Partial types

クラスの分割記述

Partial types は、一つのクラス、構造体、インタフェースを複数のファイルなどに分割して記述することができる機能です。従来 Objective-C や Smalltalk にはカテゴリという概念で存在していましたが、C 言語をベースに派生したオブジェクト指向言語にはなかなか実装されなかった機能です。

Partial types の意味はデザイン上の問題だけで、プログラムの実行にかかわるものではありません。しかし、より大きな規模のプロジェクトにおいてこの機能は極めて重要で、筆者個人としては Generics と同じかそれ以上に重要な存在だと考えています。

例えば、GUI プログラムにおいて、Form を継承したメインウィンドウを担当するクラスを作成するときです。商業レベルのツールを開発すると、高度で複雑なコントロールをいくつも作成しなければなりません。これらのコントロールを動的に表示させたり、消したりするには様々なイベント処理が必要となるため、結果としてコールバックメソッドが大量に必要となります。

Partial types を使えば、数多くのコントロールを宣言する非公開フィールドや、イベント用メソッド、プロパティ、非公開のマクロ的性質のメソッドなどを異なるファイルに分離して記述することができます。これらの全てを単一のファイルに記述すると、数千行の量になることは珍しくありません。

Partial types を利用するには **partial** 修飾子を型の宣言に指定します。partial はクラス、構造体、インタフェースのいずれかに指定することができます。

partial class 型名 { ... }

例えばこの宣言は、Partial class を宣言しています。そのため、同様にこのクラスを別の場所で宣言することで、このクラスのメンバを分離して記述することができます。

```
partial class Test {
 static void Main() {
  System.Console.WriteLine(new Test());
 }
}

partial class Test {
 public override string  ToString() {
  return "Test : ふるふるふるむーーーーん";
 }
}
```

このプログラムでは Test クラスを重複して宣言していますが、partial 宣言によってそれらのクラスは同一のクラスであることが分かります。コンパイラは、これらの記述を同一のクラスとして認識してくれます。

極めてコード量が多くて管理が難しかったクラスなどは、partial によって複数のクラスに分割するなどして、管理しやすくするとよいでしょう。