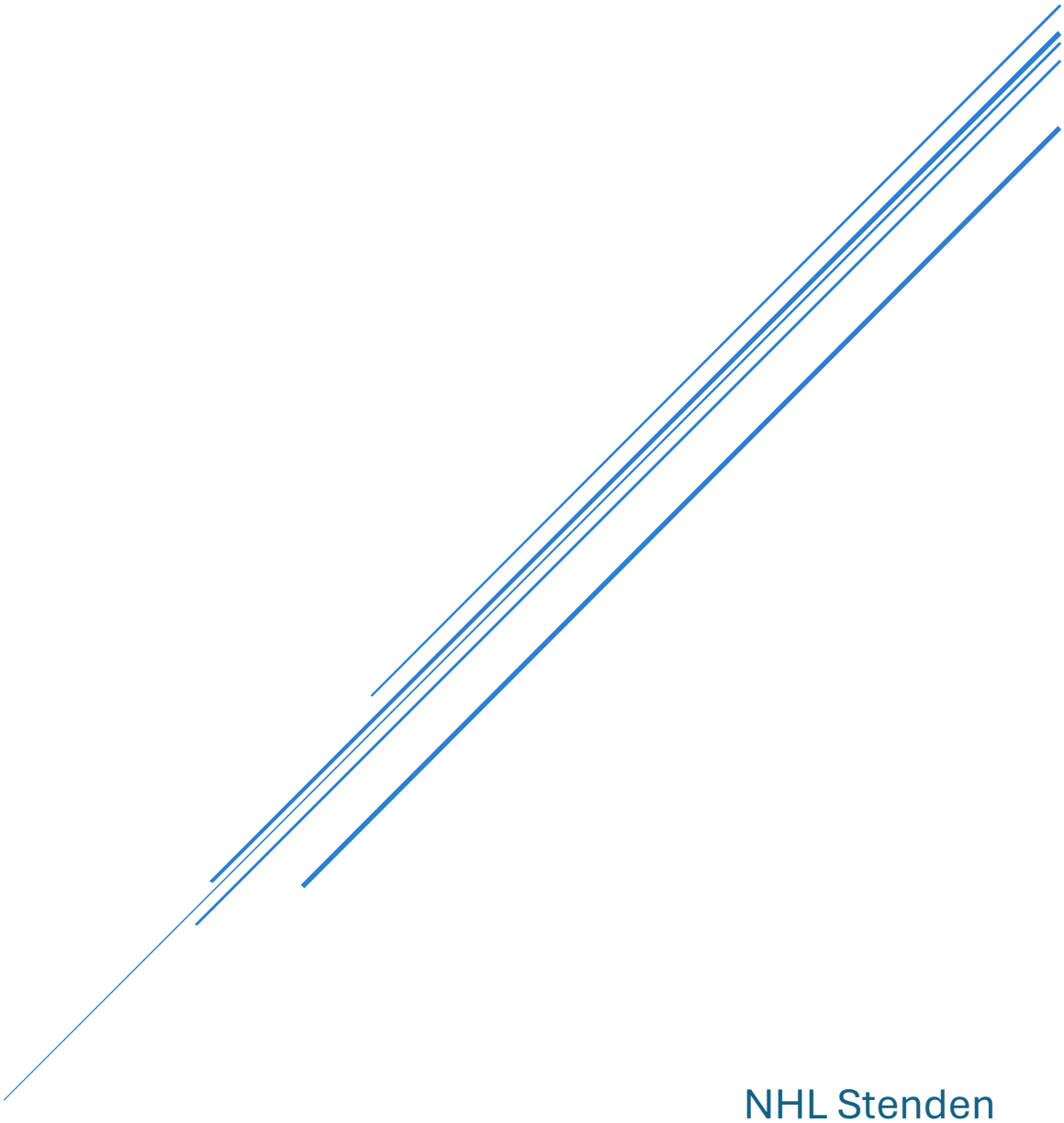
SYSTEEM ANALYSE

Jabberpoint



NHL Stenden

Software Quality – Elmedin Arifi & Yunus Karakoc

Contents

Inleiding

Dit document bevat een analyse van JabberPoint, een programma om presentaties te bekijken. Deze analyse is gemaakt voor het vak Software Quality en is gebaseerd op de code op GitHub:

https://github.com/NHL-Stenden-Emmen/Jabberpoint.

