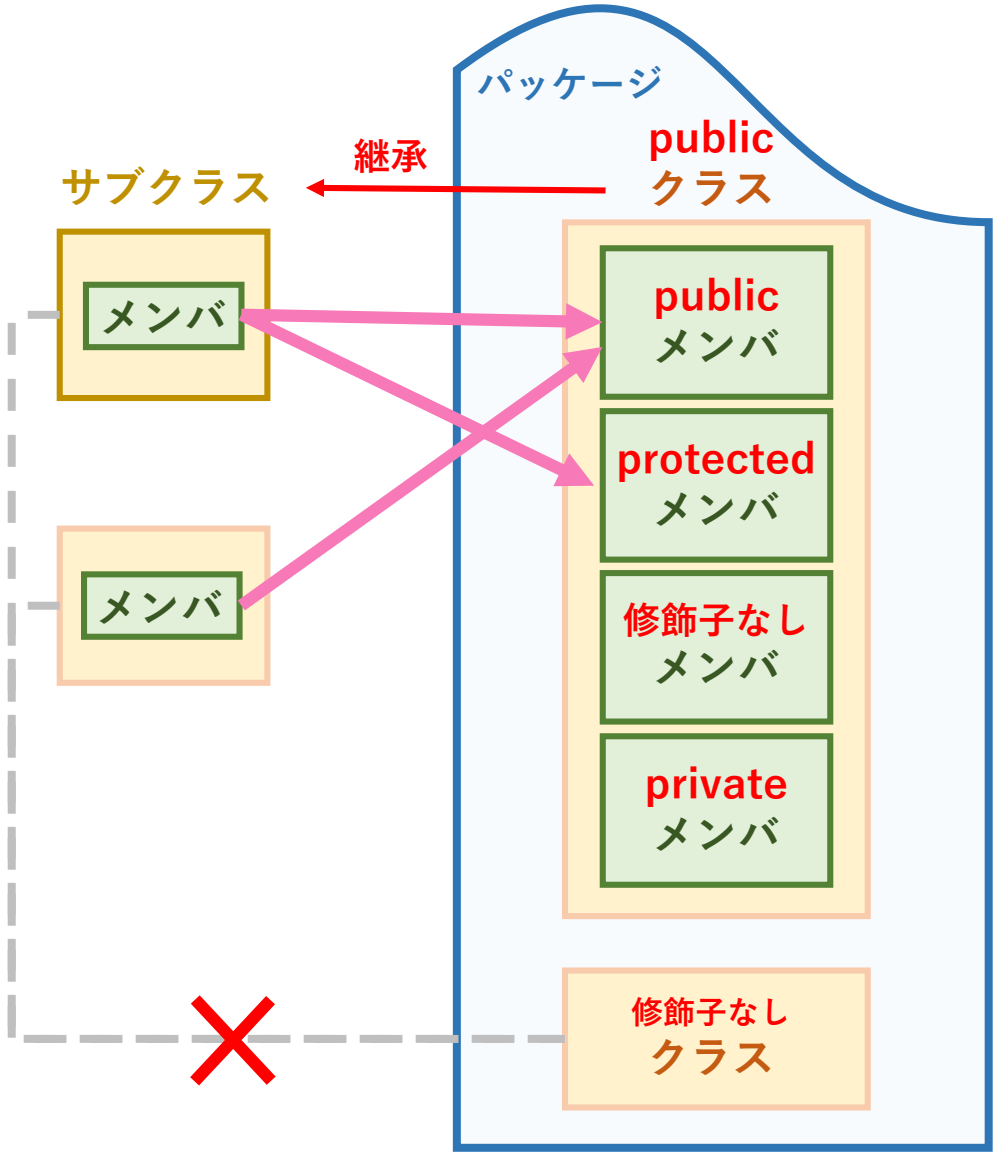
□**フィールド**に使用できるアクセス修飾子は「public」「private」の他に「**protected**」「**無指定**」があります。
クラスと同じくアクセス修飾子を何もつけない（無指定）場合、パッケージ外からフィールドへのアクセスはできません。
protectedについては後ほど学ぶ「継承」と一緒にご説明しますので今は気にしなくて結構です。

