Java基礎演習
(オブジェクト指向編)

スコープ
と
アクセス修飾子

スコープとアクセス修飾子

「基礎構文編」では、以下のようなプログラムがありました。

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

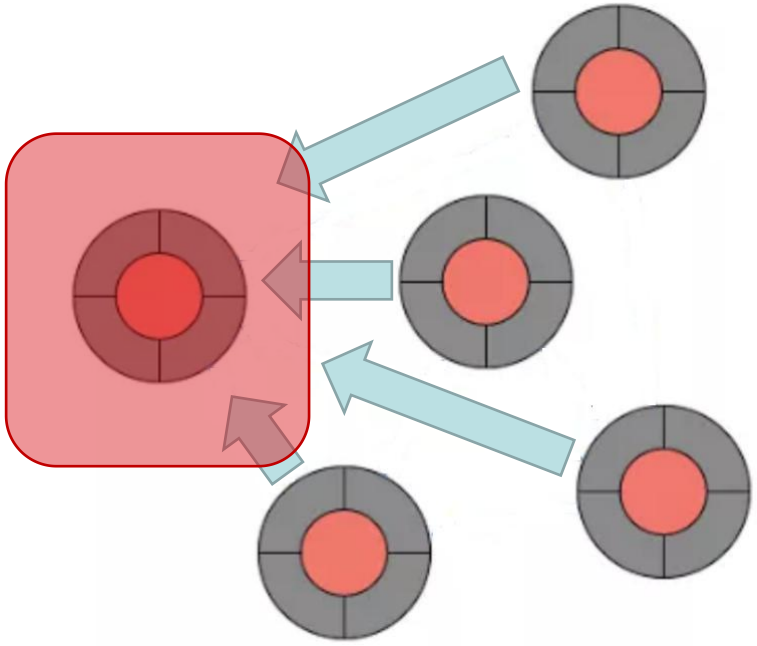
スコープとアクセス修飾子

クラスやメソッドの先頭には、publicという宣言がされています。

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

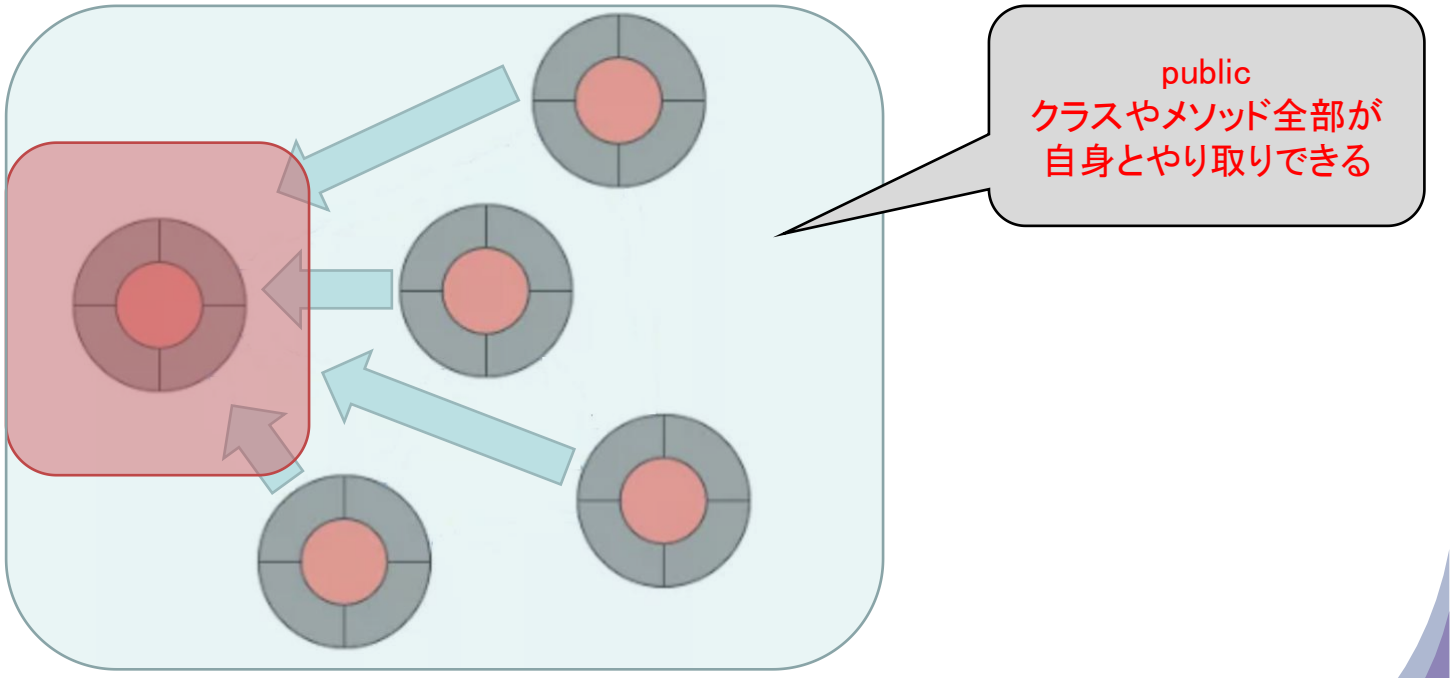
スコープとアクセス修飾子

これは、自身のクラスやメソッドが、他のクラスやメソッドからやり取りをされる際、どのように見えるかを宣言する仕組みです。



スコープとアクセス修飾子

publicと書くと、他のクラスやメソッド全部が、自身のクラスやメソッドとやり取りできるようになります。



スコープとアクセス修飾子

