Jabberpoint verbeterd

Toets voor OGO1

Victor Peters
4985664

Inhoud

Inleiding

Gedurende de tweede periode van het tweede jaar van de opleiding Technische Informatica heb ik het vak Object Georiënteerd Ontwerpen (OGO1) gevolgd. Tijdens deze module heb ik gewerkt met een primitieve presentatietool genaamd JabberPoint. JabberPoint wordt gebruikt om presentaties in XML vorm in te lezen. Dit programma is echter niet modern en er zitten ontwerpkeuzes in die onlogisch zijn. Mijn opdracht deze periode was om JabberPoint te verbeteren. In dit document worden deze verbeteringen gedocumenteerd.

Refactor-strategie

Gedurende het tweede deel van Jabberpoint ga ik het programma refactoren en waar nodig verbeteren.

Ik ga dit doen door eerst een klasse te selecteren die ik wil verbeteren.

Vervolgens ga ik eerst kijken wat deze klasse moet doen en hoe deze klasse dit doet.

Dan ga ik kijken hoe ik deze klasse zou kunnen verbeteren (Moet er een aparte klasse voor de fields komen? Moeten er klassen komen die bepaalde taken van de klasse overnemen? Is de klasse überhaupt nodig? Is het beter om een factory klasse te maken om objecten te maken binnen de klasse?).

Dan ga ik Jabberpoint opstarten om te kijken wat het doet voor de verbetering.

Vervolgens ga ik een functie selecteren waarvan ik denk dat deze verbeterd moet worden.

Ik ga dan kijken naar hoe deze verbeterd kan worden (Moet ik de functie opdelen in meerdere functies? moet ik een parameter weghalen omdat deze eigenlijk onnodig is? Is de functie bugged en moet deze gefixed worden?).

Als ik weet hoe de functie verbeterd kan worden, dan ga ik deze verbetering uitvoeren.

Als ik klaar ben met de code schrijven voor de verbetering, dan start ik Jabberpoint weer op om te kijken of het programma nog werkt. Als het programma niet meer werkt, dan ga ik in de code kijken hoe ik dit kan oplossen. Als ik geen oplossing kan vinden, dan revert ik de verbetering.

Als het wel werkt, dan ga ik kijken of er een merkbaar verschil in snelheid is vergeleken met de oude implementatie van de functie. Als dit niet zo is, dan is het een succesvolle verbetering.

Verbetering 1: Het navigeren naar een specifieke slide

Binnen JabberPoint bestaat de functie om te navigeren naar een specifieke slide. Voordat ik deze functie verbeterd had kon de gebruiker navigeren naar een niet bestaande slide (slide 999 van 4). Ik heb deze functie verbeterd door aan de functie een extra controle toe te voegen die kijkt of het ingevoerde nummer kleiner of gelijk aan de hoeveelheid slides is en of het ingevoerde nummer groter is dan 0. Hierdoor blijft de gebruiker op dezelfde slide als er een enorm groot getal of een getal lager dan 1 ingevoerd wordt.

Verbetering 2: Het opslaan van een bestand.

Binnen JabberPoint bestaat de functie om een bestand op te slaan. Voordat ik deze functie aangepast had was dit een enorme functie (30+ regels). Na het verbeteren van deze functie is deze functie in totaal 14 regels. De verbeteringen die ik heb gemaakt zijn het vervangen van de originele for loop door een for each loop. Dit is naar mijn mening meer leesbaar en het bespaart een operatie. Ook is het opslaan van items in een slide nu een functie in de slide klasse. Dit heb ik gedaan omdat dit niet hoorde bij de accessor maar bij de slide zelf. Ook heb ik aan de slideitems een functie printItem toegevoegd. Dit is een abstracte functie binnen SlideItem. Dit heb ik gedaan zodat je niet in de functie hoeft te controleren of het slideitem een bitmapitem of een textitem is, en als er in een later stadium andere items toegevoegd worden aan JabberPoint, dan kunnen deze SlideItem extenden, en dan hoeft er geen aanpassing aan de saveFile functie gemaakt worden.

