コンストラクタ

メソッドとよく似た形をした**コンストラクタ**（*constructor*）の役割は、クラスのインスタンスを初期化することです。

そのコンストラクタが呼び出されるのは、インスタンスの生成時です。すなわち、プログラムの流れが以下の宣言文を通過して、網かけ部の式が評価される際に、コンストラクタが呼び出されて実行されます。

```
1 Account adachi = new Account("足立幸一", "123456", 1000);
2 Account nakata = new Account("仲田真二", "654321",  200);
```

Fig.8-9 に示すように、呼び出されたコンストラクタは、仮引数 *n*，*num*，*z* に受け取った値をフィールド *name*，*no*，*balance* に代入します。

代入先は、*adachi.name* や *nakata.name* ではなく、“単なる *name*” です。1で呼び出されたコンストラクタでの *name* は *adachi.name* を表し、2で呼び出されたコンストラクタでの *name* は *nakata.name* を表します。

このように、フィールド名だけで表せるのは、コンストラクタが、**自分自身のインスタンスが何者であるのかを知っているからです**。図に示すように、個々のインスタンスに対して、専用のコンストラクタが存在します。

▶ 個々のインスタンスにコンストラクタを用意することは、現実には不可能です。『個々のインスタンスに対して、専用のコンストラクタが存在する』というのは、概念上のものであって、物理的にそうなっているわけではありません。コンパイルによって生成されるコンストラクタ用の内部的なコードは、実際は 1 個だけです。

*

さて、1と2の宣言を以下のように書きかえてみましょう。コンパイルすると、エラーになります。

```
Account adachi = new Account();              // エラー：引数がない
Account nakata = new Account("仲田真二");     // エラー：引数が不足
```

このことから、コンストラクタが**不完全あるいは不正な初期化を防止する**ことが分かります。コンストラクタの役目は、**インスタンスを適切に初期化する**ことです。

重 要 クラス型を宣言するときは、必ずコンストラクタを用意して、インスタンスを確実かつ適切に初期化する手段を提供しよう。

▶ construct は『構築する』という意味です。そのため、コンストラクタは**構築子**（こうちくし）と呼ばれることもあります。

*

なお、コンストラクタはメソッドとは異なり、**値を返却できません**。誤って返却型を指定しないようにしましょう。

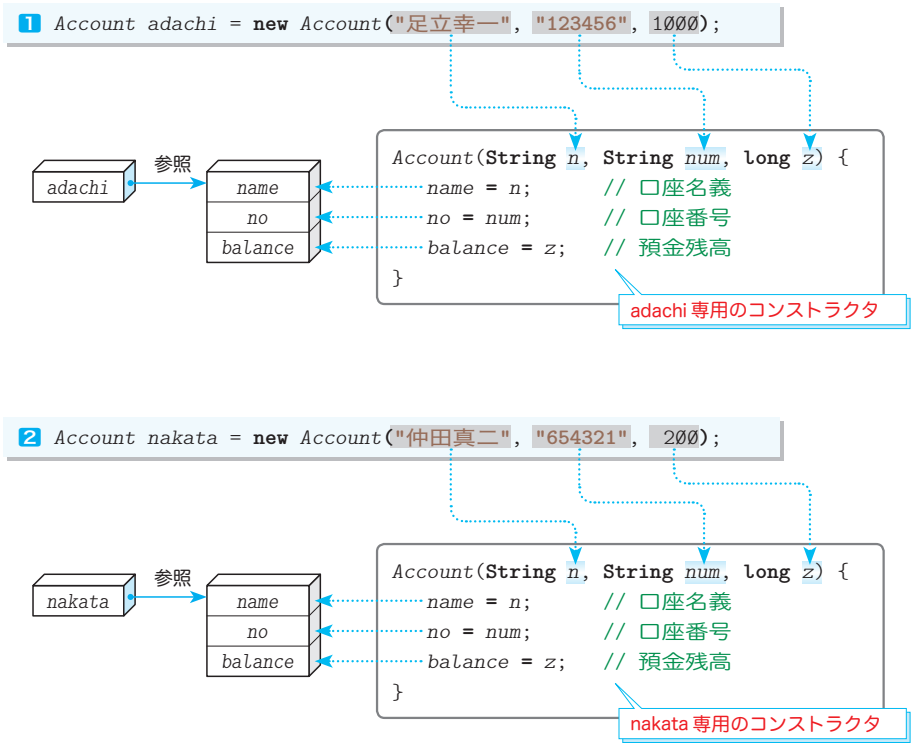

Fig.8-9 クラスのインスタンスとコンストラクタ

第 i 版のクラス *Account* ではコンストラクタを定義していませんでした。コンストラクタを定義していないのに、どうしてインスタンスを生成できたのでしょうか。

実は、コンストラクタを定義しないクラスには、引数を受け取らず、その本体が空であるデフォルトコンストラクタ（*default constructor*）が自動的に作られます。

重要 クラスにコンストラクタを定義しなければ、本体が空のデフォルトコンストラクタが自動的に定義される。

すなわち、第 1 版のクラス *Account* では、以下のコンストラクタがコンパイラによって作られていたのです。

```
Account() { }
```

▶ 第 1 版のクラス *AccountTester* で、以下のように ( ) の中を空にしてインスタンスを生成したのは、引数を受け取らないデフォルトコンストラクタを呼び出すためでした。

```
Account adachi = new Account();  // 引数を受け取らないコンストラクタを呼び出す
```

なお、デフォルトコンストラクタの内部は、本当は空ではありません。第 12 章で学習します。