

NAME

Pod::InputObjects - objects representing POD input paragraphs, commands, etc.

SYNOPSIS

```
use Pod::InputObjects;
```

REQUIRES

perl5.004, Carp

EXPORTS

Nothing.

DESCRIPTION

NOTE: This module is considered legacy; modern Perl releases (5.18 and higher) are going to remove Pod-Parser from core and use Pod-Simple for all things POD.

This module defines some basic input objects used by [Pod::Parser](#) when reading and parsing POD text from an input source. The following objects are defined:

package **Pod::Paragraph** 4

An object corresponding to a paragraph of POD input text. It may be a plain paragraph, a verbatim paragraph, or a command paragraph (see `perlpod`).

package **Pod::InteriorSequence** 4

An object corresponding to an interior sequence command from the POD input text (see `perlpod`).

package **Pod::ParseTree** 4

An object corresponding to a tree of parsed POD text. Each “node” in a parse-tree (or *ptree*) is either a text-string or a reference to a **Pod::InteriorSequence** object. The nodes appear in the parse-tree in the order in which they were parsed from left-to-right.

Each of these input objects are described in further detail in the sections which follow.

Pod::Paragraph

An object representing a paragraph of POD input text. It has the following methods/attributes:

Pod::Paragraph->new()

```
my $pod_para1 = Pod::Paragraph->new(-text => $text);
my $pod_para2 = Pod::Paragraph->new(-name => $cmd,
    -text => $text);
my $pod_para3 = new Pod::Paragraph(-text => $text);
my $pod_para4 = new Pod::Paragraph(-name => $cmd,
    -text => $text);
my $pod_para5 = Pod::Paragraph->new(-name => $cmd,
    -text => $text,
    -file => $filename,
    -line => $line_number);
```

This is a class method that constructs a `Pod::Paragraph` object and returns a reference to the new paragraph object. It may be given one or two keyword arguments. The `-text` keyword indicates the corresponding text of the POD paragraph. The `-name` keyword indicates the name of the corresponding POD command, such as `head1` or `item` (it should *not* contain the `=` prefix); this is needed only if the POD paragraph corresponds to a command paragraph. The `-file` and `-line` keywords indicate the filename and line number corresponding to the beginning of the paragraph

\$pod_para->cmd_name()

```
my $para_cmd = $pod_para->cmd_name();
```

If this paragraph is a command paragraph, then this method will return the name of the command (*without* any leading `=` prefix).

```
$pod_para->text()
```

```
my $para_text = $pod_para->text();
```

This method will return the corresponding text of the paragraph.

```
$pod_para->raw_text()
```

```
my $raw_pod_para = $pod_para->raw_text();
```

This method will return the *raw* text of the POD paragraph, exactly as it appeared in the input.

```
$pod_para->cmd_prefix()
```

```
my $prefix = $pod_para->cmd_prefix();
```

If this paragraph is a command paragraph, then this method will return the prefix used to denote the command (which should be the string “=” or “==”).

```
$pod_para->cmd_separator()
```

```
my $separator = $pod_para->cmd_separator();
```

If this paragraph is a command paragraph, then this method will return the text used to separate the command name from the rest of the paragraph (if any).

```
$pod_para->parse_tree()
```

```
my $ptree = $pod_parser->parse_text( $pod_para->text() );
$pod_para->parse_tree( $ptree );
$ptree = $pod_para->parse_tree();
```

This method will get/set the corresponding parse-tree of the paragraph's text.

```
$pod_para->file_line()
```

```
my ($filename, $line_number) = $pod_para->file_line();
my $position = $pod_para->file_line();
```

Returns the current filename and line number for the paragraph object. If called in a list context, it returns a list of two elements: first the filename, then the line number. If called in a scalar context, it returns a string containing the filename, followed by a colon (:), followed by the line number.

Pod::InteriorSequence

An object representing a POD interior sequence command. It has the following methods/attributes:

```
Pod::InteriorSequence->new()
```

```
my $pod_seq1 = Pod::InteriorSequence->new(-name => $cmd
-lldelim => $delimiter);
my $pod_seq2 = new Pod::InteriorSequence(-name => $cmd,
-lldelim => $delimiter);
my $pod_seq3 = new Pod::InteriorSequence(-name => $cmd,
-lldelim => $delimiter,
-file => $filename,
-line => $line_number);
```

```
my $pod_seq4 = new Pod::InteriorSequence(-name => $cmd, $ptree);
my $pod_seq5 = new Pod::InteriorSequence($cmd, $ptree);
```

This is a class method that constructs a `Pod::InteriorSequence` object and returns a reference to the new interior sequence object. It should be given two keyword arguments. The `-lldelim` keyword indicates the corresponding left-delimiter of the interior sequence (e.g. '<'). The `-name` keyword indicates the name of the corresponding interior sequence command, such as `I` or `B` or `C`. The `-file` and `-line` keywords indicate the filename and line number corresponding to the beginning of the interior sequence. If the `$ptree` argument is given, it must be the last argument, and it must be either string, or else an array-ref suitable for passing to **Pod::ParseTree::new** (or it may be a reference to a `Pod::ParseTree` object).

```
$pod_seq->cmd_name()
    my $seq_cmd = $pod_seq->cmd_name();
```