以下ように、自身のクラスにpublicと書いた場合には、他のクラスがこのクラスとやり取りができるようにプログラムする、という意味になります。

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

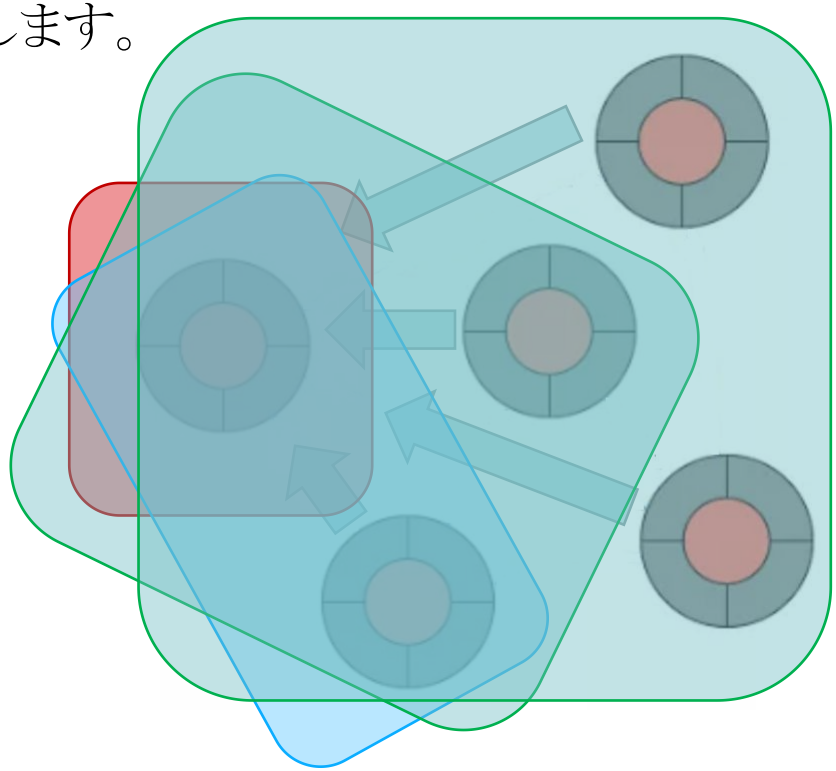
スコープとアクセス修飾子

また、自身のメソッドにpublicと書いた場合には、他のクラスのメソッドがこのメソッドとやり取りができるようにプログラムする、という意味になります。

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

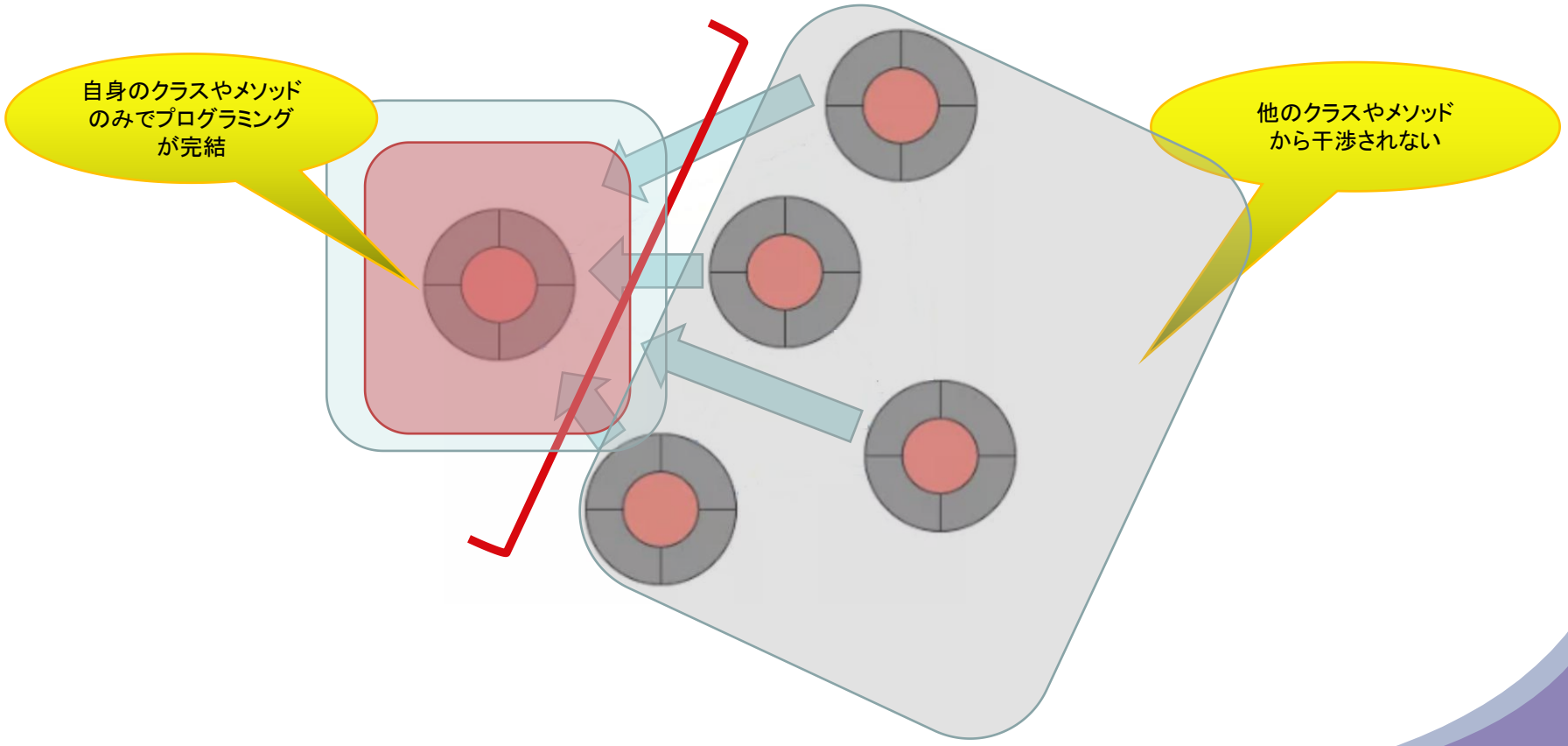
スコープとアクセス修飾子

このようなやり取りの範囲を「スコープ」といいます。
また、publicのように、これを決定する言葉を「アクセス修飾子」といいます。
Javaでは4つの「アクセス修飾子」が準備されており、我々はこれを選択しながらプログラミングします。



