|  |
| --- |
| Högskolan i Gävle |
| Miniprojekt 2 |
| OODP2 |
|  |
| Datum  2024-02-21 |

|  |
| --- |
| Kurs: Objektorienterad design och programmering 2 |
|  |
| Lärare: Goran Milutinovic |

**Innehållsförteckning**

[1 Inledning 1](#_Toc159438051)

[2 Metod 2](#_Toc159438052)

[3 Resultat 4](#_Toc159438053)

[4 Diskussion 7](#_Toc159438054)

1. Inledning

I vidareutvecklingen av miniprojekt1, som då kallas miniprojekt2, har ytligare mönster lagts till för att kunna ge användaren möjligheten att ändra och ångra sina tilläggningar av komponenter i dokumentet. Därtill har också konvertering till både Html och Markdown implementeras. Utöver detta har finjustering av vissa metoder och strukturen på klasser gjorts om en del för att göra det mer generiskt vid skapandet av delar i dokumentet.

1. Metod

Command-mönstret är den centrala delen av de nya förändringarna som gjorts, vilket ger oss möjligheten att utföra komplexa dokumentmanipulationer som att lägga till, redigera och ta bort komponenter i ett dokument.

*AddComponent* , *RemoveComponent* och *EditComponent*, kapslar in en specifik åtgärd och dess motsatta åtgärd, detta mönster tillåter inte bara en återgång av användares senaste handlingar utan också sparar en historik av ändringar som kan navigeras fram och tillbaka, se figur 1 och 2.

A screen shot of a computer screen

Description automatically generated

Figur 1, CommandManager

A screen shot of a computer program

Description automatically generated

Figur 2, AddCommand

Figurerna visar den överliggande klassen *DocumentCommandManager*, som delegerar arbetet till respektive underliggande klass.

För att stödja representationen av dokument i olika format, har systemet utrustats med konverteringsfunktioner som omvandlar dokumentkomponenterna till HTML och Markdown. Genom att använda *HTMLBuilder* och *MarkdownBuilder*, kan dokumentkomponenterna som tabeller och paragrafer representeras på ett korrekt sätt i båda formaten. Detta tillåter en flexibel presentation av innehåller, beroende på användarens behov, se figur 3. A screen shot of a computer code

Description automatically generated

Figur 3,HTML builder

För att förbättra systemets flexibilitet och underlätta tilllägg av nya komponenttyper har *addComponent* metoden i *DocumentFacade* gjorts mer generisk. Detta metod använder nu en fabrik som skapar komponenter baserat på en tippindikator och argument som tillhandahålls dynamiskt. Genom att separera komponentens skapande från dess användning, kan system lätt utvidgas med nya komponenter utan att behöva ändra i den centrala logiken.

1. Resultat

Vid körning av figur 4, får vi texten vi ser i figur 5. Detta visar då att funktionen som tar bort senaste handlingen från användaren fungerar.

A screen shot of a computer code

Description automatically generated

Figur 4, main körning

A black background with white text

Description automatically generated

Figur 5, resultat körning figur 4

Om vi därefter ångrar oss och vill återgå till de vi hade innan ändringar kan vi använda oss av *redoLastAction* funktionen, som då återställer dokumenttypen Table till som den var innan, se figur 6 och 7.

Det vi mer kan se på dessa är också att command som går igenom edit också fungerar då vi ändrat på paragrafen. A screen shot of text

Description automatically generated

Figur 6,Main klass för undo och edit

A screen shot of a computer

Description automatically generated

Figur 7,Resultat körning figur 6

För att översätta detta till Markdown alt HTML så behövs endast funktionen facade.buildHtmlDocument alt buildMarkdownDocument anropas med det dokument man vill översätta. Se figur 8, 9 och 10.

A screen shot of a computer code

Description automatically generated

Figur 8,Anrop till facade buildfuntion

A computer screen shot of white text

Description automatically generated

Figur 9,Resultat Html

A screenshot of a computer screen

Description automatically generated

Figur 10,Resultat Markdown

1. Diskussion

Det har varit otroligt svårt att få allt att fungera, det känns inte riktigt som att det finns något rätt eller fel, som gör det mer utmanande att förstå om man gjort ett bra eller dåligt jobb.   
Jag har tagit ett par tillbaka blickar på miniprojekt 1 och gjort några ändringar och flyttat lite logik så det känns lite bättre. Jag försöker hålla mig till att var sak har sin egen uppgift och har därav gjort fler factory metoder för att hålla ena till dokument och andra till komponenterna inuti dokumenten.   
Command-mönster i sig var nog det som var det svåraste att få till då det var svårt att få logiken att stämma med det jag hade gjort innan. Jag bestämde mig att allt bör köras genom Facade mönstret och tror att jag fått till det ganska bra. Utöver detta var det nog bara Markdown som behövde pluggas på då jag aldrig använt det tidigare men känner att jag gjort ett bra försökt till att implementera det fullt ut.

Jag har också problem att få till matriser och tabeller, just nu har jag en klass vardera men funderar på att ta bort den ena då dom i princip gör samma sak båda 2.  
Sist men inte minst så har jag försökt att implementera så att man kan lägga till och ta bort rader i tabeller/matriser med command, detta visade sig vara svårt då raderna är en del av en komponent i mitt system och inte en egen komponent. Då jag har funktioner som lägger till rader gjorde jag ett försök till att göra en egen klass addRowCommand som skulle ansvara för detta men fick desvärre inte till detta innan inlämningen som var lite tråkigt.