

## 10.3 A simple text formatter system

### Description

In this section, we present another example of using nested classes. The domain is text formatting as known from e.g. Microsoft Word, and LaTeX. The complexity of most of these systems is huge. Here we will thus stick to a very simple text formatting system.

The phenomena of the domain are documents that consists of a number of paragraphs. The paragraphs of a document are formatted according to a given template, which defines a number of styles for forming the paragraphs in the document. A style may e.g. define the font being used, the size of the font, the form of the font, etc. A paragraph may be left-aligned, centred, right-aligned, og with flushed margins.

In addition to styles, there are also properties associated with a document like page orientation (portrait or landscape), paper size (like A4, A3, and US letter), and number of columns.

To avoid ending up with a large example, we will keep the possible template and style properties to a minimum. Our text formatter can handle documents with two types of paragraphs: headings and body text. Headings are automatically numbered sequentially and body texts are indented by three spaces. A document has only one property which is the maximum length of a line in a paragraph.

Given a simple text formatter, represented by the class `SimpleTextFormatter`, it can be use as shown here.

```
myFormatter: obj SimpleTextFormatter
  doc: obj StandardDocument(StandardTemplate, 10)
  doc.addHeading("Introduction")
  doc.addBodyText("Once upon a time, ...")
  doc.addHeading("The problem")
  doc.addBodyText("There was a programmer, ...")
  doc.print
```

- The objects `myFormatter` is derived from `SimpleTextFormatter`.
- It defines an object `doc` which is an instance of class `StandardDocument`.
  - `StandardDocument` has two arguments: `StandardTemplate` and `10`.
  - `StandardTemplate` is an attribute of `SimpleTextFormatter` and defines the template to be used for `doc`.
  - The argument `10` sets the maximum line length to `10`.
- Then 4 paragraphs are added to `doc` by invoking `doc.addHeading` and `doc.addBodyText`.
  - `addHeading` uses a heading style and `addBodyText` uses a body style defined in `StandardTemplate`.
  - Each invocation has a `String`.
- Finally `doc` is printed giving the output:

```
1. Introduction
   Once up
   on a tim
   e, ...
2. The problem
   There w
   as a pro
   grammer,
   ...
```

A real text formatter should of course not insert newlines inside a word, but as mentioned, we keep it simple here.

Class `SimpleTextFormatter` has the following structure:

```
class SimpleTextFormatter:
  class Document:
    :::
  class Template:
```

```

:::
class Style:
  :::
class StandardTemplate: Template
  :::
class StandardDocument: Document
  :::

```

- It defines classes `Document`, `Template`, `Style`, `StandardTemplate`, and `StandardDocument`. `StandardTemplate` is a subclass of `Template`, and `StandardDocument` is a subclass of `Document`.

Class `Document` has the following structure

```

class Document(theTemplate: ref Template, lineMax: var integer):
  class Paragraph(theContent: var String, theStyle: ref Style):
    format -> formattedPar: var String:<
      :::
  addParagraph(theContent: var String, theStyle: ref Style):
    :::
  format -> formattedDoc: var String:
    ...
  print:
    :::

```

Class `Document` has the following attributes:

- Two parameters `theTemplate` and `lineMax`.
- A local class `Paragraph` representing a paragraph of the document.
- A method `addParagraph` for adding a `Paragraph` to the `Document`.
- A method `format` that returns a formatted version of the `Document`.
- A `print` method.

Class `Paragraph` has the following attributes:

- Two parameters `theContent` and `theStyle`.
- A virtual method `format` that returns a formatted version of the paragraph.

Class `Template` has the structure:

```

class Template:
  heading: ref Style
  bodyText: ref Style
  ...

```

It has two attributes `heading` and `bodyText` holding references to the styles for paragraphs being headings and paragraphs being body text.

Class `Style` has the structure:

```

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

```

It has a virtual method `format` for formatting the `String` being hold in `content` given that `lineMax` defines the maximum line length.

In the next section we show the details of the above classes and methods. The reader may skip this part during a first reading.