Informatik IIB

BibTeX-Datenbank

alternative Prüfungsleistung

Jürgen Döffinger (631551),
Michael Goldbuch (631430),
Christoph Kebbel (631556)

23. Januar 2012

Inhaltsverzeichnis

Abbildungsverzeichnis

Teil I.
Analyse

1. Aufgabenstellung

Wir haben uns für das Projekt BibTeX-Datenbank entschieden. Die Aufgabenstellung beinhaltete folgende Punkte:

> Entwickeln Sie eine Software zur Bearbeitung und Verwaltung von BibTeX-Datenbanken. BibTeXist eine Software, mit deren Hilfe Literaturdatenbanken aufgebaut und Literaturangaben in LATEX-Dokumente eingebunden werden können. Das BibTeX-Datenbankformat ist eine strukturierte Textdatei der Form
>
> ```
> @article{mrx05,
> author = {Mr. X},
> title = {Something Great},
> publisher = {nobody},
> year = 2005,
> }
> ```
>
> Das Format lässt viele verschiedene Einträge zu, die mit dem Dokumenttyp beginnen, z. B. article für einen Zeitschriftenartikel, mrx05 als Eintragschlüssel, und viele Eintragfelder besitzen, z. B. author für den Autor des Artikels. Manche dieser Felder sind optional, müssen also nicht angegeben sein, andere sind obligatorisch, müssen also vorhanden sein. Welche Felder obligatorisch und welche optional sind, hängt vom Eintragtyp ab. Bei der Angabe der Feldnamen, z. B. author, wird nicht zwischen Groß- und Kleinschreibung unterschieden. Eine Beispiel- und Testdatei mit BibTeX-Einträgen finden Sie auf der Website der Informatik IIb-Übung:
> http://www.fh-jena.de/ jack/Lehre/InfoIIb/Uebung/xampl.bib,
> http://www.fh-jena.de/ jack/Lehre/InfoIIb/Uebung/test.bib
> Weitere Informationen zu BibTeXfinden sie z. B. hier: http://www.bibtex.org/

Die von Ihnen zu entwickelnde Software soll eine grafische Benutzeroberfläche besitzen und soll

- mehrere BibTeX-Dateien laden können,
- die darin enthaltenen Einträge auf Korrektheit und Konsistenz prüfen, ggf. eine Bildschirm-Ausgabe erzeugen, die die unkorrekten bzw. inkonsistenten Eintrage anzeigt,
- neue Literatureinträge anlegen und bestehende bearbeiten als auch löschen können, sowohl durch Tastatureingabe als auch durch Eingabe über den Copy-Paste-Puffer, bei letzterem ist der Inhalt des Paste-Puffers ein Text wie oben im Beispiel @article{mrx05,
- Freitextsuche in den Feldern der Eintragtypen ermöglichen d. h. die gefundenen Einträge werden mit Ihrem Schlüssel in einer Liste angezeigt in einem anderen Fenster können die gefunden Einträge im Detail betrachtet und bearbeitet werden,
- eine Auswahl beliebiger Eintrage als neue BibTeX-Datenbankdatei schreiben konnen,
- ein zusätzliches optionales Feld für jeden Eintragtyp mit dem Bezeichner abstract beinhalten,
- ein weiteres zusätzliches optionales Feld für jeden Eintragtyp mit der Bezeichnung link beinhalten, das einen Hyperlink zu einem Dokument (das wird sinnvollerweise das Dokument, das im Eintrag beschrieben ist, sein) enthält; bei einem Mausklick auf den Link soll dieser Link mit dem auf den Zielsystem verbundenen Programm geöffnet werden, z. B. bei einem PDF-Dokument mit dem Adobe-Reader.

2. Entwurf

In der Analysephase ging es darum, die Aufgabenstellung in einen Programmablauf zu bringen. Dabei mussten zunächst die grundsätzlichen Aufgaben ermittelt werden und in sinnvolle Pakete aufgeteilt werden. Wir entschieden uns für drei wesentliche Pakete.

Das erste Paket ist die grafische Benutzeroberfläche (GUI - Graphical User Interface). Dieses hat folgende Aufgaben:

- Auswahlmöglichkeiten der anzuzeigenden Daten
- Anzeige der Daten mit oder ohne Filterung
- Befehlseingabe durch den User und Weiterleitung an die entsprechenden Objekte

Dazu ist eine Kommunikations zwischen Mensch und GUI und eine Kommunikation zwischen den anderen Paketen und der GUI sicherzustellen.

Das zweite Paket ist die Datenbank (Database). Die Datenbank hat folgende Aufgaben zu realisieren:

- Aufnehmen, Ändern und Löschen von Daten aus der Datenbank (sowohl Arbeitsspicher als auch Datenbankdatei)
- Austausch von Daten zwischen Datenbank und Bibtex
- Austausch von Daten zwischen Datenbank und GUI

Das dritte Paket ist das Paket *BibTex*. Das Paket umfasst folgende Aufgaben:

- Parsen der Daten aus BibTeX-Datei bzw. von BibTeX-Datensätzen aus der Zwischenablage
- Serialisieren der Daten aus der Datenbank in eine BibTeX-Datei
- Validierung der Daten aus einer BibTeX-Datei.

Teil II.
Implementierung

In den folgenden Abschnitten wird die Umsetzung in die entsprechenden Paketen und Klassen erläutert.

3. Package de.BibTexMaker

Das Package *de.BibTexMaker* beinhaltet das gesamte Programm BibTexMaker. Die darin enthaltenen Packages und Klassen werden im folgenden aufgezählt und deren Funktion erläutert.