JabberPoint is een simpel programma: je laadt er presentaties mee in en kunt ze dan bekijken. Er zitten wat basis-functies in om tussen de dia's te wisselen (pijl-omhoog, pijl-omlaag). We gaan analyseren hoe JabberPoint is opgebouwd en documenteren onze bevindingen in dit document.

Om dat te doen, maken we vier diagrammen: een use case, class, activity en een sequence diagram. Die vier diagrammen geven een beter beeld over hoe JabberPoint opgebouwd is.

Werking van software

JabberPoint.java is het begin van de applicatie en begint met het bepalen van de stijl van de tekst op de dia's.

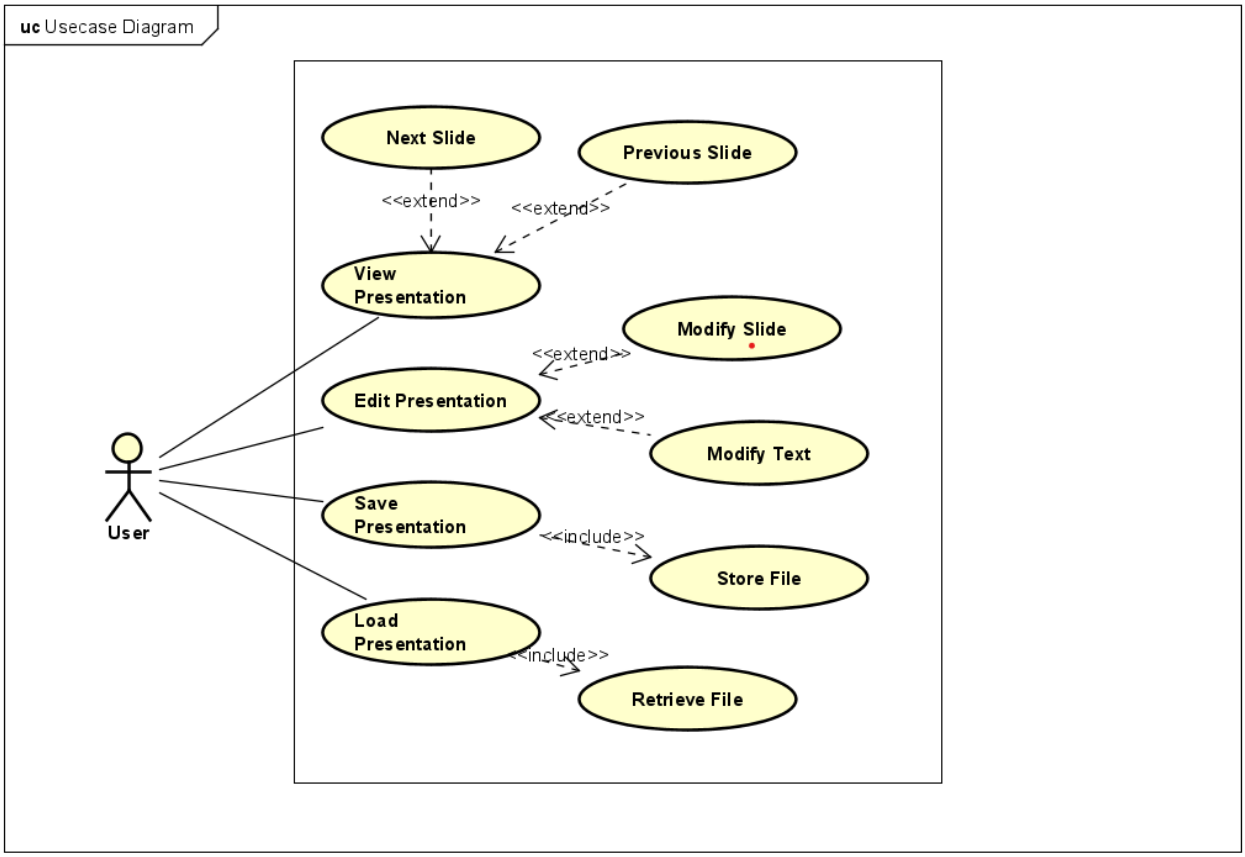
Daarna wordt de een Presentation object gemaakt. Dit object is als een map die alle dia's en hun inhoud bevat. Vervolgens wordt een Frame gecreëerd waarop de presentatie getoond wordt: een SlideViewerFrame. Binnen dit venster bevindt zich de SlideViewerComponent.

