PA1458 Hemtenta Exempel II

Mikael Svahnberg

May 19, 2021

Contents

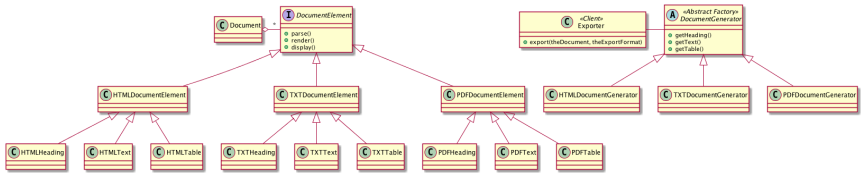
1 Systembeskrivning

En del av en ordbehandlare som exporterar text (heading, text, table) till olika format (html, txt, pdf). Abstract factory används skapa rätt textkomponenter för det exportformat vi vill använda.

2 Klassdiagram

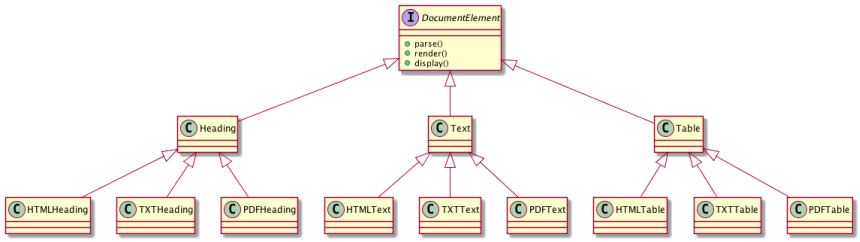
```
Document o- "*" DocumentElement

interface DocumentElement {
 +parse()
 +render()
 +display()
}

DocumentElement <|-- HTMLDocumentElement
DocumentElement <|-- TXTDocumentElement
DocumentElement <|-- PDFDocumentElement

HTMLDocumentElement <|-- HTMLHeading
HTMLDocumentElement <|-- HTMLText
HTMLDocumentElement <|-- HTMLTable

TXTDocumentElement <|-- TXTHeading
TXTDocumentElement <|-- TXTText
TXTDocumentElement <|-- TXTTable

PDFDocumentElement <|-- PDFHeading
PDFDocumentElement <|-- PDFText
PDFDocumentElement <|-- PDFTable

class Exporter <<Client>> {
 +export(theDocument, theExportFormat)
}

abstract class DocumentGenerator <<Abstract Factory>> {
 +getHeading()
 +getText()
 +getTable()
}

Exporter - DocumentGenerator

DocumentGenerator <|-- HTMLDocumentGenerator
DocumentGenerator <|-- TXTDocumentGenerator
```

```
DocumentGenerator <|-- PDFDocumentGenerator
```



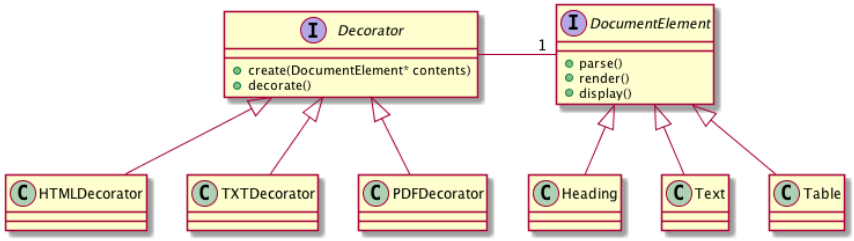
2.1 Ett altermativ till DocumentElement-arven Utvikning

```
interface DocumentElement {
 +parse()
 +render()
 +display()
}

DocumentElement <|-- Heading
DocumentElement <|-- Text
DocumentElement <|-- Table

Heading <|-- HTMLHeading
Heading <|-- TXTHeading
Heading <|-- PDFHeading

Text <|-- HTMLText
Text <|-- TXTText
Text <|-- PDFText

Table <|-- HTMLTable
Table <|-- TXTTable
Table <|-- PDFTable
```



2.2 Ett Alternativ till den alternativa utvikningenUber:Utvikning

```
interface DocumentElement {
 +parse()
 +render()
 +display()
}

DocumentElement <|-- Heading
DocumentElement <|-- Text
DocumentElement <|-- Table

interface Decorator {
 +create(DocumentElement* contents)
 +decorate()
}
Decorator <|-- HTMLDecorator
Decorator <|-- TXTDecorator
Decorator <|-- PDFDecorator

Decorator - "1" DocumentElement
```



3 Beskrivning av hur designmönstret används

När man anropar `Exporter::export()` så skapas en konkret fabrik. Sedan går man igenom det givna dokumentet och ber fabriken skapa objekt som representerar varje dokument-element man hittar. Fabriken skapar rätt objekt för varje dokument-element och enligt det exportformat som den representerar.

3.1 TODO Beskriv varje klass och dess ansvarsområden.

4 Pseudokod

```
Exporter::export(theDocument, theExportFormat) {
 DocumentGenerator* gen;
 switch (theExportFormat) {
  case "HTML" : gen = new HTMLDocumentGenerator(); break;
  case "TXT" : gen = new TXTDocumentGenerator(); break;
  case "PDF" : gen = new PDFDocumentGenerator(); break;
 }

 Document* output = new Document();

 theDocument->getElements()->forEach( function(e) {
   switch(e->getType()) {
    case Type.Heading: output->append(gen->getHeading()); break;
    case Type.Text: output->append(gen->getText()); break;
    case Type.Table: output->append(gen->getTable()); break;
   }
 });

 output->getElements()->forEach( render );
 return output;
}

HTMLDocumentGenerator::getHeading() {
 return new HTMLHeading();
}

HTMLHeading::render() {
 return "<H1>" + myText + "</H1>"
}
```

5 Designmönstrets användande av GRASP

- **GRASP1** Low Coupling
- **GRASP2** Controller

Genom att delegera till en konkret factory vilka objekt som faktiskt skapas så frigörs klienten från att behöva hålla koll på detta. Det blir alltså `low coupling` (eller i varje fall *lösare* coupling). «Client»-klassen blir controller; den gör inget direkt själv, utan delegerar till sitt factory-objekt att skapa rätt typ av objekt.

6 Systemets användande av GRASP

I systemet blir `Exporter` controller. Low Coupling åstadkoms genom att bara de konkreta DocumentGenerator-klasserna vet vilka objekt som skapas i DocumntElement-arvshierarkin.