拡張しやすいJavaコードを書く技術

Acroquest Technology
石田 浩司

前回は、Javaの良いコード（イケてるコード）の例として「読みやすいコード」を書く技術を解説しました。良いコードは、読みやすいだけでなく、機能を「拡張」しやすいことも大事です。そこで今回は、良いコードの例として「拡張しやすい」コードについて説明します。例えば、「hello」と入力すると「world」と出力する単純なプログラムを考えてみてください。元々このプログラムは「hello」以外の文字列を与えても何も出力しません。しかし、使っているうちに飽きてきて、「How are you?」と入力すると「I'm fine.」と出力させたくなるかもしれません。それが、拡張です。

単純に考えると、入力された文字が「How are you?」かどうかを判断するif文を追加すればよさそうに思えます。このような単純なプログラムではそれでも問題がありませんが、プログラムが少し複雑になるといろいろと厄介な問題が出てきます。

ここでは、何らかのJavaプログラムのソースファイル（xxx.javaファイル）を読み込んで、そのステップ数やコメント行数を単にカウントする、というJavaアプリを考えてみます。このプログラム（以下、ラインカウンタと呼ぶ）*1は、元々はJavaプログラムのソースファイルにしか対応していませんが、これをJavaだけではなくHTMLやSQLのファイルもカウントできるように拡張してみます。

そもそも拡張性が高いとは、どのような意味を持つのでしょうか。筆者は、(1)拡張する仕組みがある、(2)拡張する際に修正個所の範囲が小さい、(3)ソースコードの重複がない、といったコードが拡張性のあるコードだと思っています。

ラインカウンタの場合、入力するファイルの行数をカウントする処理は、どの言語に関しても同じです。しかし言語によってコメントアウトの記号が異なるため、コメント行数をカウントするロジックに手直しが必要です。例えば、Javaでは「/* */、//」を検知してコメントと判断しています。これをHTMLのソースファイルにも対応するためには、HTMLのコメントである「<!-- -->」部分を検知するロジックが必要になります。それさえ変更すれば、HTMLのソースファイルをカウントする際にも使えそうです。同じように、SQLのコメントは「/* */、--」なので、変更次第ではSQLにも対応できそうです。

具体的に見てみましょう。**リスト1**は、ラインカウンタのソースです。ちなみに、このラインカウンタは本連載の第1回で「読みやすいコード」として作ったものです。詳しくは第1回をご覧ください。parselineというメソッドに、入力するテキストファイルの文字列を第2引数（line）に与えて実行すると、第1引数であるcountStatusという変数に、カウント結果が戻ってくるようになっています。このcountStatusは、ライン数やコメント数などをCountStatusという一つのクラスにまとめたものです。

このリスト1では、Javaコードを1文字ずつチェックしていきます。あらかじめコードやコメントが存在する場合のフラグを設定して、コードの場合はフラグを立てて、コメントならばcommentsに1、コード行ならstepsに1を加えます。コメントを判断する処理を変更すれば、HTMLをカウントする処理に流用できそうです。

まず、各言語に対応するため、入力ファイルの拡張子ごとにラインカウントの内容を切り替える処理を加えましょう。拡張子「fileExtension」をあらかじめ抽出し、判断するロジックを加えます。拡張子が.javaの場合は、**リスト2**のようにリスト1の先頭に拡張子を判断するif文を追加します。

HTMLの場合も拡張子ごとに処理をif文で分岐します。実装するに当たって必要なことは、(1)拡張子はそれぞれ「java」「html」「sql」に対応する、(2)それぞれのコメントの書き方を判断する、の2点です。とりわけ(2)のコメントの書式は、Javaのコメントが「/* */、//」、HTMLのコメントが「<!-- -->」、SQLのコメントが「/* */、--」と判断しな