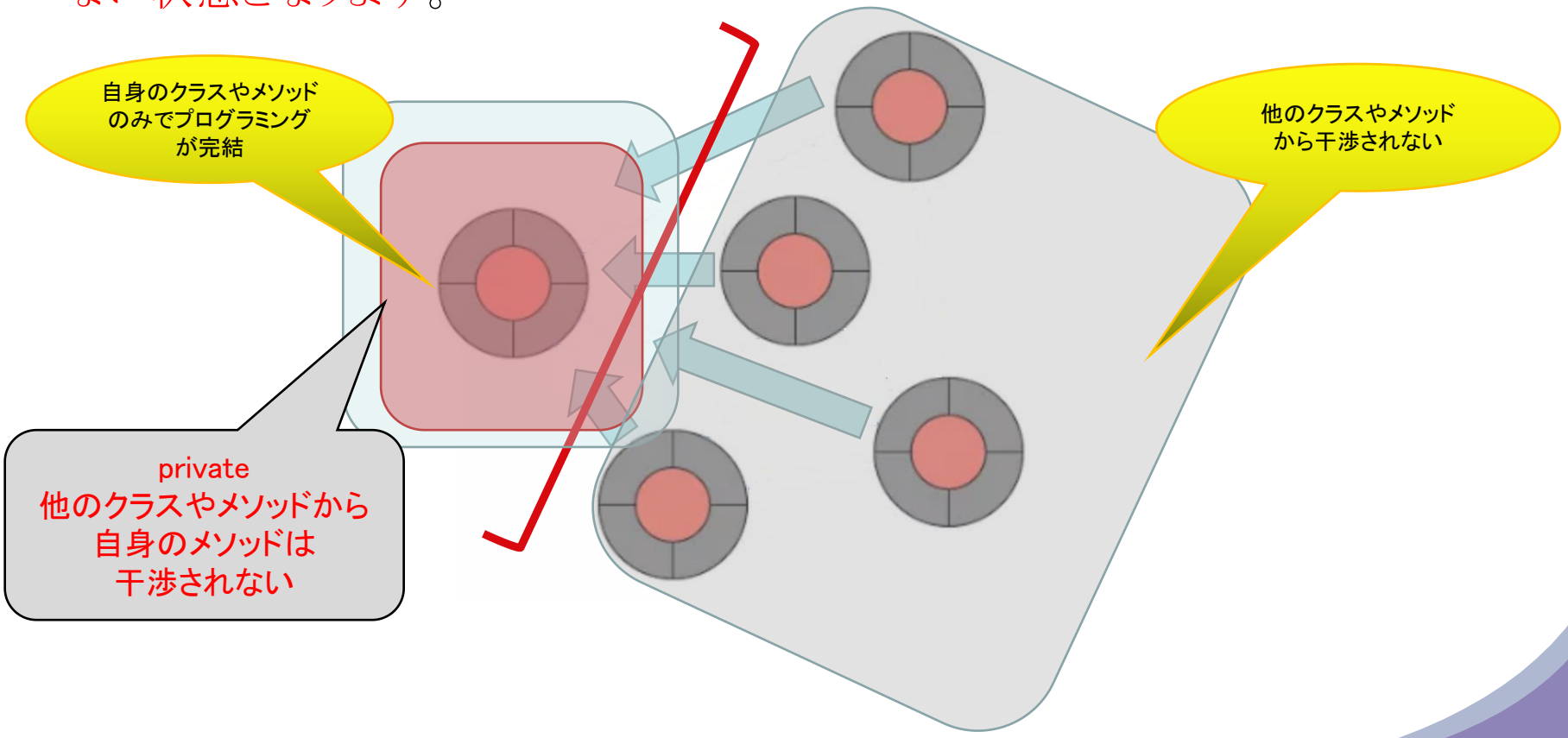
スコープとアクセス修飾子

まず、自身のクラスやメソッドのみでプログラミングが完結し、他のクラスやメソッドからもやり取りされたくない、つまり干渉されたくない場合があります。



スコープとアクセス修飾子

こうした場合には、privateと書くと実現できます。publicの代わりにprivateと書いた場合、自身のメソッドは他のクラスやメソッドからやり取りができない、干渉されない状態となります。



