Steffan van der Werf

JABBERPOINT rEFACTOR

Contents

[Inleiding 3](#_Toc124418164)

[Dooie/ redundant code 4](#_Toc124418165)

[Probleem omschrijving 4](#_Toc124418166)

[Probleemoplossing 4](#_Toc124418167)

[Meerdere verantwoordelijkheden Accessor klass 4](#_Toc124418168)

[Probleem omschrijving 4](#_Toc124418169)

[Probleemoplossing 4](#_Toc124418170)

[Factory class accessor 5](#_Toc124418171)

[Probleem omschrijving 5](#_Toc124418172)

[Probleemoplossing 5](#_Toc124418173)

[Bug fix “go to” niet bestaande slides 5](#_Toc124418174)

[Probleem omschrijving 5](#_Toc124418175)

[Probleemoplossing 5](#_Toc124418176)

[Toekomst 5](#_Toc124418177)

[Presentation klasse control verantwoordelijkheden 5](#_Toc124418178)

[Probleem omschrijving 5](#_Toc124418179)

[Probleemoplossing 5](#_Toc124418180)

[Presentation Factory 5](#_Toc124418181)

[Probleem omschrijving 5](#_Toc124418182)

[Probleemoplossing 5](#_Toc124418183)

[Menuitem Factory 6](#_Toc124418184)

[Probleem omschrijving 6](#_Toc124418185)

[Probleemoplossing 6](#_Toc124418186)

[SetupWindow method 6](#_Toc124418187)

[Probleem omschrijving 6](#_Toc124418188)

[Probleemoplossing 6](#_Toc124418189)

[Een xml reader verantwoordelijkheden 6](#_Toc124418190)

[Probleem omschrijving 6](#_Toc124418191)

[Probleemoplossing 6](#_Toc124418192)

[Style constructor verantwoordelijkheid 6](#_Toc124418193)

[Probleem omschrijving 6](#_Toc124418194)

[Probleemoplossing 6](#_Toc124418195)

[SlideItem Style 7](#_Toc124418196)

[Probleem omschrijving 7](#_Toc124418197)

[Probleemoplossing 7](#_Toc124418198)

[Utility 7](#_Toc124418199)

[Probleem omschrijving 7](#_Toc124418200)

[Probleemoplossing 7](#_Toc124418201)

[MenuItem Factory 7](#_Toc124418202)

[Probleem omschrijving 7](#_Toc124418203)

[Probleemoplossing 7](#_Toc124418204)

# Inleiding

In dit document voor object georiënteerd ontwerpen (OGO) gaan we het Programma Jabberpoint refactoren. Dit houdt in dat we delen van code gaan optimaliseren door grote classes (bloaters) opnieuw in te delen in nieuwe classes waardoor grote classes minder verantwoordingen krijgen hierdoor is code bijvoorbeeld beter te hergebruiken en probeer je een betere logica te creëren.

Voor mijn refactor stratagie heb ik gekozen om eerst problemen te vinden deze 1 voor 1 op te lossen via try en error om vervolgens er een vernieuwd klassen diagram van te maken. Deze is te vinden in het mee geleverde asta bestand die bij dit document hoort.

# Dooie/ redundant code

## Probleem omschrijving

In Jabberpoint zitten methodes die niet gebruikt worden en daardoor beschrijven kunnen worden als dooie code. Ook wordt er gebruik gemaakt van casting waar dit niet nodig is waardoor de redundant code word genoemd dit gaan we ook opruimen en netter maken

## Probleemoplossing

Ik vind dat code die niet gebruikt worden verwijderd kunnen worden dit zorgt er namelijk voor dat er meer code is en daardoor het minder duidelijk is wat alles doet. Daarom gaan de volgende functies weg:

* Lege niet gebruikte constructor in BitmapItem
* Presetation getSlide methode wordt er bij de return gecast van Slide naar een Slide dit is niet nodig
* In de Presetation class is showlist al een array van slides hierdoor hoeft niet opnieuw worden aangegeven
* In Slide classe wordt er bij de constructor een slide item vector gemaakt terwijl dit al in de fields is aangegeven dit is ook redundant
* In Slide classe bij de methode getSlideItem wordt er gecast naar een Slide item terwijl dit al een vector van slideitems is dit is daardoor redundant
* In de draw methode van slide wordt er ook gecast naar slide item terwijl dit al een slide item teruggeeft
* 2e Constructor van SlideItem wordt niet gebruikt.
* 2e Constructor van TextItem wordt niet gebruikt
* Bij Presetation wordt de SlideViewerComponent in de fields op null gezet dit is redundant
* Bij de XML-reader is de loadSlideItem protected deze klasse wordt niet geextend en word alleen in de reader gebruikt en kan dus private worden

# Meerdere verantwoordelijkheden Accessor klass

## Probleem omschrijving

In Jabberpoint zit de klasse assessor deze klas is verantwoordelijk voor het schrijven en lezen van een file deze klasse heeft daardoor 2 verantwoordelijkheden heeft

## Probleemoplossing

Ik wil dit oplossen door deze klassen op te splitsen naar 2 klassen een reader en een writer klassen die deze verantwoordelijkheden hebben.

