|  |
| --- |
| Högskolan i Gävle |
| Miniprojekt 3 |
| Dokument |
|  |
| *Hanna Medén, Niklas Nordgren* |
|  |
| 2020-02-07 |

|  |
| --- |
| Miniprojekt 3 i kursen Objektorienterad design och programmering II, DVG503 |
|  |

**Innehållsförteckning**

[1 Inledning 1](#_Toc31564855)

[2 Metod 2](#_Toc31564856)

[3 Resultat 3](#_Toc31564857)

[4 Diskussion 4](#_Toc31564858)

[5 Bilaga – kodlistningar 5](#_Toc31564859)

[5.1 Klassen Element 5](#_Toc31564860)

[5.2 Klassen CompositeElement 6](#_Toc31564861)

[5.3 Klassen Header 6](#_Toc31564862)

[5.4 Klassen Paragraph 7](#_Toc31564863)

[5.5 Klassen ElementFactory 7](#_Toc31564864)

[5.6 Klassen Document 8](#_Toc31564865)

[5.7 Klassen Main 8](#_Toc31564866)

1. Inledning

I detta projekt skulle det byggas på funktionalitet på det tidigare projektet där vi skulle skapa ett backend system för en dokumenthanterare som kan skapa olika beståndsdelar av ett dokument. Nu skulle detta förbättras på så sätt att det även skulle finnas konverterare, som kan konvertera dokumentet och dess innehåll till exempelvis HTML, LATEX eller MarkDown. Det skulle även vara möjligt att göra ändringar på dokumentet och sedan även ångra dessa genom att lagra vilka ändringar som gjorts i dokumentet.

1. Metod

Inledningsvis implementerades designmönstret Command genom att skapa ett Command interface innehållande två metoder redo() och undo() som används för att göra ändringar i dokumentet och sedan även ångra ändringarna. Detta görs genom att systemet lagrar ändringar av dokumentet i två olika stackar, med namnen redoStack och undoStack, den ena sparar för ändringarna som utförs och den andra sparar de ändringar som ångrats.

Ett annat designmönster som introducerades i systemet var AbstractFactory, vilket gjordes genom att skapa två nya klasser som ärver från en abstrakt klass AbstractFactory. Namnet på dessa klasser är HTMLElementFactory och LATEXElementFactory som har i uppgift att skapa nya instanser av element.

För att konvertera texten i dokumentet mellan olika textrepresentationer användes designmönstret Visitor. Visitor delen av systemet består av två stycken interface, Visitor och Visitable där klasserna HTMLVisitor och LATEXVisitor implementerar Visitor interface:et och klassen Element implementerar interface:et Visitable. Anledningen till dessa implementationer var att varje Element skall kunna besökas av ett objekt av typen Visitor som i sin tur kan verka på objekt av typen Element utan att behöva göra runtime kontroller.

Vi gjorde även om klassen Element, där det lades till två attribut av typen String, startTag och endTag.

1. Resultat

Slutresultatet innehåller det system från Miniprojekt 2 där man skapar ett Document som agerar som en Facade och håller i en lista av alla element och kan även skapa de olika typer av element, header, paragraf etc. Det har sedan utvecklats så man kan skapa element i format av HTML eller LATEX direkt, man kan även skapa ett dokument med dess element som vanligt – utan format - och sedan, med hjälp av en Visitor kan man konvertera detta till det ena formatet och sedan det andra om man så önskar.

Det implementerades även ett MacroCommand vilket håller i en lista av Commands och möjliggör att göra flera command på ett och samma objekt och sedan kalla på redo() metoden endast en gång för att utföra alla i listan. Detta fungerar på samma sätt med undo()

1. Diskussion

Anledningen till att vi utökade klassen Element med två attribut av typen String var att ha dessa två som kommer runt brödtexten hellre än att sätta taggarna i brödtexten, då detta gör det lättare att hantera om man vill konvertera om ett redan konverterat dokument.

1. Bilaga – kodlistningar