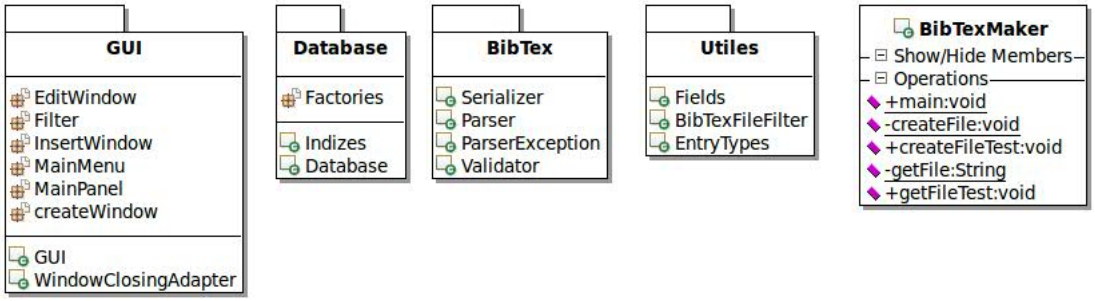

Abbildung 1: Klassendiagramm Package *de.BibTexMaker*

3.1. Klasse BibTexMaker

Die Klasse *BibTexMaker* beinhaltet die Methoden *main, createFile* und *getFile*. Die Klasse bildet den Einstieg in das Programm BibTexMaker

Die Methode *main* ist der Einstieg in das Programm BibTexMaker. Sie prüft ob Argumente übergeben wurden und reagiert entsprechend. Wurde ein Dateiname mit Pfad für eine BibTexMaker-Datenbank-Datei übergeben, so veranlasst die Methode das Laden dieser. Ansonsten wird eine leere Datenbank (./library.xml) angelegt. Dabei greift die Methode *main* auf die Methode *getFile* zu und übergibt dieser die Argumentenliste.

Die Methode *getFile* liest die dem Programm übergebenen Parameter aus und wertet den Parameter *-f* aus und gibt den Dateinamen mit Pfadangabe zurück. Der Parameter *-f* ist dafür gedacht eine bereits vorhanden Datenbankdatei direkt beim Programmstart zu laden. Wurde kein Parameter *-f* übergeben, so wird der MEthode *main* ein leerer String zurückgegeben.

Die Methode *main* prüft nun ob die Datei exisitert. Existiert die Datei noch nicht wird diese erzeugt. Dazu wird die Methode *createFile* aufgerufen.

Die Methode *createFile* erstellt eine neue Datenbankdatei mit dem entsprechenden Format. Anschließend übergibt die Methode *main* den Pfad und den Dateinahmen der Datenbank, so dass diese die Daten laden kann.

Zum Schluss startet die Methode *main* die GUI.

3.2. Package de.BibTexMaker.GUI

Das Package *de.BibTexMaker.GUI* beinhaltet alle Packages und Klassen welche zum Graphical User Interface (GUI) gehören.

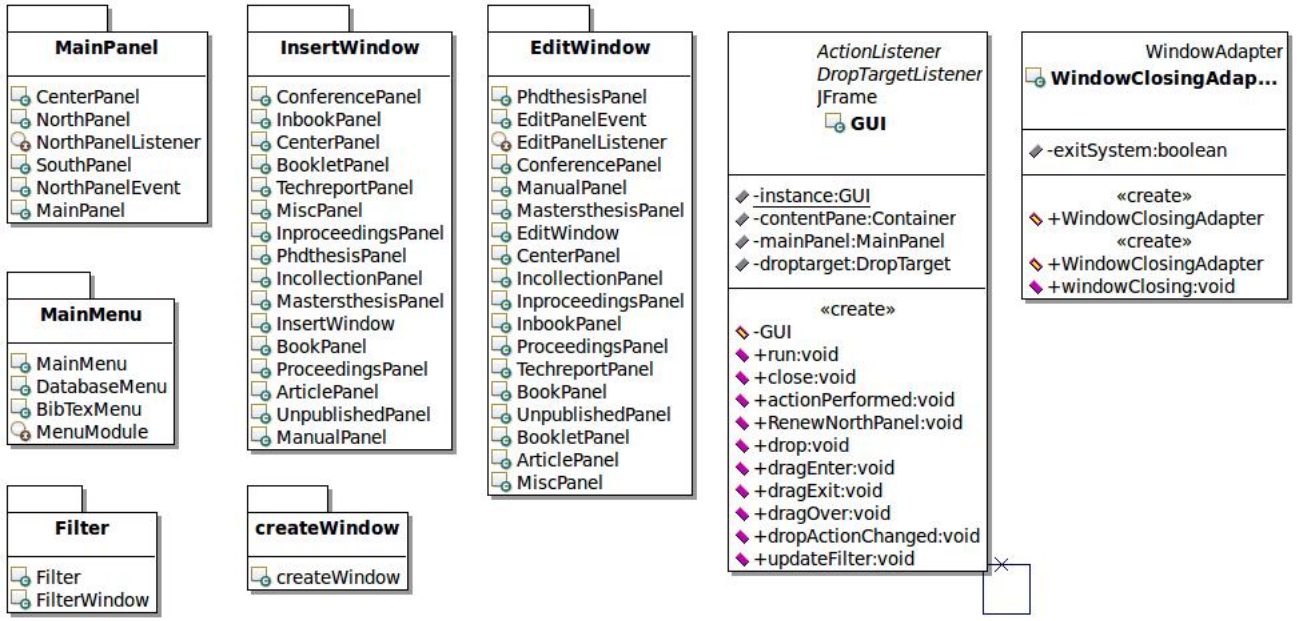

Abbildung 2: Klassendiagramm Package *de.BibTexMaker.GUI*

3.2.1. Klasse GUI

Die Klasse *GUI* ist, wie die Abkürzung schon vermuten lässt, das Graphic Unit Interface des Programms bzw. beinhaltet alle notwendigen Methoden, um dieses darzustellen. Die Klasse ist nach dem Singleton-Pattern-Verfahren programmiert. Das heißt es kann im Programm nur eine Instanz der Klasse *GUI* erzeugt werden. Dazu wurde der Konstruktor private gekennzeichnet und eine Methode *getInstance* implementiert. Mithilfe der Methode *getInstance* kann auf die Instanz der Klasse zugegriffen werden. Ist die Instanz noch nicht angelegt, wird Sie durch die Methode *getInstance* erzeugt.

