SKILL UP!

将来にわたって使えるスキルを身につけるために

# オブジェクト指向ではじめるプログラミング

日向 俊二
*HYUGA, Shunji*

最終回

## C#クラスの再利用

**Technology Tools**

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

**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¥CSHARPディレクトリに収録しています。

¥TXTFRM
テキスト整形プログラム

※この記事では.NET Framework SDKを利用して解説していますが、C#Builder、SharpDevelopなどでも利用できます(C#コンパイラならすべて利用できます)。

### はじめに

オブジェクト指向プログラミングの重要な目標のひとつは、コードを再利用できるようにすることです。今回はC#のクラスをゼロから作って、そのクラスをほかのクラスから利用するだけでなく、クラスを継承したり別のプログラムで使ったりと、さまざまな使い方をしてみましょう。

### プレーンテキストを読みやすく

仕様や解説、論文やレポートを読むことは、ソフトウェア技術者にとって欠かせないことのひとつです。しかし、ソフトウェアのデザインやプログラミング、デバッグなどに時間を取られてしまって、読んでおきたいものはもちろん、読むべきものさえ、なかなか読めないもの。そんなときには、読むべきドキュメントをPDAや超小型のノートパソコンに保存しておいて持ち歩き、待ち時間などに少しずつ読むという方法がオススメ。

特にPDAはそういう用途に向いていますが、保存できるファイルのサイズに限りがあるうえに、表示できる情報もパソコンの一般的なディスプレイに比べると少ないというところが困った点です。

しかし、仕様や論文のようなテキスト主体の文書をプレーンなテキストにしておけば、サイズも小さくなり、テキストエディタのような単純なソフトウェアで読むことができます。その際に、無駄な空白や改行、長い----------や==========などで区切られたセクションのような、その内容を読む上で不必要な部分を削除してコンパクトにすると、サイズはさらに小さくなり、PDAのような比較的狭い表示領域でも快適に読めるようになります。

そこで今回は、テキストファイルを読み込んで、テキストを読む上で不要な部分を削除する、テキスト整形ツールを作ってみましょう。

### あえてクラスを使う必要はない？

COLUMN

今回作成するようなテキスト整形ツールの歴史は古く、C言語の時代からたくさん書かれています。C言語やそのほかのオブジェクト指向でないプログラミング言語で書かれたプログラムは、当然のことながらオブジェクト指向ではない手続き型のプログラムです。つまり、テキストを処理するツールならば、あえてクラスを使わずに手続き型のプログラムとして作ることができるともいえそうです。

ただし、それには条件があります。その条件とは、処理の内容が比較的単純で、コードを再利用したり、拡張することを考えない、ということです。コードを再利用したり、拡張することをあらかじめ予定している場合は、やはりクラスを使ったほうがあとの作業がずっと容易になります。

なお、テキストの処理に独自のクラスを使わない、手続き型のプログラムに準じたコードの例は、付録CD-ROMに収録しているformater.csの末尾を参照してください。

## テキストデータを保存／管理する

最初に、処理するテキストデータを保存して管理する、「TextData」という名前のクラスを作りましょう（リスト1）。

このクラスの作り方はいろいろ考えられますが、ここでは最も単純な方法をとることにします。つまり、読み込んだテキストファイル全体をすべてひとつの文字列（string）変数に保存しておき、テキスト全体もしくは個々のテキスト行を必要に応じて取り出して扱えるようにします。

テキストを保存する変数は「text」という名前のprivate変数にして、これにアクセスするためのプロパティアクセサ（getとset）を作成します。また、コンストラクタを2種類と、GetNextLineメソッドを作成します。GetNextLineメソッドは、呼び出すごとに現在処理中の位置から次の行末文字までを取り出して返します。行末文字は、このクラスを利用するプログラムコードのどこかで定義されていて、引数EndOfLineで渡されるものとします。

なお、GetNextLineメソッドは「次の行を取り出す」ためのメソッドですが、最初に呼び出されたときはもちろん最初の行を取り出し、データの最後に達したら、ファイル終了マーク（定数EOFMarkで定義）文字列を返すことにします。

このようなテキスト整形プログラムであれば、プロパティアクセサ（getとset）と「次の行を取り出す」ためのメソッドGetNextLineがあれば、それで充分でしょう。

## テキストを整形する

このプログラムで行なう処理を、次のように決めます。

①行の先頭と行末の空白を削除する
②連続する記号を最大で21個までに制限して、あとは余分なものとみなして削除する

つまり、図1のようなテキストがあるときに、図2のような結果を生成するようにしましょう。

図1：もとのテキストファイルの内容

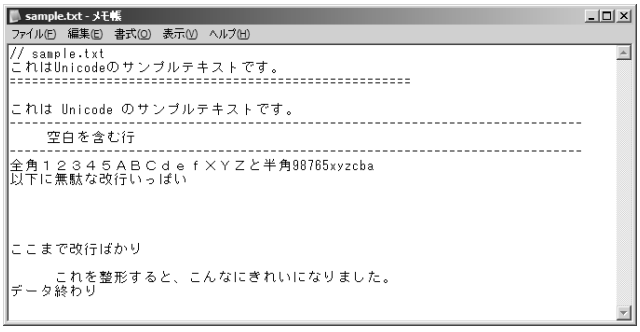


図2：整形したテキストファイルの内容