

10.3.1 Details of the simple text formatter

Description

In the section we describe the details of the simple text formatter.

The overall structure of class Document is the following:

```
class Document(temp: ref Template, lineMax: var integer):
  :::
  content: obj OrderedList(Paragraph)
  addParagraph(theContent: var String, theStyle: ref
Style):
    content.insert(Paragraph(theContent, theStyle))
  :::
```

A Document has the following attributes:

- A data-item content, which is an OrderedList of the Paragraphs in a given Document.
- A method addParagraph that inserts a new Paragraph with arguments theContent and theStyle into content.

Next we have added format methods to Document and Paragraph:

```
class Document(temp: ref Template, lineMax: var integer):
  class Paragraph(theContent: var String, theStyle: ref Style):
    format -> formattedPar: var
String:
    formattedPar := theStyle.format(theContent, lineMax)
  ...
  format -> formattedDoc: var String:
    formattedDoc := ""
    content.scan
      formattedDoc := formattedDoc + current.format
  print:
    print(format)
```

- The format method for Paragraph returns a String in formattedPar as computed by invoking theStyle.format(theContent, lineMax).
- The format method of Document returns a String in formattedDoc by adding the formatted strings for each Paragraph in the content (content) of the Document.
- Finally the print method prints the String returned by invoking the format method of the Document.

For class Template, we just add the data-item styles which is an OrderedList of possible styles that are defined for the Template – it is not used in the examples, and we leave it to the reader to make use of it. In addition we add an inner(Template).

```
class Template:
  styles: obj OrderedList(Style)
  heading: ref Style
  bodyText: ref Style
  inner(Template)
```

For class Style, we make explicit that generation of a Style object returns a reference to the newly generated object – this is not really needed since such a references returned by default. We also add inner(Style) and inner(format).

```
class Style -> S: ref Style:
  format(content: var String, lineMax: var
integer)
                                -> formattedString: var String:<
```

```

    inner(format)
    inner(Style)
    S := this(Style)

```

The following example describes the classes `DefaultHeading` and `DefaultBodyText`:

```

class DefaultHeading: Style
    secNo: var integer
    format::<
        secNo := secNo + 1
        formattedString := "\n" + secNo + ". " + content
class DefaultBodyText: Style
    format::<
        pos: var integer
        pos := 3
        formattedString := "\n  "
        for (1):to(content.length):repeat
            pos := pos + 1
            if (pos > lineMax):then
                formattedString := formattedString + "\n  "
                pos := 3
            formattedString := formattedString + content.get[inx]

```

A `DefaultHeading` has a local variable `secNo` keeping track of the heading numbers. The `format` method increments `secNo` by one and returns a `String` consisting of a newline ("`\n`"), followed by `secNo` followed by a dot and a blank ("`.` ") and the string hold in the `content` parameter of `format`.

The `format` method of class `DefaultBodyText` appends the characters in `content` to `formattedContent`. A newline and there blanks ("`\n` ") are inserted so each line has a maximum of `lineMax` characters.

We define a standard template as a subclass of `Template`:

```

class StandardTemplate: Template
    heading := DefaultHeading
    bodyText:= DefaultBodyText

```

A `StandardTemplate` object initializes the predefined styles `heading` and `bodyText` to `defaultHeading` and `defaultBodyText` respectively.

Finally we define class `StandardDocument`:

```

StandardDocument: Document
    addHeading(H: ref String):
        addParagraph(H, theTemplate.heading)
    addBodyText(B: ref String):
        addParagraph(B, theTemplate.bodyText)

```

Class `StandardDocument` is a subclass of `Document`. It adds two methods `addHeading` and `addBodyText`. Each invoke `addParagraph` with their `String` parameter and the appropriate style defined within `theTemplate`.

This completes describing the details of the simple text formatting system.