スコープとアクセス修飾子

具体的には、以下ようなメソッドにprivateと書いた場合、他のクラスやメソッドからやり取りはできなくなります。

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

スコープとアクセス修飾子

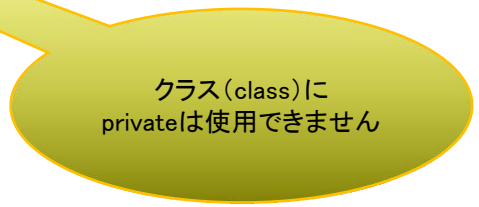
なお、クラス部分にprivateは使用できません。

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

スコープとアクセス修飾子

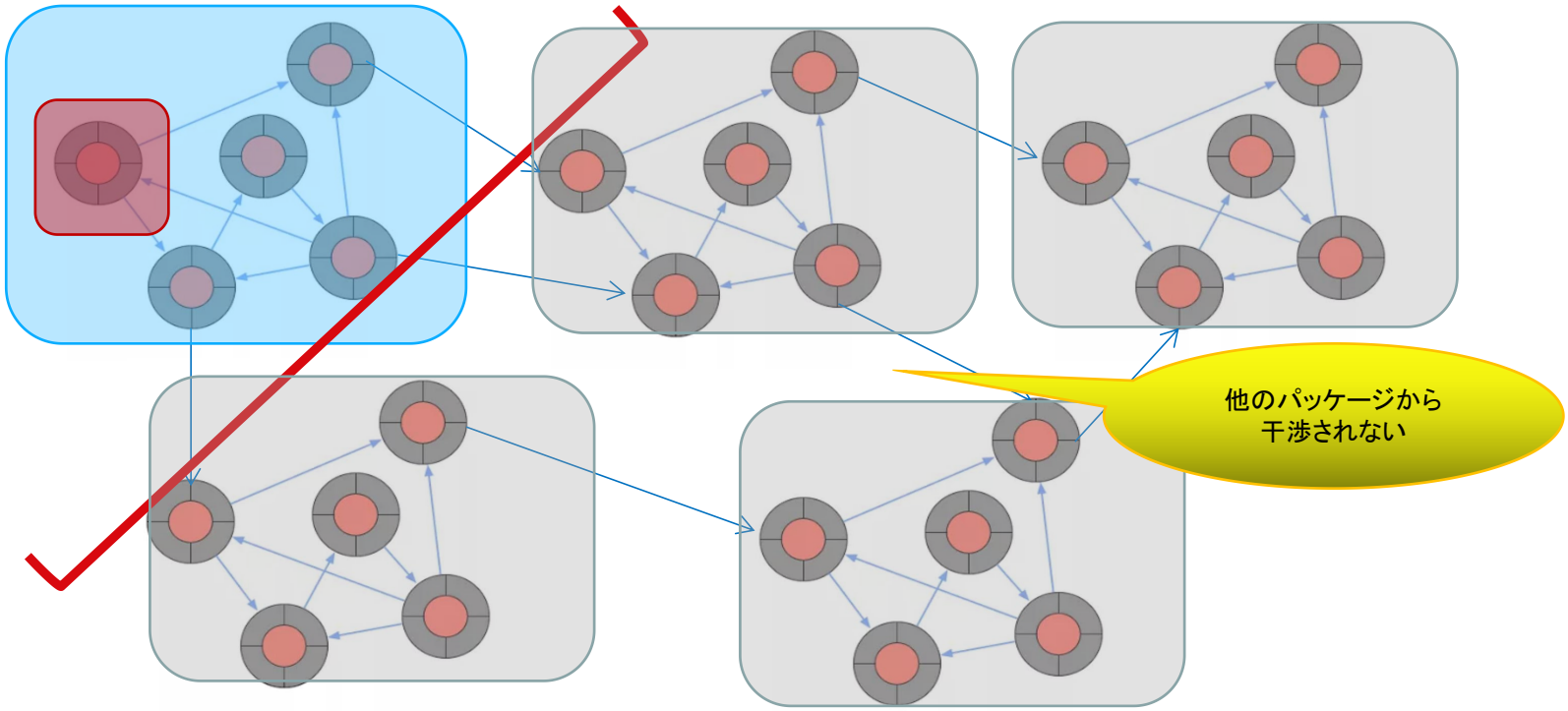
これは、1つのクラスだけで実行する場合でも、実は実行命令をおこなう為のやりとりが裏側で発生している為です。privateは、メソッドの場合のみ使用できます。