The name of the interior sequence command.

```
$pod_seq->prepend()
    $pod_seq->prepend($text);
    $pod_seq1->prepend($pod_seq2);
```

Prepends the given string or parse-tree or sequence object to the parse-tree of this interior sequence.

```
$pod_seq->append()
    $pod_seq->append($text);
    $pod_seq1->append($pod_seq2);
```

Appends the given string or parse-tree or sequence object to the parse-tree of this interior sequence.

```
$pod_seq->nested()
    $outer_seq = $pod_seq->nested || print "not nested";
```

If this interior sequence is nested inside of another interior sequence, then the outer/parent sequence that contains it is returned. Otherwise undef is returned.

```
$pod_seq->raw_text()
    my $seq_raw_text = $pod_seq->raw_text();
```

This method will return the *raw* text of the POD interior sequence, exactly as it appeared in the input.

```
$pod_seq->left_delimiter()
    my $ldelim = $pod_seq->left_delimiter();
```

The leftmost delimiter beginning the argument text to the interior sequence (should be "<").

```
$pod_seq->right_delimiter()
    The rightmost delimiter beginning the argument text to the interior sequence (should be ">").
```

```
$pod_seq->parse_tree()
    my $ptree = $pod_parser->parse_text($paragraph_text);
    $pod_seq->parse_tree($ptree);
    $ptree = $pod_seq->parse_tree();
```

This method will get/set the corresponding parse-tree of the interior sequence's text.

```
$pod_seq->file_line()
    my ($filename, $line_number) = $pod_seq->file_line();
    my $position = $pod_seq->file_line();
```

Returns the current filename and line number for the interior sequence object. If called in a list context, it returns a list of two elements: first the filename, then the line number. If called in a scalar context, it returns a string containing the filename, followed by a colon (':'), followed by the line number.

Pod::InteriorSequence::DESTROY()

This method performs any necessary cleanup for the interior-sequence. If you override this method then it is **imperative** that you invoke the parent method from within your own method, otherwise *interior-sequence storage will not be reclaimed upon destruction!*

Pod::ParseTree

This object corresponds to a tree of parsed POD text. As POD text is scanned from left to right, it is parsed into an ordered list of text-strings and **Pod::InteriorSequence** objects (in order of appearance). A **Pod::ParseTree** object corresponds to this list of strings and sequences. Each interior sequence in the parse-tree may itself contain a parse-tree (since interior sequences may be nested).

Pod::ParseTree->new()

```
my $ptree1 = Pod::ParseTree->new;
my $ptree2 = new Pod::ParseTree;
my $ptree4 = Pod::ParseTree->new($array_ref);
my $ptree3 = new Pod::ParseTree($array_ref);
```

This is a class method that constructs a `Pod::ParseTree` object and returns a reference to the new parse-tree. If a single-argument is given, it must be a reference to an array, and is used to initialize the root (top) of the parse tree.

`$ptree->top()`

```
my $top_node = $ptree->top();
$ptree->top( $top_node );
$ptree->top( @children );
```

This method gets/sets the top node of the parse-tree. If no arguments are given, it returns the topmost node in the tree (the root), which is also a **Pod::ParseTree**. If it is given a single argument that is a reference, then the reference is assumed to a parse-tree and becomes the new top node. Otherwise, if arguments are given, they are treated as the new list of children for the top node.

`$ptree->children()`

This method gets/sets the children of the top node in the parse-tree. If no arguments are given, it returns the list (array) of children (each of which should be either a string or a **Pod::InteriorSequence**). Otherwise, if arguments are given, they are treated as the new list of children for the top node.

`$ptree->prepend()`

This method prepends the given text or parse-tree to the current parse-tree. If the first item on the parse-tree is text and the argument is also text, then the text is prepended to the first item (not added as a separate string). Otherwise the argument is added as a new string or parse-tree *before* the current one.

`$ptree->append()`

This method appends the given text or parse-tree to the current parse-tree. If the last item on the parse-tree is text and the argument is also text, then the text is appended to the last item (not added as a separate string). Otherwise the argument is added as a new string or parse-tree *after* the current one.

`$ptree->raw_text()`

```
my $ptree_raw_text = $ptree->raw_text();
```

This method will return the *raw* text of the POD parse-tree exactly as it appeared in the input.

Pod::ParseTree::DESTROY()

This method performs any necessary cleanup for the parse-tree. If you override this method then it is **imperative** that you invoke the parent method from within your own method, otherwise *parse-tree storage will not be reclaimed upon destruction!*

SEE ALSO

[Pod::InputObjects](#) is part of the [Pod::Parser](#) distribution.

See [Pod::Parser](#), [Pod::Select](#)

AUTHOR

Please report bugs using <http://rt.cpan.org>.

Brad Appleton <bradapp@enteract.com>