Mithilfe der Methode *run* wird die GUI zur Anzeige gebracht. Die Methode stellt dabei die Größe des Bildschirms fest und berechnet entsprechend die Größe des Hauptfensters und setzt dieses in die Mitte des Bildschirms.

Die Methode *close* kann dazu genutzt werden einen Window_Closing-Event auszulösen, um damit das Fenster oder das Programm zu beenden.

Mit der Methode *RenewNorthPanel* wird veranlasst, dass das NorthPanel neu aufgebaut wird und somit aktualisiert wird. Das NorthPanel beinhaltet die Buttons, die Comboboxen zur Auswahl des Publikationstypen usw.

Die Methode *drop* reagiert auf einen Drop-Event und veranlasst das Prüfen des übergebenen Elementes. Liegt hier eine BibTex-Datei vor, so wird diese an den Parser weitergeleitet, damit die enthaltenden Daten in die Datenbank aufgenommen werden.

Die Methode updateFilter löst das aktualisieren des Filters aus.

Alle anderen Methoden haben keine Funktion, sondern sind vorhanden, da dies die Implementation der Interfaces ActionListener und DropTargetListener erfordert.

3.2.2. Klasse WindowClosingAdapter

Die Klasse WindowClosingAdapter dient zur Entscheidung, ob nur das Hauptfenster geschlossen werden soll oder das gesamte Programm beendet werden soll, wenn ein Window_Closing-Event ausgelöst wird.

Dazu kann die Klasse über zwei Konstruktoren instantiiert werden. Wird ein Objekt der Klasse *WindowClosingAdapter* ohne übergabe des Parameters *exitSystem* übergeben, so wird

bei auslösen des Window_Closing-Events lediglich das Hauptfenster geschlossen. Wird der Parameter übergeben, dann wird bei true auch das Programm beendet und nicht nur das Hauptfenster geschlossen.

Wird nun ein Window_Closing-Event ausgelöst, dann wird die Methode *windowClosing* ausgeführt. Dabei schließt Sie das Filterfenster falls geöffnet und das Hauptfenster. Anschließend wird geprüft ob das Attribut *exitSystem* auf true gesetzt wurde. Ist dies der Fall veranlasst die Methode das Beenden des Programmes.

3.2.3. Package MainPanel

Das Package *MainPanel* beinhaltet alle Klassen zum Aufbau des Hauptfensters.