JabberPoint kijkt of er een bestandsnaam is meegegeven. Zo niet, dan wordt de "demo-presentatie" geladen. Deze is als een vooraf ingevulde set dia's. Om dit te doen wordt een speciale "lader" gebruikt, een DemoPresentation object. Deze "lader" vult de Presentation met de demo-dia's. Is er wel een bestandsnaam gegeven, dan wordt een andere "lader" gebruikt, een XMLAccessor object. Deze "lader" leest de presentatiegegevens uit een XML-bestand en vult hiermee de Presentation.

Het SlideViewerFrame bevat de SlideViewerComponent (het scherm) en verwerkt de input van de gebruiker. Er zijn twee "controllers": de KeyController voor toetsaanslagen en de MenuController voor menu-opties. De KeyController luistert naar toetsaanslagen en geeft commando's door aan de Presentation, zoals "volgende dia" of "vorige dia". De MenuController doet hetzelfde voor menu-opties, zoals "openen", "opslaan" of navigeren.

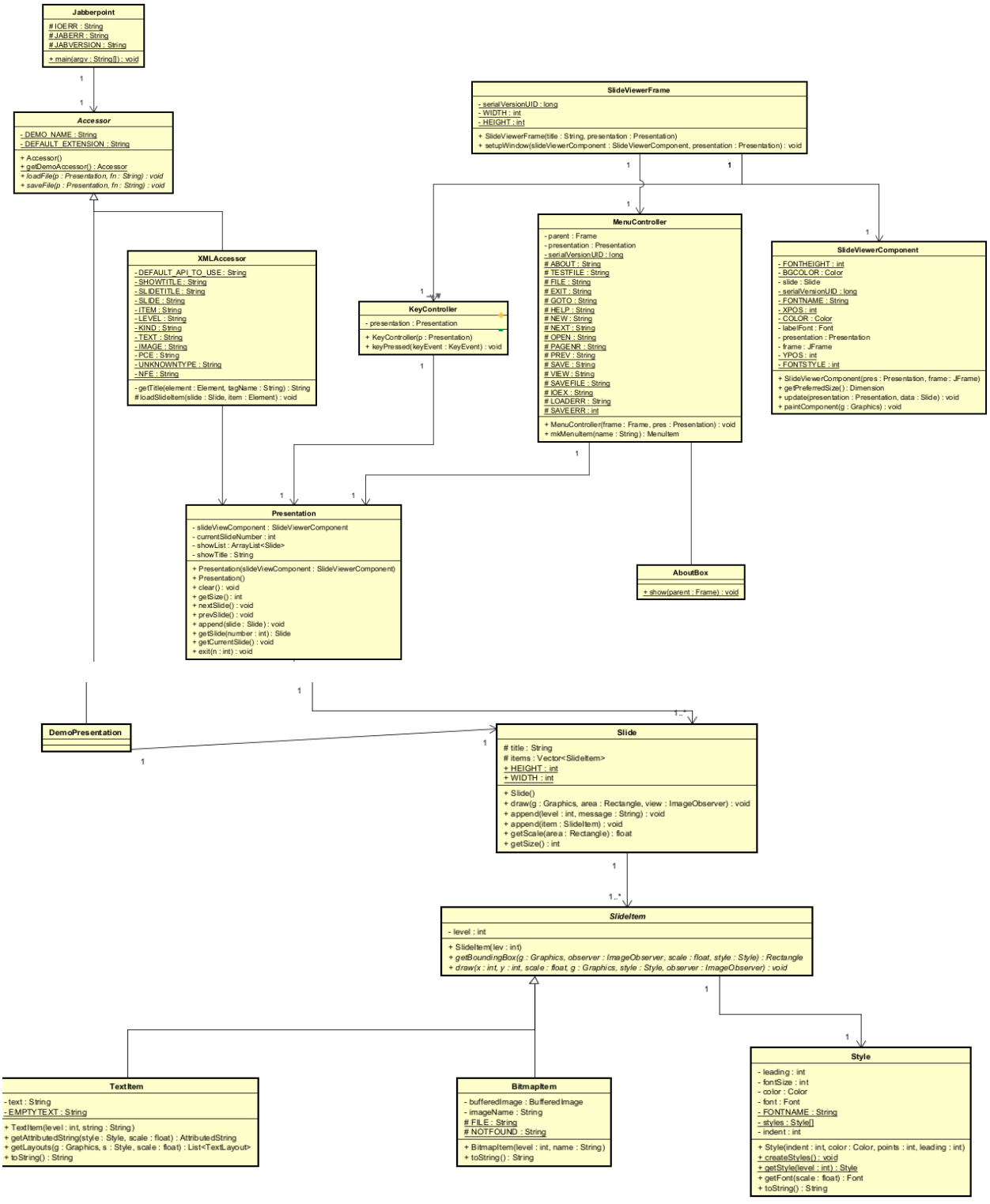
De SlideViewerComponent is verantwoordelijk voor het tekenen van de dia's. Wanneer de huidige dia verandert, krijgt de SlideViewerComponent een seintje. Vervolgens haalt het de huidige Slide op uit de Presentation. Elke Slide bevat een verzameling van abstracte SlideItem objecten. Deze elementen kunnen tekst zijn (TextItem) of afbeeldingen (BitmapItem). De SlideViewerComponent gaat door al deze elementen heen en tekent ze op het scherm. Elk element gebruikt een Style object om te bepalen hoe het eruit moet zien (lettertype, kleur, etc.). Als de gebruiker de presentatie wil opslaan, wordt de XMLAccessor weer gebruikt om de gegevens in een XML-bestand te schrijven.