Verbetering 3: Het verplaatsen van de namen van MenuItems naar een nieuwe klasse.

Voor deze verbetering stonden de namen van alle menu items in de MenuController klasse. Dit zorgde ervoor dat deze klasse erg veel constanten had. Ik heb deze constanten naar een nieuwe klasse genaamd JabberPointMenuItems verplaatst zodat de MenuController klasse makkelijker te begrijpen is en zodat de klasse minder groot is.

Verbetering 4: Het maken van nieuwe klassen voor het creëren van het menu.

Voor deze verbetering werd het creëren van het menu van Jabberpoint in de constructor van MenuController gedaan. Dit had als gevolg dat de constructor van de MenuController erg groot was en dat als er een onderdeel van het menu aangepast zou moeten worden dit in deze klasse zou moeten gebeuren. Dit maakt refactoren lastiger dan wanneer het opgedeeld zou worden in verschillende klassen en functies. Ik heb een nieuwe abstracte klasse gemaakt genaamd MenuBuilder. Deze klasse wordt geëxtend door drie klassen: FileMenuBuilder, HelpMenuBuilder en ViewMenuBuilder. Deze klassen worden aangeroepen in de MenuController klasse en bouwen de file, help en view onderdelen van het JabberPoint menu.

Verbetering 5: Het maken van verschillende factories

JabberPoint bevatte voor deze verbetering geen factories. De klassen die gebruik moesten maken van een object, bijvoorbeeld wanneer de main klasse JabberPoint een nieuw Presentation object nodig heeft, moesten dit object ook aanmaken. Dit had als gevolg dat de applicatie minder makkelijk was om uit te breiden. Als ik bijvoorbeeld de Presentation klasse zou willen aanpassen, dan zou ik hoogstwaarschijnlijk ook de Main JabberPoint klasse aan moeten passen. Het maken van factories lost dit probleem op, omdat er nu een klasse is die het object gebruikt en een klasse is die het object maakt (de factory). Het maken van factories zorgt er ook voor dat wanneer ik bijvoorbeeld een nieuwe soort presentatie wil toevoegen aan de applicatie er geen problemen zullen optreden in de bestaande code, omdat het maken van de objecten gebeurt in aparte factory klassen.

Ik heb de volgende factory klassen gemaakt:

AccessorFactory, die verantwoordelijk is voor het maken van Accessor objecten.

MenuBuilderFactory, die verantwoordelijk is voor het maken van MenuBuilder objecten.

PresentationFactory, die verantwoordelijk is voor het maken van Presentation objecten.

SlideItemFactory, die verantwoordelijk is voor het maken van SlideItem objecten.

StyleFactory, die verantwoordelijk is voor het maken van Style objecten.

Verbetering 6: DemoPresentation een kind klasse van Presentation in plaats van Accessor maken.

Voor deze verbetering was de klasse DemoPresentation een kindklasse van de Accessor klasse. Dit vond ik erg onlogisch omdat de DemoPresentation geen Accessor is, maar een ingebouwde demo presentatie, en de presentatie is niet verantwoordelijk voor het lezen van en het schrijven naar een presentatie. Hier is de accessor verantwoordelijk voor. Ik heb de applicatie zo aangepast dat de DemoPresentation klasse de Presentation klasse extend omdat de demo presentatie een presentatie is en zodat er, door middel van een factory klasse die Presentation objecten maakt, een losse koppeling kan komen tussen de main JabberPoint klasse en de Presentation klasse.

Verbetering 7: Het organiseren van de klassen binnen JabberPoint.