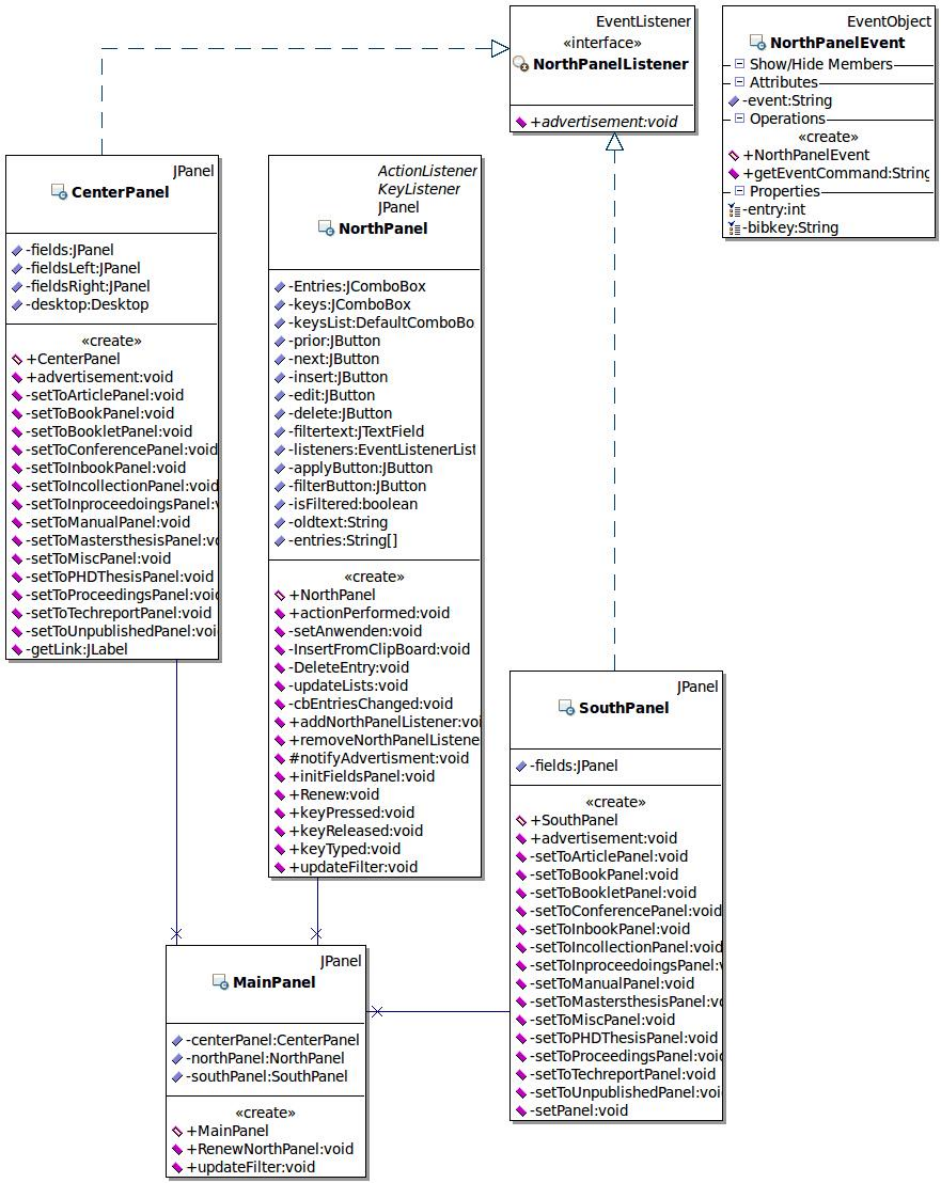

Abbildung 3: Klassendiagramm Package *de.BibTexMaker.GUI.MainPanel*

3.2.3.1. Klasse MainPanel

Die Klasse MainPanel beinhaltet alle Methoden zum Aufbau des Hauptfensters. Dazu wird im Konstruktor das Hauptfenster initialisiert. Dazu wird das Layout und die Border festgelegt. Als Layout wird das BorderLayout verwendet. An die Position North kommt das NorthPanel, Center das CenterPanel und an die Position South das SouthPanel. Es werden als Attribut jeweils ein Objekt des North-, Center- und SouthPanel angelegt.

Des Weiteren meldet der Konstruktor das CenterPanel und das SouthPanel am NorthPanel-Listener an, damit diese auf NorthPanelEvents reagieren können.

Zum Schluss wird die Methode *initFieldsPanel* des NorthPanels ausgeführt, um die Comboboxen für den Publikationstyp und der Auswhl der Publikation zu füllen.

Zur Klasse *NorthPanel* gehören auch noch die Methoden *RenewNorthPanel* und *updateFilter*, welche das Aktualisieren des Filterfensters und des NorthPanels veranlassen.

Das MainPanel ist eine abgeleitete Klasse von JPanel.

3.2.3.2. Klasse NorthPanel

Die Klasse NorthPanel erzeugt das NorthPanel und kümmert sich um die Ausführung der entsprechenden Methode bei betätigen eines Buttons oder der Änderung eines Eintrages in einer der Comboboxen. Im Konstruktor werden die einzelnen Elemente der Klasse initialisiert. Dabei wird innerhalb der Klasse das Panel in zwei weitere Panels geteilt. Das obere und das untere Panel. Das obere Panel beinhaltet die Auswahlliste für Publikationstyp und Publikation, sowie die Schaltflächen Vorheriger , Nächster, Hinzufügen, Bearbeiten und Löschen. Das untere Panel beinhaltet die Elemente des Filters. Dazu gehören das Editierfeld für den Filtertext und die Schaltflächen Felder und Anwenden / Abschalten.

Wird in der Auswahlliste der Publikationstypen ein anderer Publikationstyp ausgewählt, so wird die entsprechende Methode *actionPerformed* augeführt. Die Methode führt die Methode cbEntriesChanged aus. Anschließend wird wird die Auswahl in der Combobox neu festgelegt. und ein NorthPanelEvent ausgelöst, welcher veranlasst die Änderung in der Combobox zu übernehmen.

Wählt der Anwender in der Auswahlliste der Publikationen eine andere Publikation aus, so wird auch hier die zugehörige Methode *actionPerformed* aufgerufen. Die Methode löst

den NorthPanelEvent aus und teilt so den darauf hörenden Methoden mit, dass sich der Datensatz in der Auswahl geändert hat.

Betätigt der Anwender eine der Schaltflächen, wird die entsprechende Methode *actionPerformed* ausgeführt. Diese Methode prüft nun, welche Schaltfläche betätigt wurde.

Wurde die Schaltfläche HINZUFÜGEN betätigt, so erfolgt zunächst eine Abfrage ob Datensätze aus der Zwischenablage in die Datenbank aufgenommen werden sollen oder ob von Hand ein neuer Datensatz erstellt werden soll. Dies geschieht über ein JOptionPane-Objekt. Anschließend wird die Methode InsertFromClipBoard ausgeführt, wenn der Anwender sich entschieden hat von der Zwischenablage die Daten einzulesen, ansonsten wird ein Objekt vom Typ *InsertWindow* erstellt und mit der Methode *run* geöffnet. Anschließend wird die Methode *updateLists* ausgeführt, um die Änderungen in der Datenbank in der GUI anzuzeigen, also zu übernehmen.

Hat der Anwender die Schaltfläche BEARBEITEN betätigt, wird zunächst geprüft, ob ein Eintrag ausgewählt bzw. überhaupt vorhanden ist, welcher bearbeitet werden kann. Ist dies der Fall so wird eine Instanz des Fensters zum Bearbeiten von Datensätzen (EditWindow) angelegt. Dabei wird dem Konstruktor der BibKey und der Publikationstyp des zu bearbeitenden Datensatzes übergeben. Anschließend wird das Fenster geöffnet. Nachdem der Anwender das Fenster geschlossen hat, wird noch das Aktualisieren der entsprechenden Panels veranlasst.

Wenn die Schaltfläche LÖSCHEN betätigt wurde, wird die Methode *DeletEntry* aufgerufen, welche die ausgewählte Publikation aus der Datenbank entfernt.

