Probleemanalyse

nhlstenden | EMMEN

OGO herkansing

Yanniek Wielage

2023

Inhoudsopgave

[Plan van aanpak 2](#_Toc131367369)

[Geconstateerde probleem 2](#_Toc131367370)

[Voorgestelde oplossing 3](#_Toc131367371)

# Plan van aanpak

Ik ben begonnen met het ontwikkelen van het klassendiagram. Dit diagram is gebaseerd op het programma dat al bestond. Dit bestand kan worden gevonden binnen het project als ‘ClassDiagramHerkansing’. Dit is een Draw.io bestand en kan worden geopend op de website met de corresponderende naam.

Op basis van dit klassendiagram is er geanalyseerd en besloten welk probleem er zal worden aangepakt binnen dit programma. Hierop is een refactoring gemaakt binnen het nieuwe klassendiagram genaamd ‘ClassDiagramHerkansingRefactored’ en zijn deze veranderingen uitgevoerd binnen de code, te vinden in het folder genaamd ‘Jabberpoint\_Sourcecode\_Refactored’.

# Geconstateerde probleem

Een klasse die er op eerste blik tussen uit stond, was de MenuController klasse. Deze klasse was onnodig groot. Deze klasse is te zien in figuur 1.

Table

Description automatically generated with low confidence

Figuur – MenuController

Alle functionaliteiten van het menu binnen het programma werden uitgevoerd binnen de constructor van deze klasse. Een klein deel van deze constructor kan worden gezien in figuur 2. De gehele constructor was echter wel 5 keer zo groot en erg chaotisch.

Text

Description automatically generated

Figuur - Constructor MenuController

# Voorgestelde oplossing

Om dit probleem op te lossen, is er gekeken naar bestaande principes en Design Patterns. Binnen het object-georiënteerd programmeren zijn de SOLID-principes een standaard. Voor deze klasse was voor de Single responsibility principe belangrijk. De MenuController klasse creëerde beide het menu en gaf de menu opties de functionaliteiten. Deze klasse heeft dus meer dan een verantwoordelijkheid. Dit mag niet volgens de SOLID-principes. Vandaar dat er gekozen is om deze klasse uit te werken naar meerdere klassen.

Er is gekozen de MenuController uit te werken naar meerdere klassen, gebruik maken van de Template Method Design Pattern. Hierdoor konden alle menu opties hun eigen klasse krijgen, maar was er geen reden om dubbele code te creëren. Ook wordt het makkelijker om de klassen te testen, aangezien alle menu opties binnen de nieuwe code zijn geïsoleerd.

De klasse MenuOption is de abstracte klasse die de benodigde parameters aanneemt van de MenuController klasse. Deze klasse is te zien in figuur 3.

Text

Description automatically generated

Figuur – MenuOption

De andere drie nieuwe klasses hebben een inheritance relatie met deze abstracte klasse en gebruiken de data die hier via de parameters zijn meegegeven.

Een ander voorbeeld van de verandering binnen de code kan worden gezien in FileMenu. Dit is een van de child classes. Binnen MenuController wordt alleen FileMenu aangemaakt. Echter, binnen de constructor van deze klasse worden de menu items aangemaakt en krijgen deze een functie. Een voorbeeld hiervan is te zien in figuur 4. Deze methode wordt aangeroepen in de constructor van FileMenu. Vergelijkbare methodes zijn te vinden in HelpMenu en ViewMenu, de andere twee nieuwe klassen.

Text

Description automatically generated

Figuur - Voorbeeld methode FileMenu

De relatie tussen MenuController en AboutBox is verplaatst en vindt nu te pas tussen HelpMenu en AboutBox.

Dit allen heeft er voor gezorgd dat de MenuController meer verzorgd en overzichtelijker is, zoals te zien in figuur 5.

Text

Description automatically generated

Figuur - Menucontroller na Refactoring