Voor deze verbetering zaten alle JabberPoint klassen in dezelfde directory, namelijk de main directory ‘JabberpointSourceCode’. Ik een hoofd package gemaakt genaamd ‘com’, en ik heb hierin de volgende packages gemaakt: about, accessors, controllers, enums, factories, main, menubuilders, presentations, slide, slideitems en slideviewer. Deze verbetering heeft ervoor gezorgd dat de structuur van de broncode van JabberPoint een stuk overzichtelijker is. Je weet bijvoorbeeld gelijk waar je de klassen kunt vinden die iets te maken hebben met de Accessor klasse. Deze kun je vinden in de package ‘accessors’.

Verbetering 8: Het verwijderen van de style creatie uit de Style klasse.

Voor deze verbetering werd de creatie van verschillende styles gedaan in de style klasse zelf. Dit vond ik erg onlogisch omdat de objecten van een klasse nu in de klasse zelf gemaakt worden. In de huidige JabberPoint versie worden de styles gemaakt in de StyleFactory.

Verbetering 9: Een abstracte MenuBuilder klasse maken

In een eerdere verbetering heb ik verschillende klassen gemaakt die verantwoordelijk waren voor de verschillende onderdelen van het menu van JabberPoint (FileMenuBuilder, HelpMenuBuilder en ViewMenuBuilder). De objecten hiervan werden gemaakt in de MenuController klasse. Dit zorgde voor een strakke koppeling tussen de MenuController en de menubuilders. Ik heb dus een abstracte MenuBuilder klasse gemaakt die geëxtend wordt door de menubuilders, zodat ik een factory klasse kon maken die de objecten van de menubuilders maakt. Dit heeft als gevolg dat er nu een losse koppeling is tussen MenuController en de menubuilders, waardoor er een minder grote afhankelijkheid is tussen MenuController en de menubuilders.

Verbetering 10: Het verwijderen van de lege constructor uit Accessor

Voor deze verbetering zat er een lege constructor in de Accessor klasse. Ik heb deze constructor weggehaald omdat deze nergens gebruikt werd.

Verbetering 11: Het maken van een style field in SlideItem.

Voor deze verbetering werd de style van een slideitem niet bijgehouden in de slideitem, terwijl deze wel een style heeft. Dit had als gevolg dat de functies in een slideitem, zoals draw(), getBoundingBox() en de functie getLayouts in TextItem, die gebruikt maakten van de style van de slideitem een extra parameter voor de style moesten hebben. Ik heb dit opgelost door een style field van type Style toe te voegen aan de SlideItem klasse. Er wordt in de constructor een Style gemaakt in de StyleFactory, en deze wordt gebruikt in de draw() en getBoundingBox() functies. Door deze verandering is er een losse koppeling tussen de SlideItem en de Style klasse, en de functies die van een style gebruik maken hebben nu een parameter minder.

Verbetering 12: Het verwijderen van de constructor zonder parameters uit BitmapItem.

Voor deze verbetering was er een tweede constructor in BitmapItem die geen parameters had. Ik heb deze constructor verwijderd omdat deze verder niet gebruikt werd binnen de code.

Verbetering 13: De logica voor het maken van slides voor de DemoPresentation verplaatst naar de constructor van DemoPresentation.

Voor deze verbetering was er een functie die de slides van de demo presentatie maakte in de DemoPresentation klasse, namelijk de loadFile functie. Deze had als parameters een Presentation presentation en een String unusedFilename. Deze functie moest in DemoPresentation geïmplementeerd was, omdat DemoPresentation de Accessor klasse extende, en in Accessor was loadFile een abstracte methode. Ik heb deze functie verwijderd en de logica naar de constructor verplaatst, omdat de functie niet meer verplicht geïmplementeerd hoeft te worden. DemoPresentation extend niet meer van Accessor maar van Presentation. Ook heb ik dit gedaan omdat het enige wat deze klasse moet doen is de demo presentatie weergeven, dus als ik een object van DemoPresentation maak, dan wil ik deze gebruiken voor het weergeven van de demo presentatie. Door de logica dan in de constructor te zetten, hoef ik niet in de code ook nog de loadFile functie aan te roepen, die toch altijd de demo presentatie laadt.