Bei betätigen der Schaltfläche VORHERIGER, wird geprüft ob der selektierte Eintrag in der Combobox nicht der Erste ist. Ist dies der Fall so wird die Selektierung auf den davor liegenden Eintrag gesetzt. Die Combobbox löst somit einen Event aus der die darauf hörende Methode zu entsprechenden Änderungen veranlasst.

Bei betätigen der Schaltfläche NÄCHSTER, wird geprüft ob der selektierte Eintrag in der Combobox nicht der Letzte ist. Ist dies der Fall so wird die Selektierung auf den danach liegenden Eintrag gesetzt. Die Combobbox löst somit einen Event aus der die darauf hörende Methode zu entsprechenden Änderungen veranlasst.

Wird die Schaltfläche FELDER betätigt, so wird das Filterfenster (FilterWindow) aufgerufen.

Betätigt der Anwender die Schaltfläche ANWENDEN/ABSCHALTEN wird mithilfe der Methode *setAnwenden* der Filter ein- oder ausgeschalten.

Die Methode *setAnwenden* prüft zunächst ob der Filter bereits zugeschalten ist, wenn dem der Fall ist, wird er abgeschalten, ansonsten eingeschalten. Beim Einschalten wird der Status in *isFiltered* auf true gesetzt, der Filter-Button erhält die Beschriftung *Abschalten* und den ToolTipText *Filter abschalten*. Wird jedoch der Filter abgeschalten, so wird der Status in *isFiltered* auf false gesetzt und der Filter-Button erhält die Beschrtiftung *Anwenden* und der ToolTipText *Filter anwenden*. Weiterhin wird das Attribut *oldtext* gelöscht und dem Editierfeld des Filters zugewiesen. Sowohl beim Ein- bzw. Ausschalten, wird zum Schluss die Methode *cbEntriesChanged* ausgeführt,welche die ComboBox mit den Publikationseinträgen neu aufbaut.

Die Methode *InsertFromClipBoard* veranlasst alle Maßnahmen, um aus der Zwischenablage Datensätze in die Datenbank zu übernehmen. Dabei werden zunächst die Daten aus der Zwischenablage abgeholt und an den Parser weitergeleitet. Dieser wandelt die Datensätze in Datenbankdaten um und führt die Validierung durch, wo die Daten auf Korrektheit geprüft werden. Zum Schluss wird noch die Methode *Renew* aufgerufen, welche veranlasst, dass das NorthPanel neu aufgebaut wird, um damit eine Aktualisierung des Panels zu erreichen.

Die Methode *DeleteEntry* kümmert sich darum das ein Datensatz aus der Datenbank entfernt wird. Zunächst erfolgt eine Sicherheitsabfrage, ob der ausgewählte Datensatz auch wirklich gelöscht werden soll. Bestätigt der Anwender dies, wird das Löschen veranlasst. Dazu wird der BibKey des zu löschenden Datensatzes ermittelt und den Methoden *delFromAll* und *delFile* aus der Klasse *Indizes* übergeben. Diese Methoden sorgen dafür, dass aus den entsprechenden Listen, welche für die Combobox *keys* verwendet wird entfernt werden. Anschließend wird die Datenbank veranlasst den Datensatz aus der Datenbank zu entfernen. Wurde der Datensatz aus der Datenbank entfernt, wird die Datenbank über den Marshaller dazu aufgefordert, die Datensätze in der entsprechenden Datenbankdatei zu überarbeiten. Das heißt die Datenbank auf der Festplatte wird gelöscht und die Datensätze neu aufgebaut. Anschließend wird die Methode *Renew* der Datenbank aufgerufen und mit der Methode *cbEntriesChanged* veranlasst, die Combobox *keys* aktualisiert wird.

Die Methode *updateList* prüft zunächst ob eine Publikation vorhanden ist bzw. ausgewählt ist. Ist dies der Fall wird der BibKey des ausgewählten Eintrages ermittelt und das aktualisieren der Combobox *keys* mithilfe der Methode *cbEntriesChanged* veranlasst. War ein BibKey zu ermitteln wird zum Schluss die Selektion der Combobox *keys* auf diesen Eintrag gesetzt.

Die Methode *cbEntriesChanged* dient zum Erneuern der Liste in der Combobox *keys*. Die Combobox *keys* enthält die aufrufbaren Publikationen je nach Auswahl des Publikationstypen in der Combobox *Entries*.

Die Methode *addNorthPanelListener* dient anderen Klassen bzw. Methoden dazu sich in den NorthPanelListener einzutragen, um bei entsprechenden Events informiert zu werden.

Die Methode *removeNorthPanelListener* dient anderen Klassen bzw. Methoden dazu sich aus dem NorthPanelListener auszutragen.

Soll ein NorthPanelEvent ausgelöst werden, so muss die Methode *notifyAdvertisment* aufgerufen werden. Diese geht dann die Liste der Klassen bzw. Methoden durch welche sich am NorthPanelListener angemeldet haben und informiert diese, dass ein NorthPanelEvent ausgelöst wurde.

Mithilfe der Methode *initFieldsPanel* kann der NorthPanelEvent *KeyChanged* ausgelöst werden. Dazu wird der ausgewählte Publikationstyp und die ausgewählte Publikation ermittelt und dem Event hinzugefügt.

Über die Methode *Renew* können externe Methoden das Aktualisieren der Liste mit den Publikationen (keys) veranlassen. Dazu wird die Methode *updateLists* aufgerufen.

Die Methode *keyPressed* reagiert auf einen Tastendruck im Editierfeld des Filters. Dabei wird geprüft ob ein Text eingegeben ist und die Taste BACKSPACE gedrückt wurde. Wurde die Taste BACKSAPCE gedrückt und ist nur noch ein Buchstabe vorhanden, so wird der Filtertext gelöscht und die Methode *cbEntriesChanged* ausgeführt.

