複数インタフェースの実装

クラスの派生とインタフェースの実装に関して、最も大きく異なる点は、複数のクラスを同時に派生／実装できるかどうかということです。

クラスの派生が単一継承しか認められていない（複数のクラスをスーパークラスとしてもつことはできない：p.399）のとは異なって、**複数インタフェースの実装は可能です。**

重要 クラスは、複数のインタフェースを同時に実装できる。

▶ すなわち、いろいろなサークルに所属して、複数の《友人関係》を作れるわけです。複数の親とは《血縁関係》を結べない（多重継承できない）のとは異なります。

その一般的な形式を示したのが **Fig.14-7** です。クラスの宣言時は、`implements` の後に実装するインタフェースをコンマで区切って並べます。もちろん、クラス *A* では、二つのインタフェース *B* と *C* のメソッドをすべて実装することになります。

▶ すべてを実装しない場合は、クラス *A* は抽象クラスとして宣言しなければなりません（p.460）。

このクラスは、インタフェースBとCの両方を“実装”します。

```
class A implements B , C {
    // インタフェースBのメソッドの実装
    // インタフェースCのメソッドの実装
}
```

Fig.14-7 複数のインタフェースの実装

List 14-9 に示すのは、本章で作成したインタフェース *Player* と *Skinnable* の両方を実装するプログラムです。

クラス *PortablePlayer* は、“着せかえ可能な携帯プレーヤ”です。インタフェースとクラスの関係を示したのが **Fig.14-8** です。

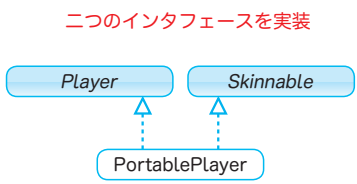

Fig.14-8 着せかえ可能な携帯プレーヤ

List 14-9 player/PortablePlayer.java

```
// 着せかえ可能な携帯プレーヤ
class PortablePlayer implements Player, Skinnable {
   private int skin = BLACK;

   public PortablePlayer() { }                                    // コンストラクタ
                                  インタフェース Player のメソッドの実装
   public void play() {                                           // ○再生
      System.out.println("◆再生開始！");
   }

   public void stop() {                                           // ○停止
      System.out.println("◆再生終了！");
   }

   public void changeSkin(int skin) {                             // ★スキン変更
      System.out.print("スキンを");
      switch (skin) {
       case BLACK:   System.out.print("漆黒");  break;
       case RED:     System.out.print("深紅");  break;
       case GREEN:   System.out.print("柳葉");  break;
       case BLUE:    System.out.print("露草");  break;
       case LEOPARD: System.out.print("豹柄");  break;
       default:      System.out.print("無地");  break;
      }
      System.out.println("に変更しました。");
   }
}                                 インタフェース Skinnable のメソッドの実装
```

クラス *PortablePlayer* では、インタフェース *Player* のメソッド *play* と *stop* と、インタフェース *Skinnable* のメソッド *changeSkin* を実装しています。クラス内では、実装したインタフェース中のフィールドを単純名でアクセスできます。

▶ すなわち、*Skinnable.BLACK* を単なる *BLACK* でアクセスできます。

*

クラス *PortablePlayer* の利用例を **List 14-10** に示します。インスタンスを生成して、三つのメソッドを順次呼び出すだけの単純なプログラムです。

List 14-10 player/PortablePlayerTester.java

```
// クラスPortablePlayerの利用例

class PortablePlayerTester {

   public static void main(String[] args) {
      PortablePlayer a = new PortablePlayer();
      a.play();                              // 再生
      a.stop();                              // 停止
      a.changeSkin(Skinnable.LEOPARD);       // スキンを豹柄に変更
   }
}
```

実行結果

```
◆再生開始！
◆再生終了！
スキンを豹柄に変更しました。
```

インタフェース *Skinnable* 中で定義された定数は、クラス変数（静的フィールド）ですから、**“インタフェース名 . フィールド名”** でアクセスします（p.461）。本プログラムで選んでいる *Skinnable.LEOPARD* は豹柄です。