Verbetering 14: De saveFile functie verwijderd uit de DemoPresentation.

Voor deze verbetering was er een functie in DemoPresentation die een Exception zou moeten gooien wanneer je de demo presentatie op probeert te slaan. Ik heb deze functie verwijderd omdat deze niks toevoegde aan het programma en alleen maar was toegevoegd aan DemoPresentation omdat deze eerst Accessor extende, en saveFile is een abstracte functie in Accessor.

Verbetering 15: Het maken van Presentation en Accessor objecten in de main JabberPoint klasse gedaan door middel van factory klassen.

Voor deze verbetering werden de Presentation en Accessor objecten in de main JabberPoint klasse gemaakt en gebruikt door de main JabberPoint klasse. Dit had als gevolg dat er een strakke koppeling was tussen de JabberPoint klasse en de Presentation en Accessor klassen. Hierdoor was de main klasse dus afhankelijk van deze klassen. Nu gebeurt de Presentation en Accessor object creatie in de PresentationFactory en de AccessorFactory, en in de main JabberPoint klasse worden de createPresentation en de buildAccessor functies aangeroepen die een Presentation en Accessor object returnen. Hierdoor is de JabberPoint klasse niet meer verantwoordelijk voor de creatie van deze objecten, en is er ook geen strakke koppeling meer maar een losse koppeling.

Verbetering 16: De switch-case structuur in KeyController verbeterd.

Ik heb de switch-case structuur in KeyController omgezet naar een verbeterde en kortere switch-case structuur.

Originele structuur:

```
switch(keyEvent.getKeyCode()) {
        case KeyEvent.VK_PAGE_DOWN:
        case KeyEvent.VK_DOWN:
        case KeyEvent.VK_ENTER:
        case '+':
                presentation.nextSlide();
                break;
```

Verbeterde structuur:

```
switch (keyEvent.getKeyCode())
{
        case KeyEvent.VK_PAGE_DOWN, KeyEvent.VK_DOWN, KeyEvent.VK_ENTER, '+' ->
presentation.nextSlide();
        case KeyEvent.VK_PAGE_UP, KeyEvent.VK_UP, '-' -> presentation.prevSlide();
        case 'q', 'Q' -> System.exit(0);
}
```

Ik heb ook de default case uit de switch-case structuur verwijderd, omdat deze geen meerwaarde bood. Deze case moest niet eens bereikt worden.

Verbetering 17: Enums gemaakt die fungeren als parameters van de bijbehorende factory klassen.

Ik heb enums gemaakt die gebruikt worden in de AccessorFactory, PresentationFactory en de MenuBuilderFactory. Deze enums zijn er om aan te geven wat voor object de factory klasse moet maken.
De enum voor de AccessorFactory heet AccessorType. Deze heeft een veld, namelijk XML. Dit is omdat er een soort accessor is binnen de huidige versie van JabberPoint. Ondanks dit heb ik er toch voor gekozen om de abstracte klasse Accessor te houden, zodat deze kan dienen als een blauwdruk voor als iemand in de toekomst een accessor klasse voor bijvoorbeeld JSON wilt toevoegen.
De enum voor de MenuBuilderFactory heet MenuBuilderType. Deze heeft drie velden, namelijk FILE, HELP en VIEW. FILE is voor een FileMenuBuilder object, MENU is voor een HelpMenuBuilder object en VIEW is voor een ViewMenuBuilder object.
De enum voor de PresentationFactory heet PresentationType. Deze heeft twee velden, namelijk DEMO en NORMAL. DEMO is voor een DemoPresentation object en NORMAL is voor een Presentation object.
De enum voor de SlideItemFactory heet SlideItemType. Deze heeft twee velden, namelijk BITMAP en TEXT. BITMAP is voor een BitmapItem en TEXT is voor een TextItem.