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



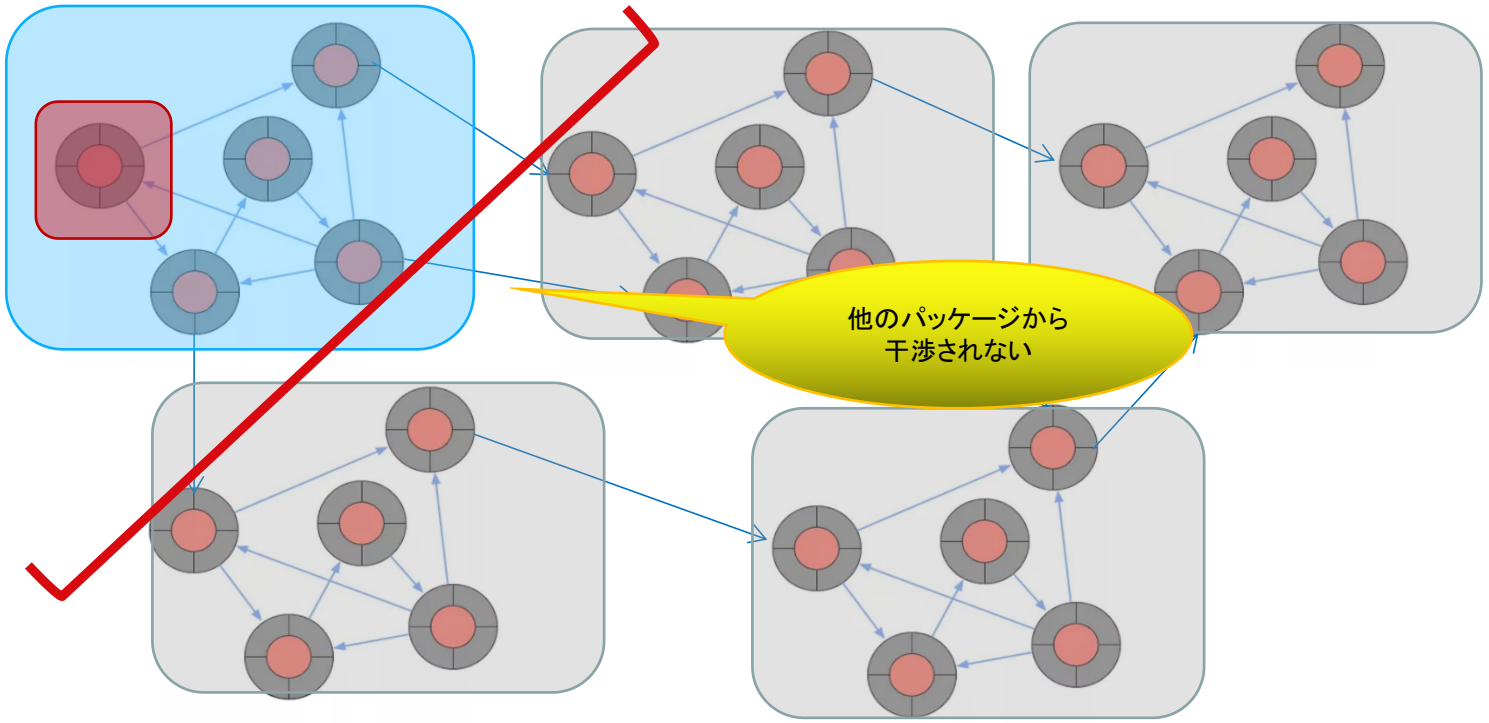
スコープとアクセス修飾子

つぎに、自身のクラスが保存されているフォルダ、つまり同じ「パッケージ」内のクラス同士だけでやり取りをしたい場合があります。



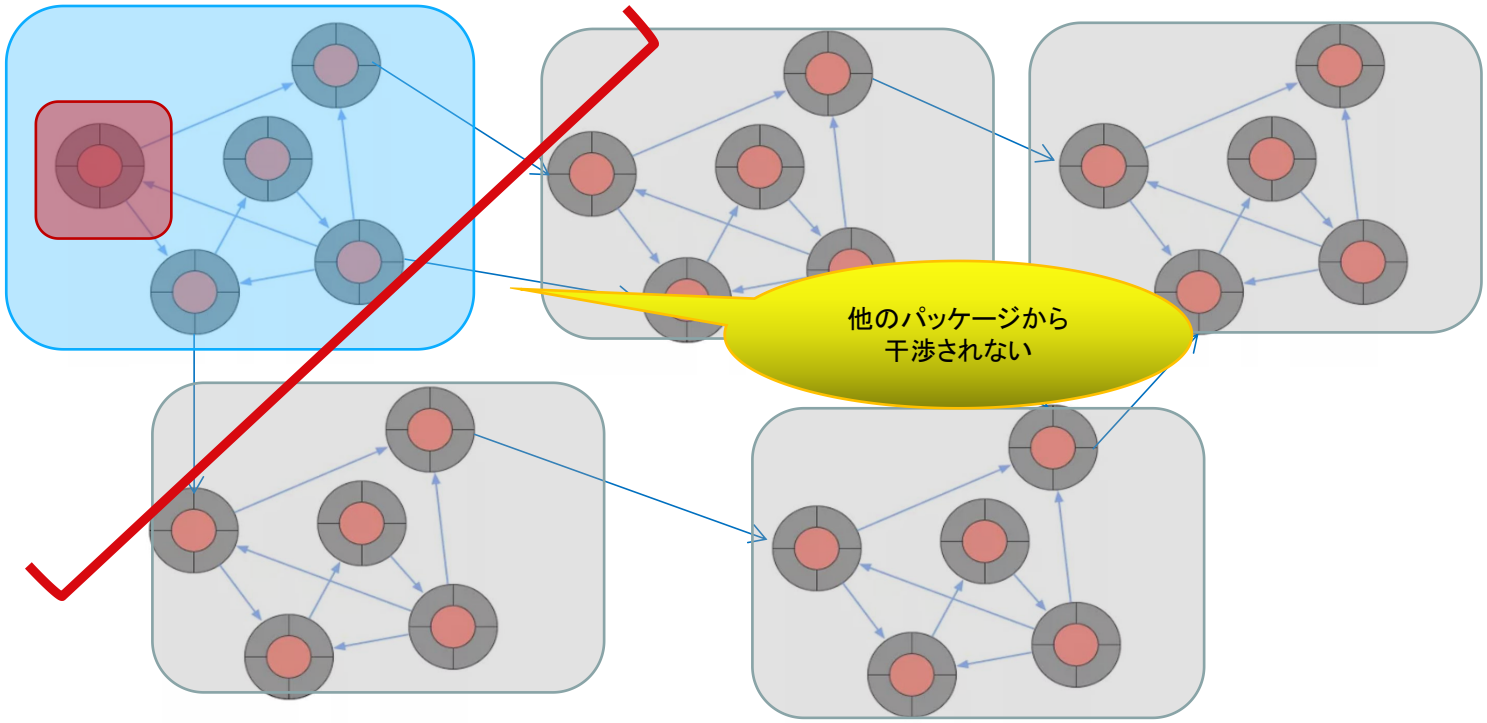
スコープとアクセス修飾子

こうした場合には、publicやprivateなどを省略して書くと実現できます。
この「アクセス修飾子」を省略した状態のことを、「デフォルト」といいます。



スコープとアクセス修飾子

publicやprivateの代わりに何も書かなければ、自身のクラスやメソッドは他のパッケージから干渉されない状態となります。



スコープとアクセス修飾子

以下のように「アクセス修飾子」を省略した場合、自身のパッケージ内でやり取りができる（他のパッケージからやり取りはできなくなる）、という意味になります。

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

スコープとアクセス修飾子