Usecase diagram



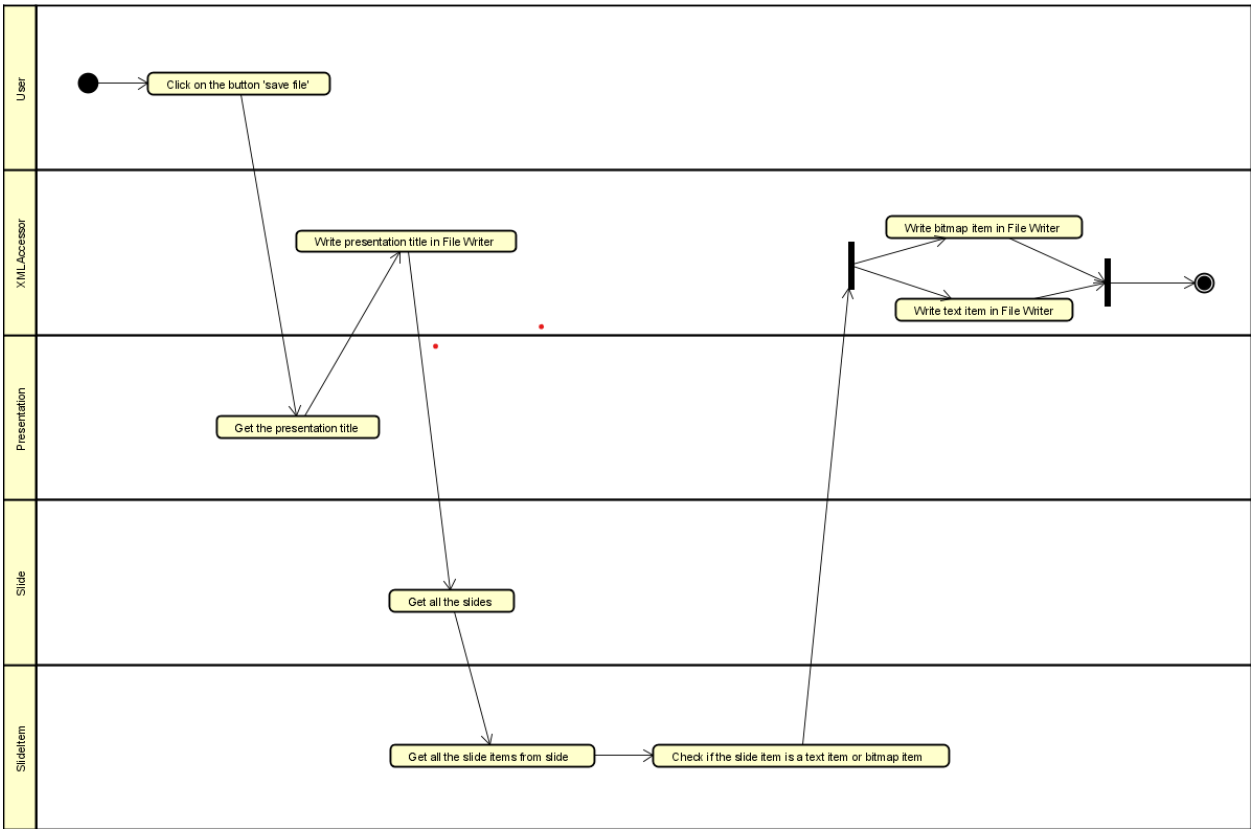
Figuur 1 : Use Case Diagram

Class diagram



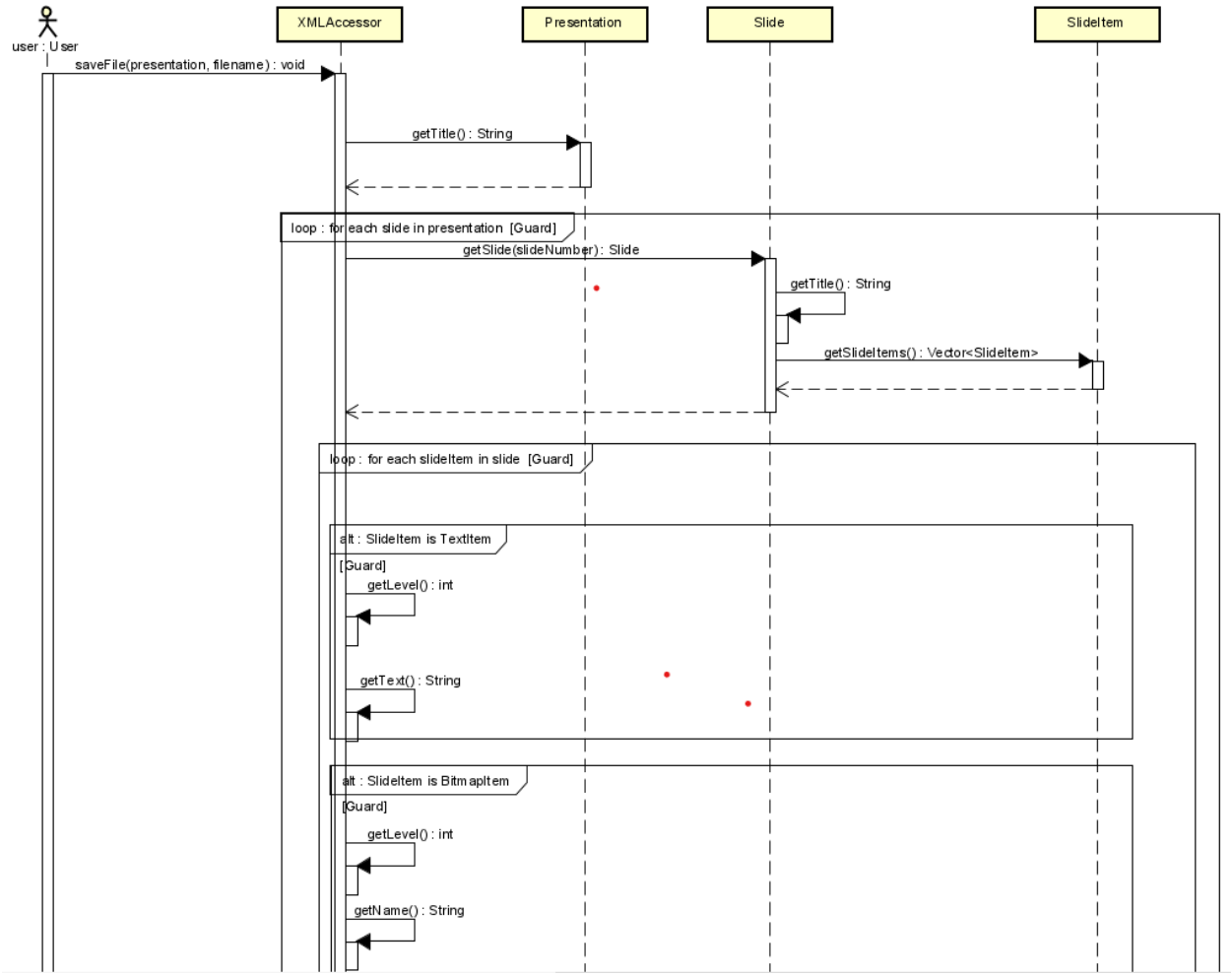
Figuur 2 : Class Diagram

Activity diagram



Figuur 3 : Activity Diagram

Sequence diagram



Figuur 4 : Sequence Diagram

Klassen

Klasse: TextView.java

Doel: Toont een lijst met dia-titels (nog niet de inhoud).

Wat het doet:

- Maakt een lijst (JList) voor de dia-titels.
- Haalt data uit het Model (presentatie data).
- Update de lijst als het Model verandert.

Klasse: Style.java

Doel: Beheert de style eigenschappen van tekst, zoals inspringen, kleur, lettertype, lettergrootte en regelafstand (leading).

Wat het doet:

- Slaat stijl-informatie op.
- Biedt getters en setters voor de verschillende stijl-eigenschappen.
- Heeft een toString() methode voor debugging/weergave.

Klasse: Slide.java

Doel: Representeert één dia in een presentatie.

Wat het doet:

- Bevat een titel (title).
- Slaat een lijst op van M objecten (ms). Deze M objecten vertegenwoordigen de inhoud van de dia (tekst, afbeeldingen, etc.).
