世界はオブジェクトの海に浮かぶ

.NET Frameworkで楽しむオブジェクト指向

第2回 「オブジェクト指向設計」事始め

επιστημη
えぴすてーめー

Level

1 2 **3** 4 5

Technology Tools

- ❑ Visual Basic
- ☑ Visual C#
- ☑ Visual C++
- ❑ SQL Server
- ❑ Oracle
- ❑ Access
- ❑ ASP.NET
- ❑ Other:

Samples

・この記事で取り上げたソースコードおよびサンプルプログラムは、http://www.shoeisha.com/mag/windev/からダウンロード可能です。

開発言語のテイスティング

昨年末から中国茶にハマりまくっています。中国茶イコール烏龍茶、ではありません。中国茶は製法によって大きく7種類に分類されていて、いわゆる烏龍茶はその中の1種類。銘柄は数千あると言われている中国茶を片っ端から試してみたいのですがそれは無理ってもんで、行きつけの茶舗で扱っている茶葉をひとつずつ試してはお気に入りを探しています。お陰で今までガブガブ飲んでたソフトドリンクをほとんど飲まなくなりました。いやー面白いっす、中国茶。茶葉それぞれのクセというか個性があってどれも魅力的。ワインのテイスティングのように楽しんでおります。

これはプログラミング言語と似たところがあるようにも思えます。どの言語にもクセというか個性があって（それがなければたくさんの言語がある意味がないもんね）、それぞれの長所は裏を返せば短所でもあるわけで。他人の（自分もだけど）評価とかいうものは客観的ってことはあり得ず、良し悪しじゃなく好き嫌いあるいは向き不向きでしか語ることはできないのかもと思うんですよね。だからどの言語が一番優れているか、どれがお勧めかなんてことを訊くのはナンセンス。誰だって自分の好きな言語、あるいは慣れた言語をお勧めするに決まっています。僕はずっとC++をやってきたから僕にとって一番慣れた言語はC++です。C#やJavaにも手を染めていますけど、どうしてもC++と比較することでしか判断できませんね。それでいいんじゃないかな。まずはひとつ、自分が得意だと胸を張れるほどに習得した言語ができれば、それとの違いという形で他の言語に手を伸ばせばいいのかな、と考えます。新しい年度が始まり、学校や会社で新人クン／新人サンが多数生まれています。彼ら／彼女らにお勧めできる言語と

して、C#はいいセンいってるんじゃないかと思います。ごめんなさい、.NET対応言語であればどれも同程度の表現力を持っているのだろうけど、僕はいきおいでC#を勧めてしまいます。理由は簡単、見た目が一番手慣れたC++に似ているから。

オブジェクト指向をもう少しおさらい

前回は「オブジェクト指向のおさらい」と題して、オブジェクト指向の3つの柱、

・データの抽象化
・継承
・多態

そしてオブジェクトの関係、

・is_A　→　AはB（の一種）である
・has_A　→　AはBを持つ
・use_A　→　AはBを利用する

さらに多重継承とインターフェイスについてざっくりと解説しました。オブジェクト指向とは、物事をまとめる仕組みのひとつです。単にまとめるだけじゃなく、効率的にきれいにエレガントにまとめる仕組みです。肝心なのは、言語が用意しているオブジェクト指向の仕組みはあくまで用意してくれているだけであって、その仕組みを活用するか否かはデザイナとプログラマの力量に掛かっているということです。オブジェクト指向を使えば開発効率が「上がるん」じゃありません。オブジェクト指向を使って開発効率を「上げる」んです。ココ重要。極端な話、3000行のコードを、

```
class Class1 {
   public static void Main() {
      ・
      ・ 延々と3000行
      ・
   }
}
```

みたいにたったひとつの関数Mainに「まとめる」ことだってできちゃうわけです。それをしないのは、複雑さが軽減されないから。「まとめる」とは、同時に“分割する”ということでもあるんですね。ある機能を実現するとき、それに必要な多数の断片に責任を持ついくつかのまとまり（断片群）に分割する、そのまとまりの単位がオブジェクト（またはクラス）です。そうやって構成されたまとまりのそれぞれが勝手に動き回ることはせず、なんらかの関係（is_A、has_A、use_A）で結び付けられています。そして複数のオブジェクトがお互いを利用し合いながら連携して目的の機能を実現させようというわけ。

事業規模が十分に小さければたったひとりで仕入れも製造も販売もこなすワンマン企業も可能でしょう。だけどやらねばならないことの種類も数も大量になると、とてもそれらをひとりでやってられない。規模が2倍になるとその複雑さは2倍以上になってしまうのです。だからこそ社員を雇い、それぞれに仕事を振り分けることによって各自のこなす仕事を簡単にしているのですから。そのとき各自は万能選手でなくていい、得意分野に精通していればいい。販売は仕入れ先の電話番号を知らなくて済む。トラブルに対する対処も楽になるでしょう。荷が届かなかったら配送、パッケージが破けていたら包装を疑えばいいのですから。

ただしこのようなきれいな形態を保つには、各自の責任をきちんと定義しておかなければなりません。配送担当が包装を受け持つようなことがあってはならないし、ましてや仕入れの一部を肩代わりしていてはトラブルに対して的確に対処できません。パッケージデザインの変更に伴う作業に配送担当が関与したくないし、たとえ関与するにしてもそれは極力小さく抑えておきたい。変更や拡張がおよぼす影響を極小化したいし局在化、つまり“小さな範囲をいじるだけで済む”ようにしたい。そうすることで変化に強くしましょう、と。そんな思いがオブジェクト指向に込められているわけです。

オブジェクト指向ではクラスを設計し、クラスの関係を設計します。企業が必要な各部門を定義し、部門間の関係を定めるのに似ています。小さな会社ならそんなにたくさんの部門を用意することはなく、かなり大まかに