□protectedを除き、アクセス制御で抑えておくべきポイントは以下のみです。

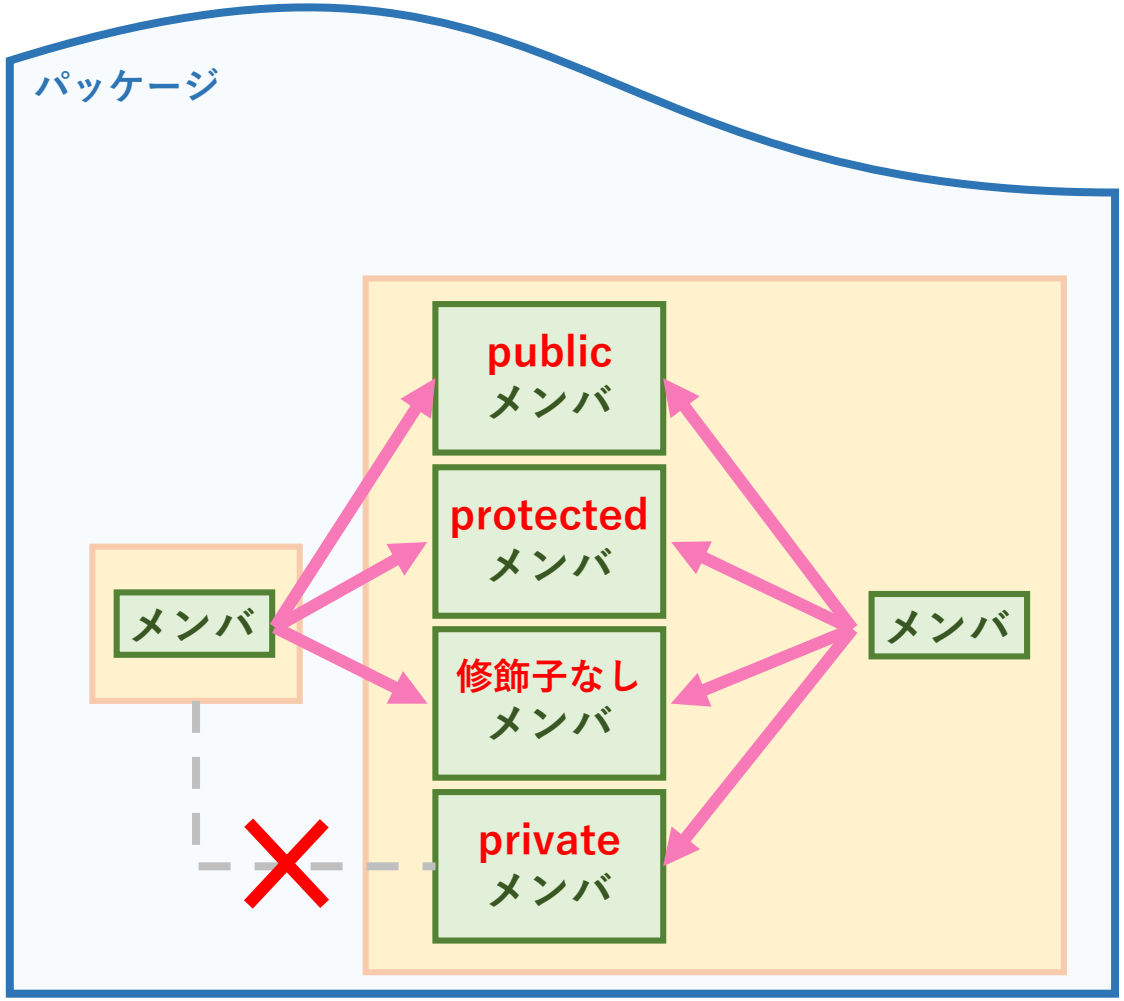
- **同じクラス内ではアクセス制限がない**
- **同パッケージ内の別クラスからのアクセスはprivateでのみ制限**
- **外部パッケージのクラスからのアクセスは基本的にpublic以外不可**