# Factory class accessor

## Probleem omschrijving

In jabber point worden de readers en writers in de menu opties en main al aangemaakt hierdoor is het aanmaken de verantwoordelijkheid van de main class en de menucontroller

## Probleemoplossing

Er moet een factory methode komen die de verantwoordelijkheid over neemt voor het maken van de assessors (writers en readers)

# Bug fix “go to” niet bestaande slides

## Probleem omschrijving

Op het moment dat je via View -> Go to een niet bestaand nummer in vult gebeurt er niks maar gaat het nummer van de slide index wel naar die slide toe.

## Probleemoplossing

Ik heb een nieuwe methode gemaakt die checkt of de slide die ingevuld is ook bestaat en returned dan true of false bij true updaten we de huidige slide index en daarna de view en bij false doen we de slide index niet updaten maar updaten we wel de view.

## Toekomst

Doordat ik veder ben gegaan met refacoteren is de oplossing die ik hiervoor geschreven heb alweer veranderd en veder geoptimaliseerd. Er wordt nu na een handeling een update view uitgevoerd in plaats bij de go to al.

# Presentation klasse control verantwoordelijkheden

## Probleem omschrijving

De presentatie klasse van jabberpoint heeft nogal veel verantwoordelijkheden dit wil ik gaan afsplitsen om te beginnen wil ik de slide methodes verplaatsen naar een eigen klasse

## Probleemoplossing

Ik wil de methodes die verantwoordelijk zin voor he besturen van de slides naar een slideControl klasse de methodes verhuizen.

# Presentation Factory

## Probleem omschrijving

De presentatie klasse wordt gemaakt om Jabberpoint te starten alleen op het moment wordt er eerst een lege presentatie met geen objecten en deze worden later toegevoegd. Dit hoort niet en moet anders.

## Probleemoplossing

Ik wil voor de presentatie een factory method maken die ervoor moeten gaan zorgen dat alle componenten worden gemaakt voor een presentatie en die vervolgens toevoegen aan de presentatie

# Menuitem Factory

## Probleem omschrijving

In de menu controller worden de acties van de knoppen bepaald en de knoppen aangemaakt hierdoor heeft de klassen 2 verantwoordelijkheden.

## Probleemoplossing

Ik heb een menuitem factory gemaakt die de verantwoordelijkheid heeft overgenomen van het maken van menu items zodat de controller puur controleert en niet bouwt.

# SetupWindow method

## Probleem omschrijving

Om het opzetten van de window wordt er de presentatie en de slideviewComponent toegevoegd die we net aan de presentatie hebben toegevoegd dit zorgt voor onnodige parameters

## Probleemoplossing

Ik wil de methode aanpassen dat je in die functie allee nog maar een presentatie hoeft mee te geven en dat die daarin alle objecten pakt die die nodig heeft in de functie. En gebruik functie in keypressed methode presentation.exit in plaats van de System.exit.

# Een xml reader verantwoordelijkheden

## Probleem omschrijving

De XML Reader heeft nu 3 verantwoordelijkheden het lezen van het bestand en de slides maken en het toevoegen aan de prestatie dit wil ik verminderen

## Probleemoplossing

Ik heb een factory gemaakt die ervoor zorgt dat er een juiste slide item aan gemaakt wordt en in de factory zit nog een methode die verantwoordelijk is voor het appenden van het item aan de slide.

# Style constructor verantwoordelijkheid

## Probleem omschrijving

De constructor van style is nu verantwoordelijk voor het aanmaken van een style object maar ook voor het maken van een font object.

## Probleemoplossing

Omdat een constructor van een style alleen een style hoort te initialiseren heb ik een factory gemaakt die de verantwoordelijkheid voor het font maken over neemt.

# SlideItem Style

## Probleem omschrijving

In de slide klasse in de methode draw wordt er een style lokaal in de methode elke keer aangemaakt om die weer tijdens het schrijven te gebruiken dit zorgt voor extra rekenkracht.

## Probleemoplossing

Ik heb een slideItem een style toegevoegd zodat die gewoon uit het geheugen kan worden gepakt in plaats van elke keer opnieuw aangemaakt moet worden in de methode zelf. Ook heb ik deze this.style bij de extends klassen toegevoegd zodat ze niet mee hoeven worden gegeven als parameter.

# Utility

## Probleem omschrijving

In veel klassen worden er finals gemaakt voor Errors knop knopen. Dit wordt allemaal in de klassen zelf geregeld dit kan leiden tot problemen als er namen moeten worden veranderd. Dit wil ik centraal hebben in een utility klassen

## Probleemoplossing

Ik heb alle static velden die vaker kunnen worden gebruikt in utility package met utility klassen zetten. Zodat als er een error moet worden veranderd dat maar op 1 plek hoeft worden gedaan en niet op meerdere plekken.

# MenuItem Factory

## Probleem omschrijving

In de menu controller klasse worden de acties van de knop aangeknoppeld en worden er menu items aan gemaakt dit zorgt voor 2 verantwoordelijkheden.

## Probleemoplossing

Ik heb een factory gemaakt die de verantwoordelijkheid over neemt voor het aanmaken van de menuitems.