Die Methode *keyReleased* reagiert auf das loslassen einer Taste im Editierfeld des Filters. Dabei wird geprüft, ob sich der Fitlertext geändert hat. Hat er sich geändert wird *cbEntriesChanged* aufgerufen, um die Auswahlliste der Publikationen zu aktualisieren. Bei eingeschaltetem Filter veranlasst die Methode die Filterung der Publikationen bevor diese in die Liste eingetragen werden.

Die Methode *updateFilter* kann von externen Methoden dazu verwendet werden um, bei eingeschalteten Filter einen Neuaufbau der Auswahlliste der Publikationen zu veranlassen.

3.2.3.3. Interface NorthPanelListener

Das Interface *NorthPanelListener* wird von den Klassen eingebunden, welche auf NorthPanelEvents hören. Es ist von *EventListener* abgeleitet und führt die Methode *advertisement* ein. Diese Methode wird in den entsprechenden Klassen ausgeführt, wenn ein *NorthPanelEvent* ausgelöst wurde und die Klassen sich am Listener angemeldet haben.

3.2.3.4. Klasse NorthPanelEvent

Die Klasse *NorthPanelEvent*, wird dazu genutzt, um bei einem auslösen eines *NorthPanelEvents* ein Objekt dieser Klasse zu erzeugen. Die Klasse ist von *EventObject* abgeleitet und erweitert diese Klasse, um die Informationen des Publikationstyps und des Index der Publikation.

3.2.3.5. Klasse CenterPanel

Die Klasse *CenterPanel* erzeugt das CenterPanel und dient zur Anzeige der Informationen zu einer ausgewählten Publikation. Die Klasse ist von *JPanel* abgeleitet und implementiert den *NorthPanelListener*, um auf *NorthPanelEvents* reagieren zu können (z.B. Änderung der Publikationsauswahl).

Die Klasse teilt das CenterPanel nochmal in zwei Panel auf. Das linke Panel (fieldsLeft) und das rechte Panel (fieldsRight). Im linken Panel werden die Labels zu den Feldern eingefügt und im rechten Panel die dazugehörigen Informationen der ausgewählten Publikation. Welcher Felder angezeigt wird entscheidet sich über den Publikationstyp der ausgewählten Publikation.

Damit immer die aktuellen Daten zur jeweilig ausgewählten Publikation angezeigt werden, hört die Klasse auf *NorthPanelEvents*. Wird solch ein Event augelöst wird die Methode *advertisement* aufgerufen. Dabei prüft die Methode ob der Event mit Kommando *KeyChanged* ausgelöst wurde. Ist dies der Fall veranlasst die Methode das Löschen aller Elemente in den Panels *fieldsLeft*, *fieldsRight* und *fields*. Somit ist das CenterPanel erst einmal leer. Anschließend fragt die Methode beim Event den BibKey ab. Wurde keiner mit dem Event mitgeliefert, so bleibt das CenterPanel leer. Wurde ein BibKey mitgeliefert so wird der Publikationstyp der Publikation abgefragt. Entsprechend des Publikationstyps werden die Panels *fieldsRight* und *fieldsLeft* über eine weitere Methode initialisiert und mit den Daten der Publikation befüllt. Zum Schluss werden die Panels dem Panel *fields* übergeben und das Panel *fields* wird veranlasst sich neu aufzubauen.

3.2.3.6. Klasse SouthPanel

Die Klasse *SouthPanel* erzeugt das SouthPanel. Es werden Methoden bereitgestellt welche die Initialisierung der Elemente des SouthPanels sicherstellen und deren Aktualität in Hinsicht auf deren Informationen zur ausgewählten Publikation. Das SouthPanel ist von *JPanel* abgeleitet.

Die Klasse hört auf *NorthPanelEvents*. Wird ein solcher Event ausgelöst. wird die Methode *advertisement* aufgerufen. Wurde dabei ein *NorthPanelEvent* mit Kommando *KeyChanged* ausgelöst, veranlasst die Methode das Löschen aller Elemente im Panel *fields*. Somit ist das SouthPanel erst einmal leer. Anschließend fragt die Methode beim Event den BibKey ab. Wurde keiner mit dem Event mitgeliefert, so bleibt das SouthPanel leer. Wurde ein BibKey mitgeliefert so wird der Publikationstyp der Publikation abgefragt. Entsprechend des Publikationstyps wird das Panel fields über eine weitere Methode initialisiert und mit den BibTex-Dateinamen und Pfadangaben der Publikation befüllt. Zum Schluss wird das Panel angewiesen sich neu aufzubauen.

3.2.4. Package MainMenu

Das Package *de.BibTex.GUI.MainMenu* enthält alle Klasse die das Hauptmenü repräsentieren.

3.2.4.1. Klasse MainMenu

Die Klasse *MainMenu* stellt das Hauptmenü des Programmes zur Verfügung. Es bindet dazu Instanzen der Klassen *BibTexMenü* und *DatenbankMenü* ein.

Die Methode *getMenu* übergibt der aufrufenden Methode das Hauptmenü.

3.2.4.2. Klasse DatabaseMenu

Die Klasse *DatabaseMenu* stellt das Untermenü für die Datenbankfunktionen dem Hauptmenü zur Verfügung. Die Klasse ist von *JMenu* abgeleitet und implementiert den *ActionListener*.

Die Methode *actionPerformed* wird ausgeführt, wenn ein Menüeintrag ausgewählt wurde. Je nachdem welcher Menüeintrag ausgewählt wurde wird die jeweilige Methode (*newDatabaseAction*, *openDatabaseAction*, *closeWindow*) ausgeführt.