なお、メソッドに対しても「デフォルト」は記述することができます。

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

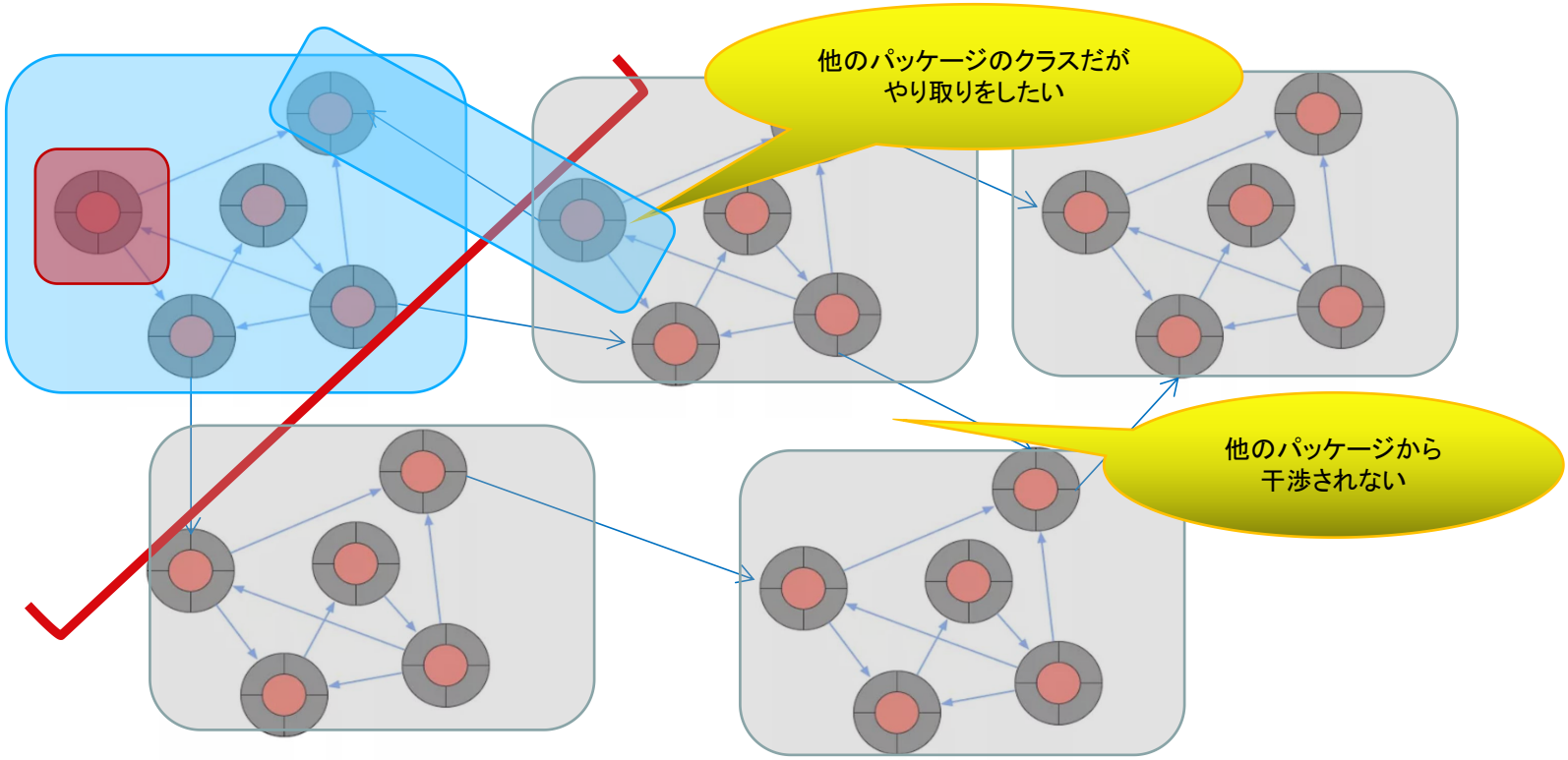
スコープとアクセス修飾子

以下の場合、自身のパッケージに属するクラスやメソッドとやり取りできる、という意味になります。（他のパッケージからやり取りはできません。）

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

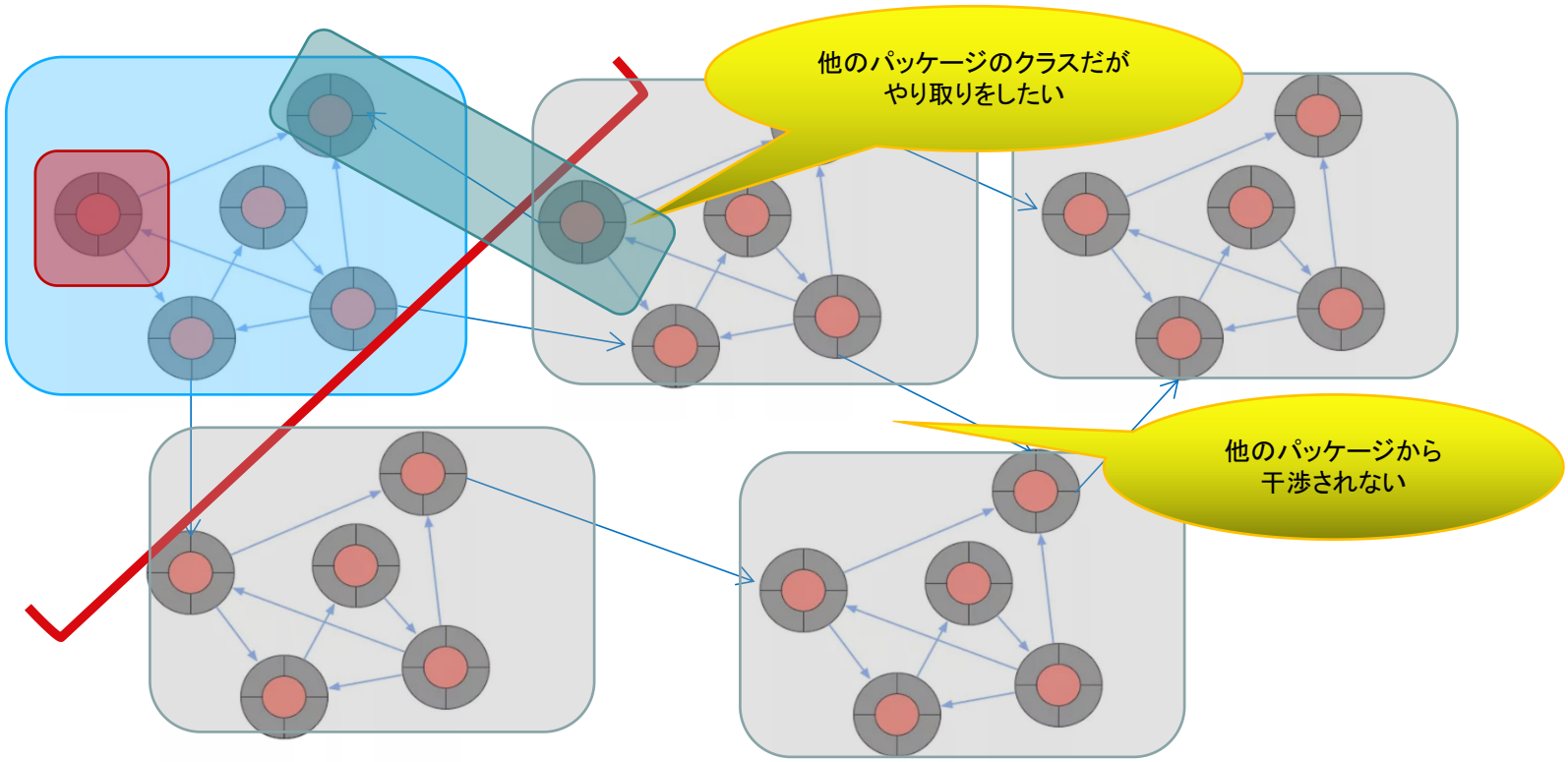
スコープとアクセス修飾子

最後に、同じ「パッケージ」内のクラスではないが、やり取りをしたい場合があります。



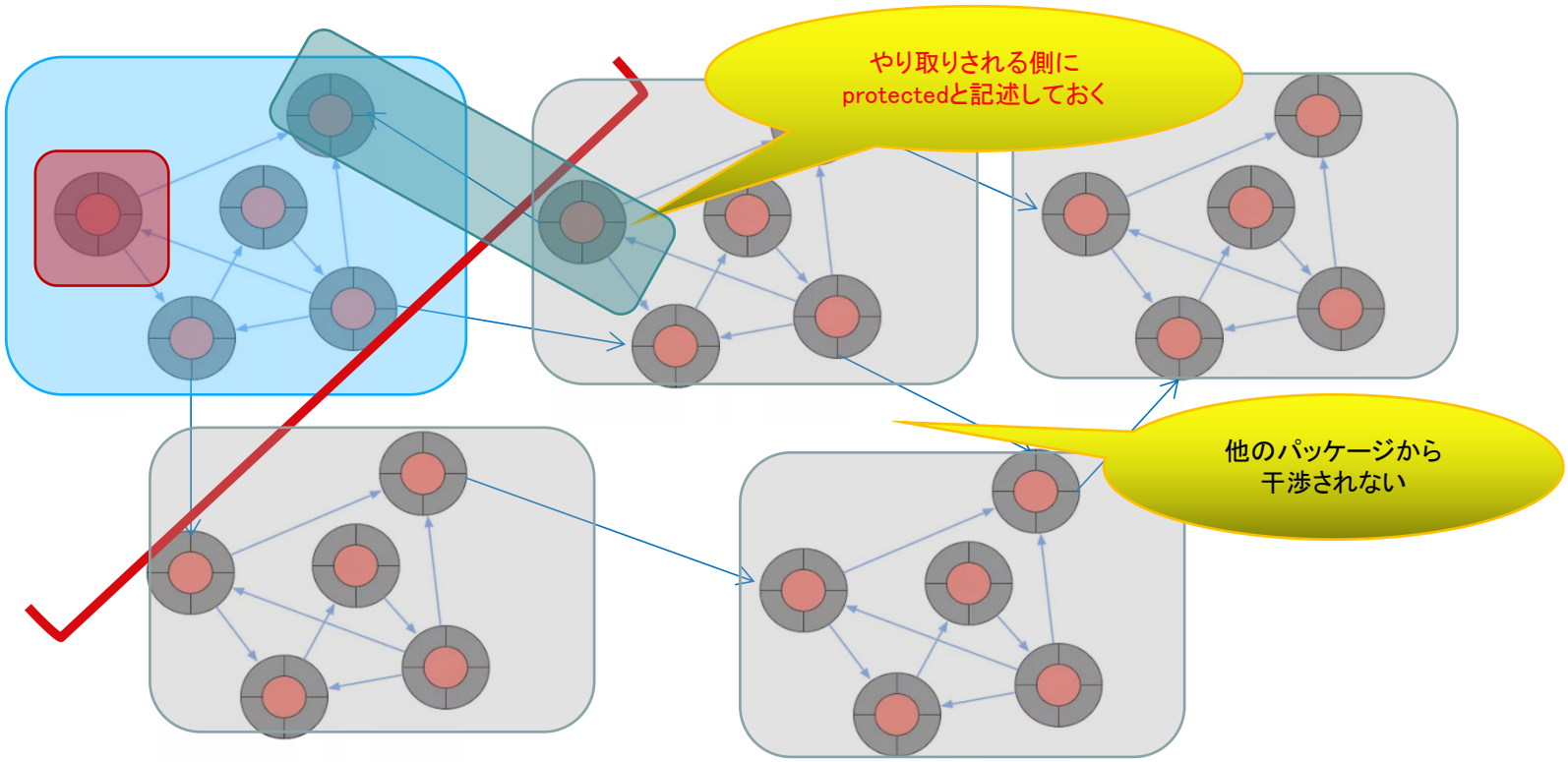
スコープとアクセス修飾子

この緑色のようなやり取りを「継承」といいます。
※この「継承」は少し難しい仕組みなので、別途詳しく解説します。



スコープとアクセス修飾子

なお、このようなやり取りをする場合には、やり取りされる側のクラスもしくはメソッドにprotectedと書くことで実現できます。



