Studentgegevens

Cursuscode IM0102

Jabberpoint Inhoudsopgave

Naam Daniel S. C. Schiavini

Studentnummer 851102873

Aanpak

De oorspronkelijke bedoeling was om deze opdracht uit te voeren in een team met een andere student. Echter, er was geen andere student beschikbaar met een vergelijkbare tempo. Daardoor heb ik deze opdracht alleen uitgevoerd.

Voordat ik wist dat ik de opdracht alleen mag uitvoeren, heb ik een Git repository gemaakt. Ik heb de repository zo geconfigureerd dat dit document automatisch wordt gebouwd door het gebruik van TravisCI. Zodra een nieuwe versie van het rapport naar GitHub wordt gepushed, draait een script dat het PDF-bestand maakt. Het script bouwt ook Jabberpoint, draait unit tests, en als alles gelukt is wordt ook een JAR-bestand gemaakt. In de master branch hangt een deployment vast dat het PDF en JAR naar GitHub pages publiceert.

Jabberpoint had nog geen geautomatiseerde tests ingebouwd, en omdat ik dat belangrijk vind, heb ik van tevoren JUnit tests geschreven. Echter niet alle klassen waren testbaar - TravisCI draait in een Linux-container waarin de Graphic Java-klassen niet gecreëerd kunnen worden. Ik had niet genoeg tijd had om dit probleem op te lossen, daarom heb ik alleen tests geschreven voor de domein- en accessor-klassen.

1 Probleemanalyse

Om het probleem te analyseren kunnen we het beste de opdracht in eisen omschrijven:

- **Eis 1** : De gebruiker moet één of meer slides met inhoudsopgaven kunnen toevoegen aan een presentatie.
- **Eis 2** : Een inhoudsopgave-slide moet alle onderwerpen van de presentatie tonen.
- **Eis 3** : Het onderwerp moet slechts eenmaal worden getoond wanneer meerdere slides achter elkaar hetzelfde onderwerp hebben.
- **Eis 4** : De inhoudsopgave moet automatisch worden geüpdatet wanneer slides worden toegevoegd of verwijderd.
- **Eis 5** : De titel moet als onderwerp worden gebruikt wanneer een slide geen onderwerp heeft (**aanname**).
- **Eis 6** : Het onderwerp mag op slides worden getoond, behalve in de inhoudsopgave (optioneel).

- **Eis 7** : De onderwerpen in de inhoudsopgave mogen volgnummers krijgen (optioneel).
- **Eis 8** : Het onderwerp van de slide die na een inhoudsopgave komt, mag een speciale aanduiding krijgen (i.e. een alternatieve letterstijl) (optioneel).
- **Eis 9** : De stijl van de inhoudsopgave is vrij te bepalen.

De opdracht om een inhoudsopgave te implementeren in Jabberpoint is best interessant. Om een inhoudsopgave te implementeren, is kennis nodig van de hele presentatie. Een verandering in elke slide kan invloed hebben op alle inhoudsopgaven. Deze circulaire afhankelijkheid maakt de opdracht uitdagend.

2 Ontwerp

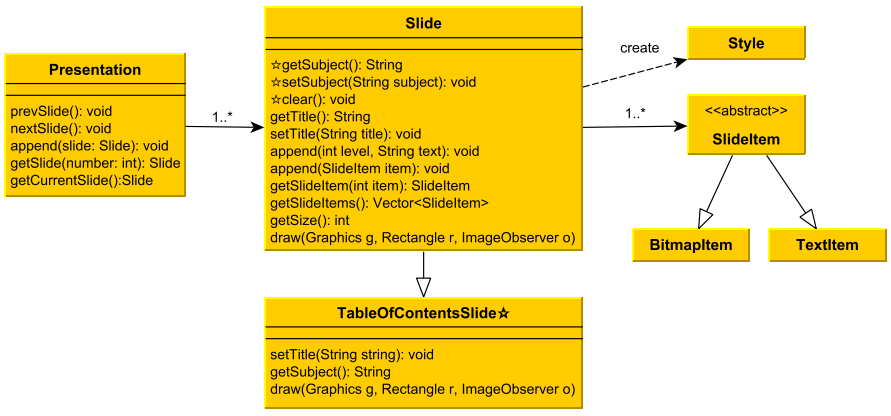
Het onderwerp zoals geïmplementeerd verandert weinig van het originele Jabberpoint-ontwerp. Er is een nieuwe klasse toegevoegd voor de inhoudsopgaven, namelijk `TableOfContentsSlide`. Deze nieuwe klasse is een sub-klasse van `Slide`.

Het grootste verschil tussen de klassen is dat de inhoudsopgave kennis moet hebben van de hele presentatie. Daardoor krijgt een `TableOfContentsSlide` een referentie naar de presentatie in de constructor.

Zodra de inhoudsopgave getekend gaat worden, worden de presentatie onderwerpen gelezen en wordt een nieuwe lijst van sub-items gemaakt. Daartoe kunnen wij de bestaande methoden van de `Slide` klasse gebruiken. Omdat er geen speciale stijl-eisen zijn voor de inhoudsopgave, kunnen wij eenvoudig de al ingebouwde levels gebruiken om de stijl te bepalen.

De aangepaste klassendiagram wordt in figuur 1 getoond.

Figure 1: Klassendiagram voor de domein-objecten. Sterren geven de nieuwe klasse en methoden aan.