～Javaにおけるアクセス制御～

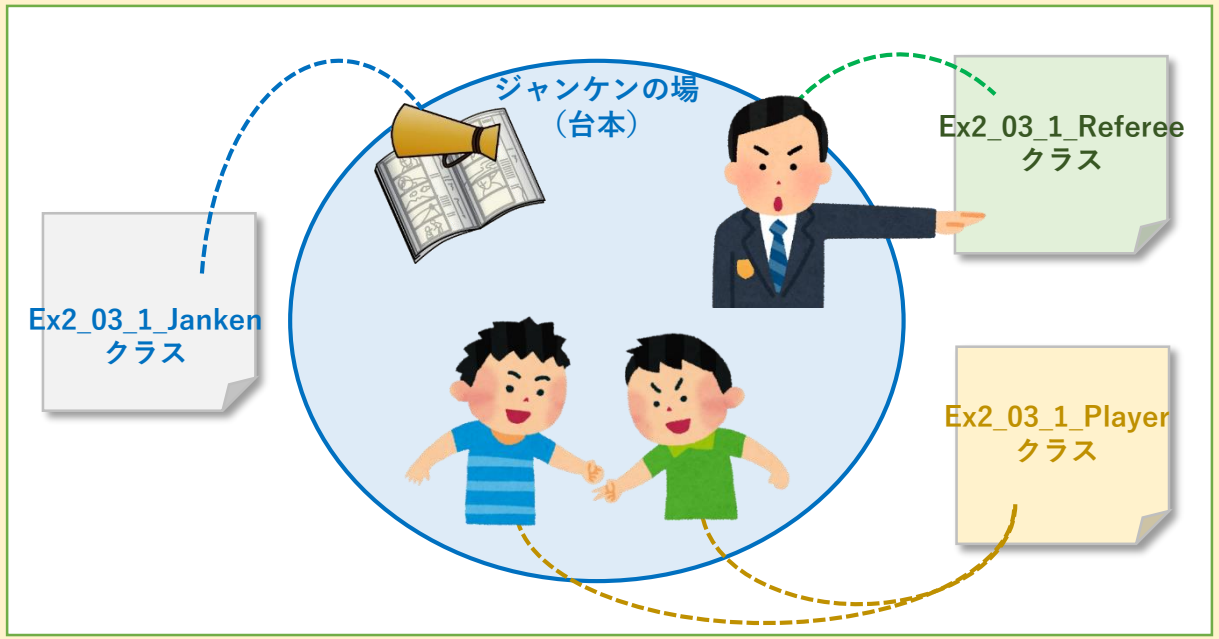
パッケージ外からのアクセスルート



同パッケージ内におけるアクセスルート



＜演習：Ex2_03_1＞
前回の演習で作成したEx2_02_2クラスを流用し、右の結果が得られるようなプログラムを作成しましょう。




```
C:¥WorkSpace>java Ex2_03_1_Janken モコ チョコ 審判タロー
審判タロー「じゃんけん・・・ぽん！！！！！」
審判タロー「モコさんの手はチョキでした！」
審判タロー「チョコさんの手はチョキでした！」
審判タロー「結果は・・・」
審判タロー「あいこ！勝負つかず！」

C:¥WorkSpace>java Ex2_03_1_Janken モコ チョコ 審判タロー
審判タロー「じゃんけん・・・ぽん！！！！！」
審判タロー「モコさんの手はチョキでした！」
審判タロー「チョコさんの手はグーでした！」
審判タロー「結果は・・・」
審判タロー「チョコさんの勝利！」

C:¥WorkSpace>java Ex2_03_1_Janken モコ チョコ 審判タロー
審判タロー「じゃんけん・・・ぽん！！！！！」
審判タロー「モコさんの手はチョキでした！」
審判タロー「チョコさんの手はパーでした！」
審判タロー「結果は・・・」
審判タロー「モコさんの勝利！」
```

＜演習：Ex2_03_1＞
前回の演習で作成したEx2_02_2クラスを流用し、右の結果が得られるようなプログラムを作成しましょう。

▼Ex2_03_1_Playerクラス
以下を除いてEx2_02_2_Playerクラスに同じ。

・フィールド ※初期化はしないこと！
- name（String型）　　…プレイヤー名（**private**）
- handStatus（String型）　…ジャンケンの手（**private**）

・コンストラクタ

<引数>
String型の文字列を1つ受け取る。

<処理>
引数で受け取った文字列をnameに設定。

・メソッド
- makeHandStatusメソッド

<引数 / 戻り値>
なし

<処理>
handStatusに「グー」「チョキ」「パー」のいずれかを
ランダムに設定。

- getter（ name ）
- getter（ handStatus ）

▼Ex2_03_1_Refereeクラス

・フィールド ※初期化はしないこと！
- name（String型）　…審判の名前（**private**）

・コンストラクタ

<引数>
String型の文字列を1つ受け取る。

<処理>
引数で受け取った文字列をnameに設定。

・メソッド
-（ Ex2_02_2 で定義したメソッド）
- getter（ name ）

▼Ex2_03_1_Jankenクラス
基本的な内容はEx2_02_2_Jankenクラスに同じ