De enums zorgen ervoor dat degene die gebruik maakt van de factories een gelimiteerd aantal keuzes heeft voor objecten die overeenkomt met het aantal mogelijke objecten.

Verbetering 18: Het verwijderen van de constructor in Presentation met de parameter SlideViewerComponent.

Ik heb deze constructor verwijderd uit de Presentation klasse omdat dit ongebruikte code was.

Verbetering 19: De naam van de field showList veranderd naar slides.

Ik heb de naam van de field showList veranderd naar slides omdat dit een ArrayList van Slide objecten is, waardoor slides dus passender is als naam.

Verbetering 20: De functies getSlideItem en getSlideItems verwijdert uit de Slide klasse.

Ik heb de functies getSlideItem en getSlideItems verwijderd uit de Slide klasse. De functie getSlideItem werd vanaf het begin al nergens gebruikt, en de functie getSlideItems was niet meer nodig vanwege de veranderingen die in verbetering 21 aangebracht zijn.

Verbetering 21: De operaties die gebruik maakten van een Style aangepast in de draw functie in de Slide klasse, en de for-loop aangepast.

Voor deze verbetering moest er in de draw functie in de Slide klasse een Style object gemaakt worden voor het weergeven van de SlideItems. Dit was omdat de draw functie in de SlideItem klasse een Style als parameter nodig had, en de Slide klasse had geen field waarin een Style bijgehouden werd. Er moest dus telkens een Style object voor een SlideItem gemaakt worden op basis van het niveau van de SlideItems. Het maken van een Style object hoeft nu niet meer omdat de style van een SlideItem nu bijgehouden wordt in een field in elke SlideItems. Deze style wordt gecreëerd door de StyleFactory klasse op basis van het niveau van de SlideItem. Het gevolg hiervan is dat de draw functie in SlideItem geen Style parameter meer nodig heeft, omdat deze functie gewoon de field met het Style object kan gebruiken.

Ik heb de for-loop ook omgezet naar een foreach loop die loopt over alle SlideItems in een Slide. Dit heeft ervoor gezorgd dat er minder code in de loop en dus ook de functie staat.

Verbetering 22: De ongebruikte constructor uit de SlideItem klasse verwijderd.

Ik heb de constructor in SlideItem zonder parameters verwijderd omdat deze niet gebruikt werd.

Verbetering 23: Het verplaatsen van de eigenschappen van een Slide van SlideViewerComponent naar een nieuwe klasse.

Ik heb de eigenschappen van een Slide in SlideViewerComponent verplaatst naar een nieuwe klasse genaamd SlideProperties. Dit heb ik gedaan zodat de SlideViewerComponent klasse wat minder groot zou worden en zodat deze klasse begrijpelijker zou worden.

Verbetering 24: Het verwijderen van de constructor zonder parameters uit de TextItem klasse.

Ik heb de constructor in TextItem verwijderd die was voor het maken van een lege TextItem, omdat deze niet gebruikt werd binnen JabberPoint.

Verbetering 25: De creatie van een BitmapItem in DemoPresentation laten doen door een factory klasse.

Voor deze verbetering werd er in de laatste slide van de demo presentatie in DemoPresentation een BitmapItem gemaakt. Dit had als gevolg dat er een strakke koppeling was tussen DemoPresentation en BitmapItem, waardoor DemoPresentation afhankelijk werd van de BitmapItem klasse. Nu wordt de creatie van een BitmapItem gedaan door de SlideItemFactory klasse, waardoor er nu een losse koppeling is tussen DemoPresentation en BitmapItem. Hierdoor is er minder afhankelijkheid van DemoPresentation op de BitmapItem klasse.

Verbetering 26: Het verplaatsen van de namen van gebruikte XML-tags naar een nieuwe klasse vanuit XMLAccessor.

Ik heb de namen van de XML-tags verplaatst naar een nieuwe klasse genaamd XMLTagNames. Dit heb ik gedaan zodat de XMLAccessor klasse minder groot zou zijn en begrijpelijker zou worden.