スコープとアクセス修飾子

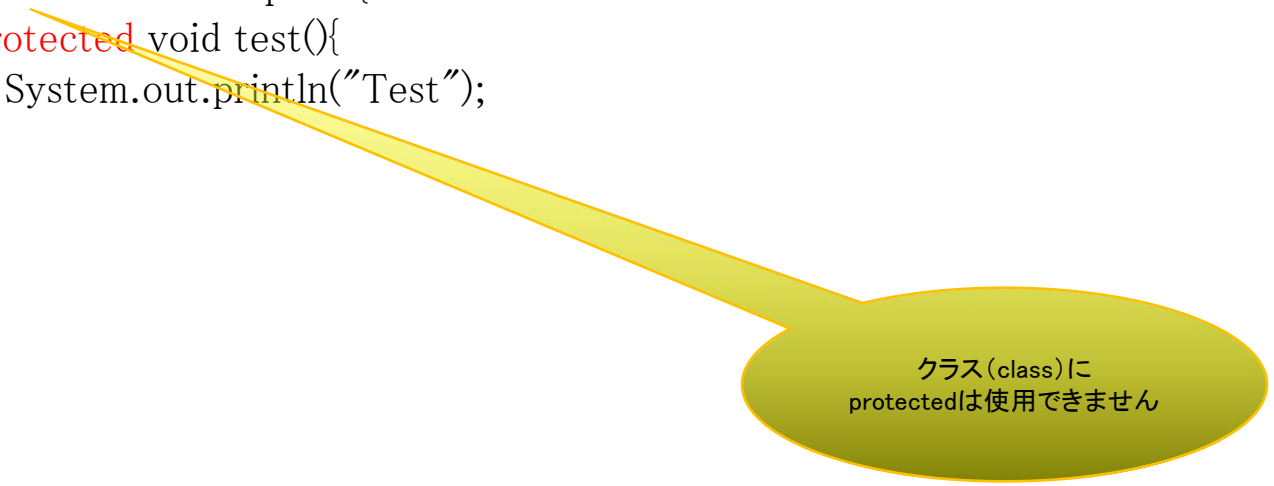
やり取りされるクラス側のメソッドにprotectedと書いた場合、「継承」によってやり取りされるクラスと自身のパッケージ内のみやり取りができます。

```
public class SampleA {
    protected void test(){
        System.out.println("Test");
    }
}
```

スコープとアクセス修飾子

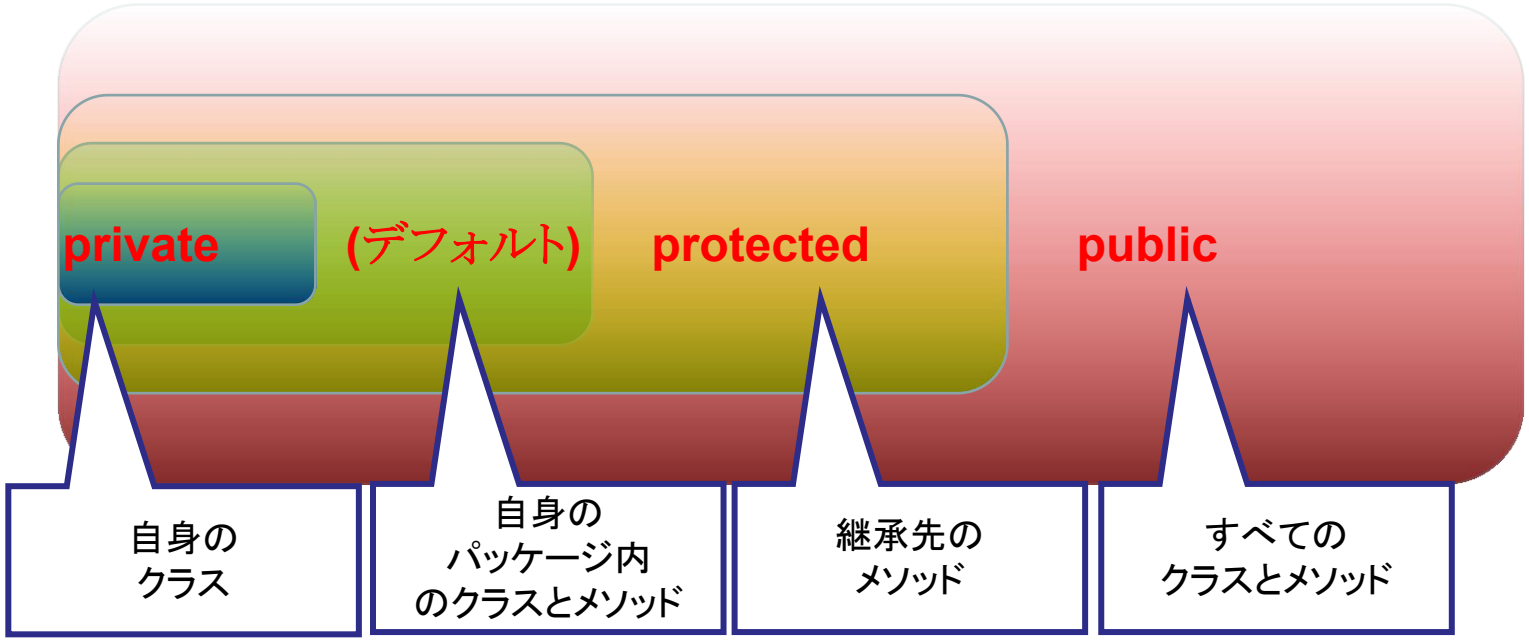
なお、クラス部分にprotectedは使用できません。これは、「継承」という仕組みが理由となるため、ここでは割愛して別途解説をおこないます。

```
protected class SampleA {
    protected void test(){
        System.out.println("Test");
    }
}
```



スコープとアクセス修飾子

このように4つの「アクセス修飾子（private/(デフォルト)/protected/public）」を使用すると、やり取りの範囲（スコープ）を決めながらプログラミングができます。



スコープとアクセス修飾子

演習：

1. すべてのクラスとやり取りしたい場合のアクセス修飾子は何が適切でしょうか？

2. 自身のパッケージ内のメソッドとやり取りしたい場合のアクセス修飾子は何が適切でしょうか？

3. 自身のクラスの中だけでメソッドを使用したい場合のアクセス修飾子は何か適切でしょうか？

4. あるクラスを「継承」して使用したい場合、この「継承」先のメソッドに使用するアクセス修飾子は何が適切でしょうか？

次のページで答え合わせをしましょう。

スコープとアクセス修飾子

演習（解答）：　答え合わせをしてみましょう。

1. すべてのクラスとやり取りしたい場合のアクセス修飾子は何が適切でしょうか？
   ⇒public

2. 自身のパッケージ内のメソッドとやり取りしたい場合のアクセス修飾子は何が適切でしょうか？
   ⇒デフォルト、何も書かない

3. 自身のクラスの中だけでメソッドを使用したい場合のアクセス修飾子は何か適切でしょうか？
   ⇒private

4. あるクラスを「継承」して使用したい場合、この「継承」先のメソッドに使用するアクセス修飾子は何が適切でしょうか？
   ⇒protected

以上