- Biedt methoden om M objecten toe te voegen (append()), de titel op te halen/in te stellen (getTitle(), setTitle()), en de lijst met M objecten op te vragen (getMs()).

Klasse: ShowView.java

Doel: Toont de presentatie-dia's grafisch op het scherm.

Wat het doet:

- Tekent de inhoud van de huidige dia (Slide) op het scherm.
- Haalt de dia-informatie uit het Model.
- Gebruikt Style objecten om de tekst opmaak te bepalen.
- Implementeert Observer om updates van het Model op te vangen (wanneer een nieuwe dia wordt geselecteerd).

Klasse: MenuController.java

Doel: Beheert de menu-balk van de applicatie en de acties die aan de menu-items zijn gekoppeld.

Wat het doet:

- Creëert de menu's (Bestand, Bewerken, Weergave, Help).
- Voegt menu-items toe en koppelt ActionListeners aan deze items.
- Implementeert de logica voor de verschillende menu-acties (openen, opslaan, afsluiten, volgende dia, vorige dia, etc.).
- Gebruikt een ResourceBundle (JabberPointMenus) voor de labels van de menu's en menu-items.

Klasse: KeyController.java

Doel: Behandelt toetsaanslagen van de gebruiker en stuurt de juiste acties door naar het Model.

Wat het doet:

- Luistert naar toetsaanslagen (KeyEvent).
- Reageert op specifieke toetsen (Pijltje-omlaag, spatie, enter, Page Down, +, Pijltje-omhoog, -, Page Up, Q).
- Roept methoden aan op het Model om naar de volgende of vorige dia te gaan, of om de applicatie af te sluiten.

Klasse: JabberPoint.java (de hoofdklasse)

Doel: Start de JabberPoint applicatie en coördineert de verschillende componenten.

Wat het doet:

- Creëert het Model (data van de presentatie).
- Instantieert de ShowView (grafische weergave) en TextView (tekstuele weergave) en koppelt ze aan het Model.
- Zet de GUI op (JFrame, JSplitPane).
- Creëert de KeyController (toetsbediening) en MenuController (menu).
- Laadt de presentatie (demo of van een bestand).
- Definieert een array van Style objecten voor de tekststijlen.

Klasse: DynamicList.java

Doel: Demonstreert het dynamisch toevoegen en verwijderen van elementen aan een JList met behulp van een DefaultListModel en een KeyListener.