*1 ラインカウンタのプログラムは、日経ソフトウエアのダウンロードコーナー（http://itpro.nikkeibp.co.jp/NSW/）で参照できます。

リスト1●前回作成したJavaコードのカウント部分

```
private void parseLine(CountStatus countStatus, String line) {
  boolean codeExisted = false;          ← コードがあるかをチェックするフラグ
  boolean commentExisted = false;       ← コメントをチェックするフラグ

  // 1文字ずつチェックする
  for (int index = 0; index < line.length(); index++) {
    char targetChar = line.charAt(index);
    String targetTwoChar = "";
    if (index + 1 < line.length()) {
      targetTwoChar = line.substring(index, index + 2);
    }                                   ← 2文字まとめて判断する

    if (targetChar == ' ' || targetChar == '¥t') {
      continue;
    }                                   ← 空文字またはタブの場合は処理を続ける

    if (countStatus.inCommentArea) {
      commentExisted = true;
      index = parseInComment(countStatus, index, targetTwoChar);
    } else if (countStatus.inSingleQuote) {
      codeExisted = true;
      index = parseInSingleQuote(countStatus, index, targetChar);
    } else if (countStatus.inDubleQuote) {
      codeExisted = true;
      index = parseInDoubleQuote(countStatus, index, targetChar);
    } else {                            ← シングルクオートなどはコードとして判定する
      if ("//".equals(targetTwoChar)) {
        commentExisted = true;
        break;
      }
      if ("/*".equals(targetTwoChar)) {
        commentExisted = true;
        index = parseInStartingComment(countStatus, index);
      } else {
        codeExisted = true;
        parseInNormalCode(countStatus, targetChar);
      }                                 ← //などがある場合はコメントとして判定する
    }
  }

  // この行が何の行であるかを判定する
  countStatus.lines++;                  ← 行をカウントする
  if (codeExisted) {
    countStatus.steps++;                ← コードがある場合はsteps（コード行）に1を加える
(1) } else if (commentExisted || countStatus.inCommentArea) {
    countStatus.comments++;             ← コメントの場合はcomments（コメント行）に1を加える
  } else {
    countStatus.empty++;                ← コードでもコメントでもない場合はempty（空行）に1を加える
  }
}
```

リスト2●先頭に分岐を追加して拡張子ごとに分けた

```
if ("java".equals(fileExtension)) {     ← 拡張子が.javaの場合に分岐する
  boolean codeExisted = false;
  boolean commentExisted = false;
  // 1文字ずつチェックする
  for (int index = 0; index < line.length(); index++) {
    char targetChar = line.charAt(index);
    String targetTwoChar = "";
    （以下、リスト1と同じ）
  }
}
```

リスト3●HTML文書にも対応するように追加した

```
else if ("html".equals(fileExtension)) {   ← 拡張子が.htmlか判断する
  boolean codeExisted = false;
  boolean commentExisted = false;

  // 1文字ずつチェックする
  for (int index = 0; index < line.length(); index++) {
    char targetChar = line.charAt(index);
    String targetFourChar = "";
    String targetThreeChar = "";        ← HTMLの場合はエリアコメントの開始と終了で文字数が異なるため、変数を二つ用意しておく
    if (index + 3 < line.length()) {
      targetFourChar = line.substring(index, index + 4);
    }
    if (index + 2 < line.length()) {
      targetThreeChar = line.substring(index, index + 3);
    }
    if (targetChar == ' ' || targetChar == '¥t') {
      continue;
    }

    // HTMLコードの場合はエリアコメントしか存在しない
    if (countStatus.inCommentArea) {
      commentExisted = true;
      codeExisted = false;
      index = parseInComment(countStatus, index,
        targetThreeChar);
      if (!countStatus.inCommentArea) {
        break;
      }
    } else {
      if ("<!--".equals(targetFourChar)) {
        commentExisted = true;
        codeExisted = false;
        index = parseInStartingComment(countStatus, index);
      } else {
        codeExisted = true;             ← htmlの場合はクオートの判断はいらない
      }
    }
  }
}
  （以下リスト1の（1）と同じ）
```