Het voordeel van dit onderwerp is dat er vrij weinig hoeft te veranderen. De Accessor-klassen krijgen de extra verantwoordelijkheid om de slide-objecten te maken en te schrijven. Voor de rest van de code is een inhoudsopgave een normale slide die zijn eigen content genereert. Accessor-klassen mogen zelf bepalen

of er een referentie wordt opslagen naar een inhoudsopgave, of een lijst van de inhoud van de slides.

Voor de implementatie van de `draw` methode was een manier nodig om alle eerdere slide-items te verwijderen. Het is natuurlijk ook mogelijk om de `items` attribuut rechtstreeks te veranderen; maar het is in Java nooit verstandig om attributen van andere klassen te veranderen. Daarvoor komt een nieuwe **protected** methode `clear` in de slide. De attributen van `Slide` zijn **private** geworden.

Het is verstandig om software te implementeren in de meest simpele manier, en dan itereren om deze te verbeteren. Daarom vind ik dat het bij de opdracht past om een aantal verbeteringen ook door te voeren. Dit wordt verder uitgelegd in de volgende sectie.

2.1 Traceerbaarheid van eisen

In deze sectie geef ik onderdelen van het ontwerp aan die verantwoordelijk zijn om de eisen (zie sectie 1) te realiseren. Dit geeft dus de relatie tussen probleemanalyse en ontwerp.

- De `Accessor`-klassen zijn verantwoordelijk voor de volgende eisen: **Eis 1**.
- De `TableOfContentsSlide` klasse is verantwoordelijk voor: **Eis 2**, **Eis 3**, **Eis 4**, **Eis 5**, **Eis 7**, **Eis 8** en **Eis 9**.
- De `Slide` klasse is verantwoordelijk voor: **Eis 6**.

3 Keuzen

Klasse `TableOfContentsSlide` is uiteraard verantwoordelijk voor het genereren van de inhoudsopgave. Daarbij hebben we de optie om de slide-items die al geïmplementeerd zijn te hergebruiken. Het is dus logisch om de software zo optimaal mogelijk te hergebruiken, en niet alleen de slide-items maar ook het tekenen van deze items. De inhoudsopgave is in feite een slide die zijn eigen content genereert.

In de rest van deze sectie gaan we een aantal vragen door die zijn opgeroepen tijdens het analyseproces.

Hoe moet de Slide klasse-structuur eruit zien?

Het is vanzelfsprekend dat `TableOfContentsSlide` een groot deel van de `Slide` code nodig heeft. Bij het analyseren van Jabberpoint heb ik gemerkt dat de nieuwe klasse eigenlijk *alle* code moet gebruiken. Het enige onderdeel dat niet nodig is voor de inhoudsopgave is het nieuwe onderwerp-veld.

Meestal als een variatie komt voor een bestaand concept, moet er een interface (of abstracte klasse) komen die het oude en het nieuwe concept implementeren. De rest van het systeem werkt dan met de interface en hoeft niets te weten van de exacte implementatie. Dit is hier ook het geval, de presentatieklassen hoeven immers niet te weten of een slide een inhoudsopgave is of niet.

We zouden dus `Slide` abstract kunnen maken en een nieuwe klasse maken die naast de bestaande methoden een onderwerp heeft. Als we echter denken aan

wat voor andere soorten `Slide` er in de toekomst nog komen, is het aannemelijk dat deze ook in de inhoudsopgave moeten worden getoond.

Een alternatief ontwerp is geïmplementeerd in PR #17. In deze PR is een extra klasse `ContentSlide` gemaakt die slide-items heeft. Hierin maakt de `TableOfContentsSlide` elke keer een nieuwe `ContentSlide`. Dit ontwerp is een geldig alternatief maar er zijn niet veel voordelen. De software wordt ingewikkelder en er moeten extra *casts* worden uitgevoerd.

Wanneer items genereren

De table of contents is in feite een slide met een inhoud die gegenereerd wordt. De vraag is wanneer deze generatie moet gebeuren. In het gepresenteerde ontwerp wordt de slide opnieuw gegenereerd in elke `draw`.

Een alternatief zou zijn om een manier te implementeren om de `TableOfContentsSlide` te laten weten als iets verandert. Bijvoorbeeld de observable pattern zou hier gebruikt kunnen worden.

Dat is echter niet nodig, de generatie is redelijk efficiënt aangezien `draw` alleen wordt aangeroepen als de slide zichtbaar is. `SlideViewerComponent` roept immers `repaint` alleen aan wanneer de slide op het scherm komt.

logic voor lezen van slides in presentatie?

toc te veel kennis van slide: misschien nieuwe slide maken en die laten tekenen? maar dan bij output heb je de items niet..

refactoring van UI-domein vermengeling

moet een inhoudsopgave verplicht een titel hebben?

layouts i.p.v. draw

4 Sourcecode

In deze sectie geef ik links naar de verschillende resultaten van de opdracht. Alle links hebben betreft zowel op Jabberpoint als voor dit rapport.

- Java en LaTeX code: github.com/DanielSchiavini/design-patterns-assignment.
- Aanpassingen op Jabberpoint: github.com/DanielSchiavini/design-patterns-assignment/compare/v0.0.1...master.
- Gegenereerde bestanden: github.com/DanielSchiavini/design-patterns-assignment/tree/gh-pages
- Laatste build-rapport van TravisCI: travis-ci.com/DanielSchiavini/design-patterns-assignment
- Rapporten van TravisCI: travis-ci.com/DanielSchiavini/design-patterns-assignment/builds
- Optionele implementaties: github.com/DanielSchiavini/design-patterns-assignment/pulls