

## 第 15 章 暗黙のパラメータと暗黙の型変換

暗黙のパラメータと暗黙の型変換は、既存ライブラリのカスタマイズや高レベルの抽象に役立つ強力なツールです。例として、未特定の演算 `add` を持つ半群の抽象クラスから始めましょう。

```
abstract class SemiGroup[A] {
  def add(x: A, y: A): A
}
```

`SemiGroup` に `unit` 要素を加えたサブクラス `Monoid` はこうなります。

```
abstract class Monoid[A] extends SemiGroup[A] {
  def unit: A
}
```

次はモノイドの 2 つの実装です。

```
object stringMonoid extends Monoid[String] {
  def add(x: String, y: String): String = x.concat(y)
  def unit: String = ""
}

object intMonoid extends Monoid[Int] {
  def add(x: Int, y: Int): Int = x + y
  def unit: Int = 0
}
```

任意のモノイド上で動作する `sum` メソッドは、純粋な Scala で次のように書けます。

```
def sum[A](xs: List[A])(m: Monoid[A]): A =
  if (xs.isEmpty) m.unit
  else m.add(xs.head, sum(m)(xs.tail))
```

この `sum` メソッドは、次のように呼び出せます。

```
sum(List("a", "bc", "def"))(stringMonoid)
sum(List(1, 2, 3))(intMonoid)
```

これらはみんな動作しますが、いまいちです。問題は、それを使うコードすべてにモノイドの実装を渡さなくてはならないことです。型引数を推論してくれるのと同様に、システムが自動的に正しい型引数を判別してくれたらいいのに、と思う場面がときどきあります。それこそが、暗黙のパラメータが提供するものです。

### 暗黙のパラメータ : 基本 (Implicit Parameters: The Basics)

Scala 2 では、パラメータリストのはじまりで使用できる新しいキーワード `implicit` があります。

```
ParamClauses ::= {' (' [Param {' , ' Param} ' ) ' ]
                  [' (' implicit Param {' , ' Param} ' ) ' ]
```

このキーワードがある場合、そのリストのすべてのパラメータが暗黙のパラメータとなります。たとえば、次のバージョンの `sum` では `m` は暗黙の引数となっています。

```
def sum[A](xs: List[A])(implicit m: Monoid[A]): A =
  if (xs.isEmpty) m.unit
  else m.add(xs.head, sum(xs.tail))
```

この例を見ると分かるように、通常の引数と暗黙のパラメータは組み合わせることができます。ただし暗黙のパラメータリストは、1 つのメソッドまたはコンストラクタに対してただ 1 回だけ指定でき、かつ、それは末尾にこななくてははいけません。

さらに、implicit は、定義や宣言に対する修飾子としても使えます。例をあげます。

```
implicit object stringMonoid extends Monoid[String] {
  def add(x: String, y: String): String = x.concat(y)
  def unit: String = ""
}
implicit object intMonoid extends Monoid[Int] {
  def add(x: Int, y: Int): Int = x + y
  def unit: Int = 0
}
```

暗黙のパラメータの基本的な考え方は、メソッド呼び出しからその引数指定を省略する、というものです。もし、暗黙のパラメータセクションに対応する引数がなければ、Scala コンパイラによって推論されます。

暗黙のパラメータに渡されうる実際の引数は、メソッド呼び出しの時点でアクセス可能で、暗黙の定義やパラメータであると印が付けられた、プレフィックスのつかないすべての識別子 X です。

もし、型が適合して暗黙のパラメータに渡しうる引数が複数ある場合、Scala コンパイラは静的オーバーロード解決の標準的な規則に従い、最も下位の特定の引数を選びます。たとえば、次の呼び出しにおいて、

```
sum(List(1, 2, 3))
```

stringMonoid と intMonoid が 見えている状況を考えてみます。sum の型パラメータ a は、int にインスタンス化される必要があります。形式上の暗黙のパラメータの型 Monoid[Int] に適合する値は intMonoid のみであり、この object が暗黙のパラメータとして渡されます。

この議論は、暗黙のパラメータの推論はすべての型引数が推論された後に行われる、ということも示しています。

## 暗黙の型変換

型 S であることが期待される型 T の式 E を考えましょう。T は、S に適合せず、事前定義された変換でも S に変換できないものとします。その場合、Scala コンパイラは、最後の手段として暗黙の型変換 I(E) の適用を試みます。ここで I は、暗黙定義または暗黙のパラメータであることを示す識別子で、変換の時点でプレフィックスなしでアクセスでき、型 T である引数に適用可能で、変換の結果が期待されている型 S に適合するものです。

暗黙の型変換はメンバーの選択にも適用できます。選択 E.x が与えられ、ただし x は型 E のメンバーではない場合、Scala コンパイラは x が I(E) のメンバーとなるよう、暗黙の型変換 I(E).x の挿入を試みます。

整数をクラス scala.Ordered のインスタンスに変換する、暗黙の型変換関数の例をあげます。

```
implicit def int2ordered(x: Int): Ordered[Int] = new Ordered[Int] {
  def compare(y: Int): Int =
    if (x < y) 1
    else if (x > y) -1
    else 0
}
```

## 可視境界

可視境界は、暗黙のパラメータのための便利な糖衣構文です。たとえば、ジェネリックなソートメソッドを考えてみましょう。

```
def sort[A <% Ordered[A]](xs: List[A]): List[A] =
  if (xs.isEmpty || xs.tail.isEmpty) xs
  else {
    val {ys, zs} = xs.splitAt(xs.length / 2)
    merge(ys, zs)
  }
```

可視境界をもつ型パラメータ [a <% Ordered[a]] は、型 a から Ordered[A] への暗黙の型変換が存在する型 a のリストに対して、sort が適用できることを表します。この定義は、次のような暗黙のパラメータを持つメソッドシグネチャの略記として扱われます。

```
def sort[A](xs: List[A])(implicit c: A => Ordered[A]): List[A] = ...
```

(ここで、パラメータ名 `c` は、プログラム内のほかの名前と衝突しない方法で任意に選ばれます)

もっと詳しい例として、上記の `sort` に付随している `merge` メソッドを考えましょう。

```
def merge[A <% Ordered[A]](xs: List[A], ys: List[A]): List[A] =
  if (xs.isEmpty) ys
  else if (ys.isEmpty) xs
  else if (xs.head < ys.head) xs.head :: merge(xs.tail, ys)
  else if ys.head :: merge(xs, ys.tail)
```

可視境界が展開され暗黙の型変換が挿入されると、このメソッドの実装は次のようになります。

```
def merge[A](xs: List[A], ys: List[A])
  (implicit c: A => Ordered[A]): List[A] =
  if (xs.isEmpty) ys
  else if (ys.isEmpty) xs
  else if (c(xs.head) < ys.head) xs.head :: merge(xs.tail, ys)
  else if ys.head :: merge(xs, ys.tail)(c)
```

このメソッド定義の終わりの2行が、暗黙のパラメータ `c` の2種類の使用例になっています。最後から2行目の条件における変換で適用され、最終行では `merge` の再帰呼び出しで暗黙の引数として渡されます。

[前ページ](#) [15章](#) [目次](#) [次ページ](#)

- “second to last”は 最後から2行目 の意味です。 -- pomu0325 (2009-12-29 19:45:05)

名前:	<input type="text"/>
コメント:	<input type="text"/>

投稿