Rapport OGO

**### Factory Class voor Accessor**

In Jabberpoint worden de readers en writers momenteel direct aangemaakt in de menu-opties en in de main-class. Dit zorgt ervoor dat de verantwoordelijkheid voor het aanmaken van deze objecten bij de main-class en de menucontroller ligt, wat niet ideaal is.

Er moet een factory-methode worden geïntroduceerd die de verantwoordelijkheid overneemt voor het creëren van de readers en writers (accessors). Hierdoor wordt de taak van het aanmaken van deze objecten losgekoppeld van de main-class en menucontroller.

**### Bugfix voor "Go To" Niet-bestaande Slides**

Wanneer je via View -> Go to een niet-bestaande slide selecteert, gebeurt er ogenschijnlijk niets. Echter, de slide index wordt wel bijgewerkt naar deze niet-bestaande slide, wat tot problemen kan leiden.

Ik heb een nieuwe methode geïmplementeerd die controleert of de opgegeven slide daadwerkelijk bestaat. Als dit het geval is, wordt de slide index bijgewerkt en de view ververst. Indien de slide niet bestaat, blijft de slide index ongewijzigd, maar de view wordt wel bijgewerkt om de gebruiker te informeren.

**### Control Verantwoordelijkheden in de Presentation Class**

De Presentation-class in Jabberpoint heeft momenteel te veel verantwoordelijkheden, wat de code moeilijk te onderhouden maakt. De methodes die de slides aansturen moeten worden opgesplitst.

Ik stel voor om de methodes die verantwoordelijk zijn voor het beheer van de slides te verplaatsen naar een nieuwe klasse, genaamd `SlideControl`. Hierdoor worden de verantwoordelijkheden duidelijk gescheiden en wordt de code beter onderhoudbaar.

**### Presentation Factory**

Op dit moment wordt de Presentation-class aangemaakt om Jabberpoint te starten, maar dit gebeurt via een lege presentatie die later met objecten wordt gevuld. Dit is niet de juiste aanpak.

Ik stel voor om een factory-methode te introduceren die verantwoordelijk is voor het volledig aanmaken van de presentatie, inclusief alle benodigde componenten. Deze methode voegt vervolgens deze componenten toe aan de presentatie, zodat deze direct compleet is.

**### MenuItem Factory**

In de menucontroller worden zowel de knoppen aangemaakt als hun acties gedefinieerd, wat resulteert in een klasse met meerdere verantwoordelijkheden.

Ik heb een `MenuItemFactory` geïntroduceerd die verantwoordelijk is voor het aanmaken van de menu-items. Hierdoor kan de controller zich richten op het controleren van de acties, zonder zich bezig te houden met het bouwen van de menu-items.

**### SetupWindow Methode**

Bij het opzetten van het venster worden zowel de presentatie als de SlideViewerComponent toegevoegd, wat resulteert in overbodige parameters.

Ik wil de `setupWindow`-methode aanpassen zodat deze alleen de presentatie als parameter accepteert en vervolgens zelf de benodigde objecten ophaalt. Daarnaast wil ik in de `keyPressed`-methode gebruik maken van `presentation.exit` in plaats van `System.exit`, voor een nettere afsluiting van het programma.

**### Verantwoordelijkheden in de XML Reader**

De XML Reader-class heeft momenteel drie verantwoordelijkheden: het lezen van het bestand, het aanmaken van slides, en het toevoegen van deze slides aan de presentatie. Dit is te veel voor één klasse.

Ik heb een factory geïntroduceerd die ervoor zorgt dat het juiste slide-item wordt aangemaakt. Deze factory bevat ook een methode die verantwoordelijk is voor het toevoegen van het item aan de slide, waardoor de verantwoordelijkheden van de XML Reader worden verminderd.

**### SlideItem Style**

In de `draw`-methode van de Slide-class wordt steeds opnieuw een style-object aangemaakt om dit te gebruiken tijdens het tekenen van de slide. Dit leidt tot onnodige belasting van de rekenkracht.

Ik heb de SlideItem-class voorzien van een style-attribuut, zodat deze uit het geheugen kan worden gehaald in plaats van elke keer opnieuw te worden aangemaakt. Daarnaast heb ik deze style toegevoegd aan de subklassen via `this.style`, zodat ze niet telkens als parameter hoeven te worden meegegeven.

**### Utility Class**

In veel klassen worden constante waarden zoals foutmeldingen en knoppen vaak in de klassen zelf gedefinieerd. Dit kan problemen opleveren als deze namen moeten worden aangepast.

Ik heb alle statische velden die vaak worden gebruikt verzameld in een utility package met daarin een utility class. Hierdoor hoeft een foutmelding of knopnaam slechts op één plek te worden aangepast, wat de code onderhoudbaarder maakt.