Die Methode *newDatabaseAction* legt eine neue Datenbankdatei an. Dazu öffnet sie ein Auswahlfenster bei der das Verzeichnis gewählt werden kann, in welches die Datei erzeugt werden soll. Des Weiteren wird über das Fenster der Dateiname angegeben. Wurde ein Dateiname übergeben, so erzeugt die Methode ein File-Objekt und übergibt dieses der Database-Methode *newDatabaseFile*. Diese Methode erstellt nun die Datenbankdatei.

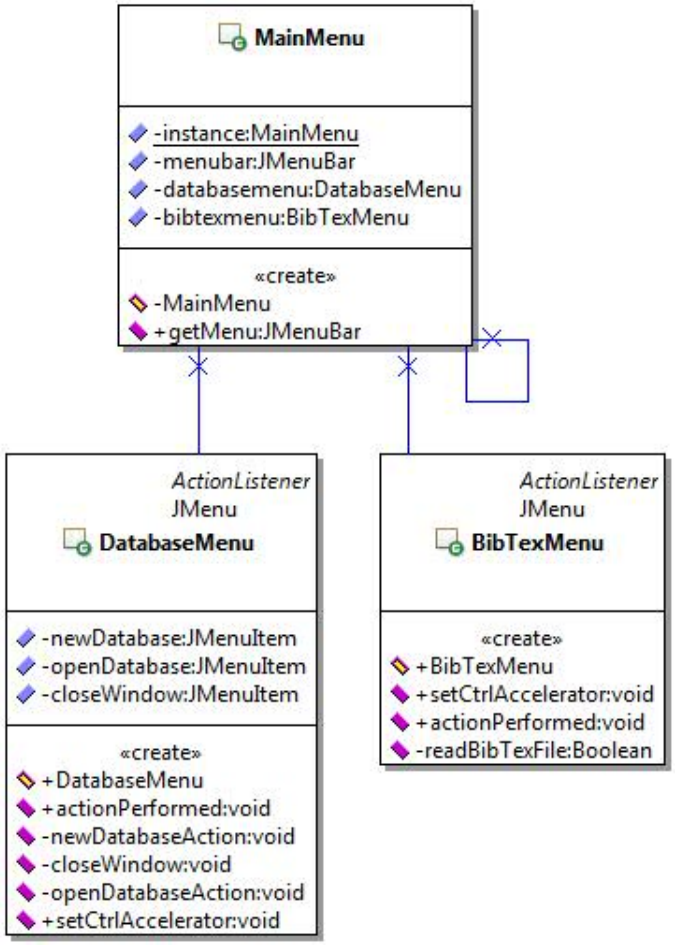

Abbildung 4: Klassendiagramm Package *de.BibTexMaker.GUI.MainMenu*

Die Methode *openDatabaseAction* öffnet eine Datenbankdatei. Dazu öffnet sie ein Auswahlfenster mit dessen Hilfe auf einem Speichermedium eine Datenbankdatei ausgewählt werden kann. Wurde eine Datei ausgewählt, so wird diese der Datenbank übergeben, so dass die Datenbank die Daten laden kann.

Die Methode *closeWindow* veranlasst über die Methode *close* der Klasse *GUI*, dass das Programm beendet wird.

3.2.4.3. Klasse BibTexMenu

Die Klasse *BibTexMenu* stellt das Untermenü für die BibTex-Funktionen dem Hauptmenü zur Verfügung. Die Klasse ist von *JMenu* abgeleitet und implementiert den *ActionListener*.

Die Methode *actionPerformed* wird ausgeführt, wenn ein Menüeintrag ausgewählt wurde. Die Methode prüft welches Kommando übergeben wurde. Wurde das Kommando *create* übergeben, so wird eine Instanz der Klasse *createWindow* erzeugt und das Fenster geöffnet. Wurde jedoch das Kommando *read* übergeben, so wird die Methode *readBibTexFile* ausgeführt und anschließend das NorthPanel angewiesen sich neu aufzubauen, um sich dadurch zu aktualisieren.

Die Methode *readBibtexFile* öffnet ein Auswahlfenster mit der BibTex-Dateien auf einem Speichermedium ausgewählt werden können. Diese Auswahl wird dann an den Parser weitergeleitet, welcher die Daten der BibTex-Dateien in die Datenbank aufnimmt. Vorher wird noch geprüft, ob eine der ausgewählten BibTex-Dateien schon in der Datenbank enthalten ist. Ist dies der Fall wird der gesamte Vorgang mit einer entsprechenden Meldung abgebrochen.

3.2.5. Package Filter

Das Package *de.BibTexMaker.GUI.Filter* enthält zwei Klassen. Die Klasse *Filter* und die Klasse *FilterWindow*.

3.2.5.1. Klasse Filter

Die Klasse Filter stellt Methoden zur Filterung bereit. Dabei werden die Publikationen gemäß den Filtereinstellungen gefiltert. Entsprechend wird die Auswahlliste der Publikationen erstellt.

Die Methode *getFields* gibt die Liste der Felder zurück mit den Informationen welches Feld in den Filter einzubeziehen ist.

Die Klasse beinhaltet zwei Methoden *setFields*, welche sich in den Parametern und des beinhaltenden Algorithmus unterscheiden.

Mit der Methode *setFields* kann über den Parameter *fields* die komplette Liste der Felder die in den Filter einzubeziehen sind überschrieben werden.

