OGO Verslag

21-4-2024

Yanniek Wielage

2024

# Inleiding

Dit verslag heeft als doel uit te leggen welke aanpassingen er zijn gemaakt aan Jabberpoint en waarom er voor deze aanpassingen is gekozen. Dit wordt bereikt door eerst een geconstateerd probleem te formuleren, waarna de oplossing wordt beschreven. = Elk subhoofdstuk bevat een probleem en een bijbehorende oplossing.

Tijdens het refactoren is er rekening gehouden met de SOLID principes. Op basis van deze vijf principes zijn er aanpassingen gemaakt in de applicatie met als doel om de applicatie aan deze principes te laten voldoen.

De SOLID-principes klinken als volgenden:

**Single-Responsibility principe**

**Open-closed principe**

**Liskov Substitution principe**

**Interface Segregation principe**

**Dependency Inversion principe**

Wanneer de principes worden benoemd in de tekst, dan zijn deze gemarkeerd.

# Refactoring van de Jabberpoint applicatie

## De MenuController klasse

De ‘MenuController’ klasse gaat over de verschillende menu opties in de Jabberpoint applicatie. Deze klasse had een grote inefficiënte contructor en had vele verschillende verantwoordelijkheden. In de nieuwe applicatie zijn bepaalde verantwoordelijkheden verplaatst naar nieuwe klassen. Elke optie in het menu heeft nu een eigen klas die erft van een overhangende abstracte parent klasse. Deze klassen hebben een MenuItem die wordt gevuld met gegevens die relevant zijn voor deze menu optie, zoals een naam en een actie die wordt uitgevoerd wanneer hierop geklikt wordt. Daarnaast hebben deze klasses een MenuType, die aangeeft in welk menu de MenuItem thuis hoort. De nieuwe klassen erfen allen van de abstracte klasse en vervullen dezelfde functionaliteiten. De klassen hebben een enkele verantwoordelijkheid, namelijk het aanmaken van een MenuItem. Hierbij is rekening gehouden met het **Single-Responsibility principe.** De nieuwe klassen houden ook rekening met het **Open-closed principe**, omdat er van deze klassen kan worden geërfd als er een nieuwe MenuItem moet worden aangemaakt met vergelijkbare functionaliteit. De klassen houden rekening met het **Liskov Substitution principe**. Dit is getest door code vanuit de abstracte klasse direct in een child klasse te plaatsen. Volgende het **Liskov Substitution principe** zou het programma nog steeds moeten werken, wat dit programma met deze constructie doet. In de lijst in de ‘MenuController’ klasse worden objecten verwacht met als klasse ‘MenuOption’. Omdat MenuOption een abstracte klasse is kunnen child klassen ook worden toegevoegd aan deze lijst. Wanneer een nieuwe menu optie moet worden toegevoegd, kan er simpelweg een nieuwe klasse aangemaakt worden die erft van ‘MenuOption’. Deze kan dan worden toegevoegd aan de lijst met menu opties. Hierdoor voldoet het aan het **Dependency Inversion principe**.

# Style

De klasse ‘Style’ is een statische klasse binnen Jabberpoint. Deze klasse wordt op meerdere plekken gebruikt om de stijl aan te geven van een dia. In de ‘Jabberpoint’ klasse wordt de methode aangeroepen. Deze klasse meerdere verantwoordelijkheden, namelijk het inladen en het beheren van de gegevens over een stijl. Dit overtreedt het **Single-Responsibility principe**. Om dit op te lossen is er een nieuw klasse aangemaakt genaamd StyleInfo, die kan worden gebruikt om een nieuwe stijl toe te voegen.

# Accessor

De ‘Accessor’ klasse en de bijbehorende child klassen, ‘DemoPresentation’ en ‘XMLAccessor’ zijn een aantal aanpassingen ondergaan. Een aantal methodes en variabelen zijn verplaatst naar de correcte klassen, waaronder de saveFile() methode. Dankzij een efficïentere invulling van deze methode is het niet nodig om het te hernoemen in alle child classes. In plaats hiervan wordt het enkel benoemd in de parent klasse, ‘Accessor’. Er stonden ook meerdere variabelen in die bedoeld waren voor de ‘DemoPresentation’ klasse, deze staan nu in de juiste klasse. Bepaalde aanpassingen zijn ook gemaakt op de ‘SlideItem’ klasses (Inclusief de child klasses), zodat de klassen zich meer bezig houden met een enkele verantwoordelijkheid (**Single-Responsibility principe**).

# Bugs

Tijdens het refactoren van de ‘MenuController’ klasse is ook direct de bug opgelost die plaats vindt wanneer een getal groter dan het maximale aantal dia’s wordt ingevoerd bij de ‘Go To’ menu optie. Dit is opgelost door een if statement toe te voegen aan de ‘goToPage’ methode in de ‘GoToMenu’ klasse. Deze if statement controleert of het ingevoerde getal beschikbaar is in de lijst met dia’s. Er wordt gecontroleerd of het getal te groot is, of te klein!