リファクタリングであなたのソースコードを256倍美しくする

日本ユニシス株式会社
.NETビジネスディベロプメント
尾島 良司
OJIMA, Ryoji

特集 2

ダサい設計からバグのない読みやすく洗練された美しいコードへ

例をひとつ挙げるとするならば、そう「モテモテなプログラマ」。この例のようにどう頑張っても両立しないものは世の中に多々あるわけで、中学生時代にMSXを買ってしまった私の人生はこのどうにもならない何かに阻害されまくりだったりします。
でもしかし、同様に両立しないように見える「動作する綺麗な設計（のプログラム）」は、実は両立できるのです。いったいどうやって？　そう、本稿でご紹介する“リファクタリング”を適用するならば、動作する綺麗な設計のプログラムが可能になるのです。

Technology Tools

- ❑ Visual Basic .NET
- ☑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ☑ Other:
  NUnit V2.2

Level

★ ★ ★ ★ ☆

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、付録CD-ROMの¥DOTNET¥F02ディレクトリに収録しています。

¥SRC
リファクタリングのサンプル

リファクタリングとは？

リファクタリングとは、“ソフトウェアの外部的振る舞いを保ったままで、内部の構造を改善していく作業”を指します。

そう、設計して実装して終わりという考え方をやめて、大まかに設計して実装して、その後で設計を洗練していくと考えるわけです。

どうしてこんな回りくどいことをするのでしょうか？　その答は簡単。なぜなら、机上で完全な設計をするなんて人間には不可能だから。

つまり、どんなに頑張っても不完全な設計で実装するしかないのです。だから、完全な設計に近づくには実装の後にソースコードの修正という形で設計を洗練させていくしかないのです。

でも、実装後に闇雲にソースコードをいじっちゃうと、動かないソースコードができあがって終わりになってしまいます。これはヤバイ。このような事態を避けられる、容易に実行できる秩序だった手順が必要でしょう。

そう、リファクタリングはこの手順、すなわち短時間で終了する単純で秩序ある作業手順なのです。リファクタリングでは定められた手順を実行するだけなのですから、バグが入る余地はほとんどありません。このような手順（＝リファクタリング）があってはじめて、安心して後から設計を洗練できるわけです。

ね、リファクタリングってなんだか良さそうでしょ？

とにかくこの本を！

と、リファクタリングの定義を書いてきたわけですが、これだけ読んでもどうにもわからないですよね。そんな場合はコレ。Martin Fowlerの名著、『リファクタリング』（刊／ピアソンエデュケーション　ISBN：4-89471-228-8）です。

『リファクタリング』には、リファクタリングに必要なすべてが明快に書かれています。リファクタリングの例、リファクタリングの定義、リファクタリングが必要な兆候、リファクタリングのやり方が満載のリファクタリング

カタログなど、リファクタリングに関連するさまざまな技術解説や考察が、詳細かつわかりやすく載っているのです。

そう、『リファクタリング』は、プログラマであれば必ず読んでいなければならない本です。ぜひ購入し、リファクタリングを身に付けてやりましょう。

実際にリファクタリング

でも、そんな高価な本（なんと5,040円！）を、私ごときの推奨で買うのは勇気がいるかもしれません。

そんな貴方のために、本稿では実際にリファクタリングしてみることにします。リファクタリングがいかに便利なものか、肌で感じてください。

とりあえず作ったソースコード

リファクタリングなしで作ったソースコードは、動作はするけど汚いソースコードとなります。

リスト1を見てください。これはフレックスタイム制での時間外勤務の時間数を計算するプログラムです。たしかに動作しそうですけど、ちょっと汚すぎますよね？ とくにMonthlyAttendanceクラスのCalculateOvertime()メソッド。これはリファクタリングしてどうにかしないといけません。

リスト1：動作するだけのコード

```
public class MonthlyAttendance
{
   private ArrayList workPeriods = new ArrayList();

   // 重複するWorkPeriodを入れないでください。
   public IList WorkPeriods
   {
      get { return this.workPeriods; }
   }

   public TimeSpan CalculateOvertime()
   {
      TimeSpan overtime = TimeSpan.Zero;

      // 日単位で集計するためにソートします。
      this.workPeriods.Sort(new WorkPeriodComparer());

      for (int i = 0; i < this.workPeriods.Count; ++i)
      {
         WorkPeriod wp = (WorkPeriod)this.workPeriods[i];

         if (HolidayChecker.IsHoliday(wp.Date))
         {
            // 休日の勤務はすべて時間外。
            overtime += wp.TimeSpan;
         }
         else
         {
            // 平日の勤務は、昼休みと夕方休みを除外し、
            // コアタイムに勤務している場合は
            // 所定労働時間を引き算します。

            // 同じ日の出勤をまとめます。
            // あらかじめソートされているので、
            // 連続して比較すれば大丈夫！
            IList l1 = new ArrayList();
            for (; i < this.workPeriods.Count &&
               ((WorkPeriod)this.workPeriods[i]).Date == wp.Date; ++i)
            {
               l1.Add(this.workPeriods[i]);
            }

            // 昼休みを抜く。
            IList l2 = new ArrayList();
            foreach (WorkPeriod wp2 in l1)
            {
               if (wp.Date.AddHours(12) <= wp2.Start &&
                  wp2.End <= wp.Date.AddHours(13))
               {
                  ;
               }
               else if (wp2.End <= wp.Date.AddHours(12) ||
                  wp.Date.AddHours(13) <= wp2.Start)
               {
                  l2.Add(wp2);
               }
               else if (wp2.Start <= wp.Date.AddHours(12) &&
                  wp.Date.AddHours(13) <= wp2.End)
               {
                  l2.Add(new WorkPeriod(wp2.Start,
                     wp.Date.AddHours(12)));
                  l2.Add(new WorkPeriod(
                     wp.Date.AddHours(13), wp2.End));
               }
               else if (wp2.Start <= wp.Date.AddHours(12))
               {
                  l2.Add(new WorkPeriod(wp2.Start,
                     wp.Date.AddHours(12)));
               }
               else if (wp2.End >= wp.Date.AddHours(13))
               {
                  l2.Add(new WorkPeriod(
                     wp.Date.AddHours(13), wp2.End));
               }
            }

            // 夕方休みを抜く。
            IList l3 = new ArrayList();
            foreach (WorkPeriod wp2 in l2)
            {
               if (wp.Date.AddHours(17).AddMinutes(30) <= wp2.Start &&
                  wp2.End <= wp.Date.AddHours(18))
```