オーバライドとメソッドのアクセス性

メンバのアクセス性について前章で学習しました（p.382）。メソッドをオーバライドする際は、アクセス性と関連する、以下の規則を必ず知っておく必要があります。

重要 メソッドをオーバライドする際は、上位クラスのメソッドと**同等もしくは弱い**アクセス制限をもつ修飾子を与えなければならない。

アクセス制限の強さ／弱さの関係を示したのが **Fig.12-11** です。修飾子のアクセス制限が最も弱いのが、公開（public）アクセス制限をもつ *m1* であり、最も強いのが非公開（private）アクセス制限をもつ *m4* です。

弱い（ゆるい）

強い（きつい）

```
public class A {
    public    void m1() { }  // 公開（public）
    protected void m2() { }  // 限定公開（protected）
              void m3() { }  // パッケージ（デフォルト・修飾子無し）
    private   void m4() { }  // 非公開（private）
}
```

Fig.12-11　メソッドのアクセス制限

下位クラスでメソッドをオーバライドする際は、上位クラスのものよりも強いアクセス制限の修飾子を与えることはできません。その規則をまとめたのが **Table 12-3** です。

Table 12-3　オーバライドするメソッドに与えることのできるアクセス制限（修飾子）

A ＼ B	公開	限定公開	パッケージ	非公開
公開	○	×	×	×
限定公開	○	○	×	×
パッケージ	○	○	○	×
非公開	×	×	×	×

※A … スーパークラスにおけるメソッドのアクセス制限（修飾子）。
　B … サブクラスでオーバライドするメソッドのアクセス制限（修飾子）。

たとえば、クラス *A* から派生したサブクラスでメソッド *m1* をオーバライドする際は、必ず宣言に **public** を付けなければなりません（**public** を付けなければコンパイルエラーとなります）。

また、メソッド *m2* をオーバライドする際は、必ず **public** あるいは **protected** を付ける必要があります。

▶　派生元スーパークラスと派生したサブクラスのそれぞれが **public** であるかどうかとは関係なく、ここに示した規則に基づいて、**public** や **protected** などのアクセス修飾子を与えます。

なお、非公開メソッドは、（サブクラスで利用できる形としては）継承されないため、当然オーバライドできません。

ただし、クラス *A* から派生したサブクラスでメソッド *m4* を定義することができます。というのも、以下の規則があるからです（同じ名前の別ものとして扱われます）。

重 要 非公開メソッドを、下位クラスで同一シグネチャ・同一返却型のメソッドとして定義しても、オーバライドとみなされるのではなく、たまたま同じ仕様の無関係なメソッドとなる。

*

第 9 章では、『文字列表現を返すメソッド toString を定義する際は **public** メソッドとする』という指針を学習しました（p.319）。

Java のすべてのクラスの親玉である **Object** クラスでは、toString が **public** メソッドとして定義されています（**Column 12-6**：次ページ）。そのため、toString クラスをオーバライドする際は、必ず **public** を付ける必要があるのです。

重 要 メソッド **String** toString() は **public** 修飾子を付けて定義しなければならない。

*

なお、スーパークラスのクラスメソッドを、インスタンスメソッドとしてオーバライドすることはできません。したがって、以下のプログラムはエラーとなります。

```
class A {
   static void f() { /* … */ }      // fはクラスメソッド
}

class B extends A {
   void f() { /* … */ }              // エラー
}
```

Column 12-5 宣言における修飾子の順序

クラスやフィールドなどの宣言では、アナテイションや **public** や **final** などの修飾子で属性を指定します。修飾子が複数の場合は、どんな順序で指定しても構わないのですが、**Table 12C-1** の順で指定することが推奨されています（この表は、本書で学習しない修飾子も含んでいます）。

Table 12C-1 宣言に与える修飾子（推奨される順序）

クラス	アナテイション **public protected private abstract static final strictfp**
フィールド	アナテイション **public protected private static final transient volatile**
メソッド	アナテイション **public protected private abstract static final synchronized native strictfp**
インタフェース	アナテイション **public protected private abstract static strictfp**