Wat het doet:

- Creëert een JList met een DefaultListModel.
- Implementeert een KeyListener die:
  - Tekens toevoegt aan de lijst wanneer ze worden getypt (behalve backspace).
  - Het laatste element verwijdert wanneer backspace wordt getypt.
- Zet de GUI op (JFrame, JScrollPane).

Klasse: AboutBox.java

Doel: Toont een "Over" dialoogvenster met informatie over JabberPoint.

Wat het doet:

- Toont een JOptionPane met de JabberPoint naam, beschrijving, copyright informatie en auteur.

Klasse: MText.java

Doel: Representeert een tekstitem binnen een dia. Het beheert de tekst zelf en de opmaak ervan.

Wat het doet:

- Slaat de tekst op (text).
- Bepaalt de opmaak (lettertype, kleur, etc.) via een Style object.
- Berekent de afmetingen van de tekst (getBBox()).
- Tekent de tekst op het scherm (draw()).
- Splits lange tekst in meerdere regels indien nodig (getLayouts()).

Klasse: Model.java

Doel: Beheert de data van de presentatie, inclusief de dia's en de huidige dia. Het fungeert als het centrale gegevensopslag en communiceert wijzigingen naar de views.

Wat het doet:

- Slaat de dia's op in een List (showList).
- Houdt het huidige dianummer bij (pageNumber).
- Biedt methoden om dia's toe te voegen (append()), op te halen (getSlide(), getCurrentSlide()), en de huidige dia te wijzigen (setSlideNumber()).
- Implementeert Observable om views te informeren over wijzigingen in de data.
- Implementeert ListModel om te integreren met Swing-componenten zoals JList.
- Beheert de titel van de presentatie (showTitle).

Klasse: MCodeInsert.java

Doel: Vertegenwoordigt een code-fragment dat in de presentatie is opgenomen. Het toont een placeholder en opent een extern tekstbestand in een editor (xterm/vi).

Wat het doet:

- Slaat de bestandsnaam op (fileName).
- Toont een placeholder tekst "(see code window)" op de dia.

- Bij het tekenen (draw()), wordt een extern proces gestart (xterm/vi) om het codebestand te openen.

Klasse: Mcode.java

Doel: Vertegenwoordigt een stuk code dat *direct* in de presentatie wordt weergegeven (in tegenstelling tot MCodeInsert dat een extern bestand opent). Het is een subklasse van MText en erft de meeste functionaliteit.

Wat het doet:

- Slaat de code op (geërfd van MText).
- Gebruikt een speciale Style (JabberPoint.getCodeStyle()) voor de weergave.
- Definieert een vaste grootte voor de bounding box (getBBox()).

Klasse: Mbitmap.java

Doel: Vertegenwoordigt een afbeelding (bitmap) binnen een dia.

Wat het doet:

- Slaat de bestandsnaam van de afbeelding op (imageName).
- Laadt de afbeelding met Toolkit.getDefaultToolkit().getImage().
- Berekent de afmetingen van de afbeelding (getBBox()).
- Tekent de afbeelding op het scherm (draw()).

Klasse: M.java (abstract)

Doel: De abstracte basisklasse voor alle items die op een dia kunnen worden weergegeven (tekst, afbeeldingen, code). Het definieert de gemeenschappelijke eigenschappen en methoden voor deze items.

Wat het doet:

- Definieert het niveau (indentatie) van het item (level).
- Declareert abstracte methoden die door subclasses moeten worden geïmplementeerd:
  - getBBox(): Berekent de afmetingen van het item.
  - draw(): Tekent het item op het scherm.
- Implementeert de getStyle() methode om de juiste stijl op te halen op basis van het niveau.

Fouten

- Zeer verwarrende comments (Bijv: SlideItem.java, Lijn: 27)
- Verkeerde indentatie van code (Bijv: SliderViewerFrame.java, Lijn: 31-46)
- Geen this.ATTRIBUTE gebruikt op veel plekken van de code (Bijv: SlideViewerComponent.java, Lijn: 48-57)
- XMLaccessor.java geen constructor
- Hardcode (Style.java)
- Fouten in strings van Exceptions (DemoPresentation.java, Lijn: 52)
- Fout in path naar image (DemoPresentation.java, Lijn: 47)
- Geen gebruik van packages in het systeem
- Geen UnitTests