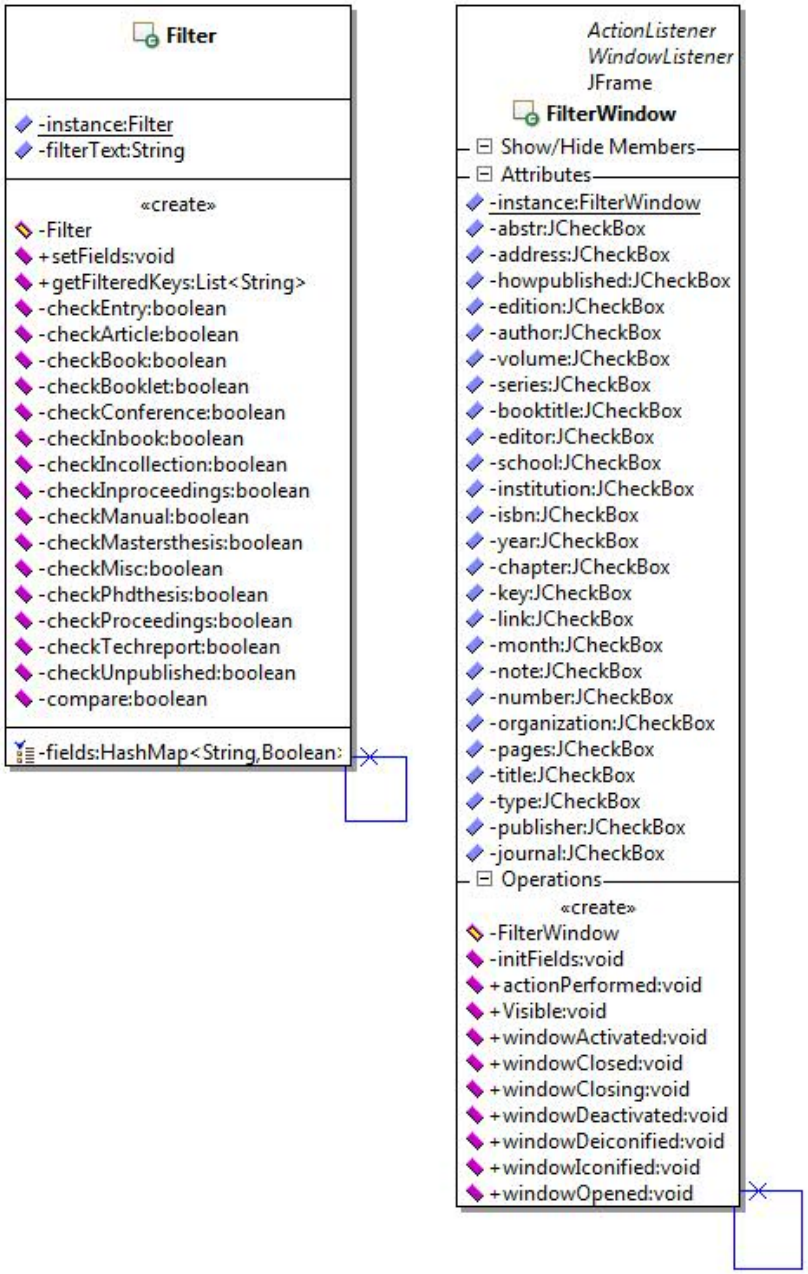

Abbildung 5: Klassendiagramm Package *de.BibTexMaker.GUI.Filter*

Mit der Methode *setFields* kann über den Parameter *fields* Änderungen in der Liste der Felder welche in den Filter einzubeziehen sind vorgenommen werden.

Mit der Methode *getFilteredKeys* wird der aufrufenden Methode eine Liste der BibKeys

übergeben, welche einen Treffer in der Datenbank auf den Filter angewandt erzielen. Dazu ist eine Liste (allkeys) mit BibKeys der Methode zu übergeben. Des Weiteren muss mittels des Parameters filterText der entsprechende Filtertext übergeben werden, nach dem die BibKeys in der Datenbank zu filtern sind.

Mit der Methode *checkEntry* wird geprüft, ob es eine Übereinstimmung mit dem Filtertext und der Publikation welche mit dem Parameter *bibkey* der Methode übergeben wird. Dabei werden nur die Felder berücksichtigt, welche auch in den Filter einzubeziehen sind.

Die Methoden *checkArticle*, *checkBook* usw. untersuchen die einzelnen Daten der Publikation, welche mit dem Parameter *bibkey* der Methode übergeben wurde, gemäß den Filtereinstellungen auf Übereinstimmung mit dem Filtertext.

Die Methode *compare* vergleicht den Filtertext mit dem Text der über den Parameter *content* der Methode übergeben wird. Bei Übereinstimmung gibt die Methode true zurück, ansonsten false.

3.2.5.2. Klasse FilterWindow

Die Klasse *FilterWindow* ist von *JFrame* abgeleitet und implementiert den *ActionListener* und den *WindowListener*. Die Klasse stellt das Fenster bereit in dem der Anwender die Felder auswählen kann, welche in die Filterung einzubeziehen sind.

Mit der Methode *initFields* werden die Checkboxen gemäß der Liste der einzubeziehenden Felder initialisiert.

Die Methode *actionPerformed* wird ausgeführt, wenn ein Klick auf eine Checkbox erfolgt. Die Methode ändert entsprechend die Liste der einzubeziehenden Felder im Filter ab.

Mit der Methode *Visible* kann eine Methode das Fenster öffnen. Die Methode stellt anhand der Bildschirmauflösung die Größe des Fensters ein und positioniert es auf dem Bildschirm.

Die Methode *windowClosing* wird ausgeführt wenn ein *WindowClosingEvent* ausgelöst wird. Die Methode ruft dann die Methode *initFields* auf.

3.2.6. Package InsertWindow

Das Package *de.BibTexMaker.GUI.InsertWindow* stellt die Klassen für das Fenster zum Hinzufügen von Publikationen zur Verfügung.

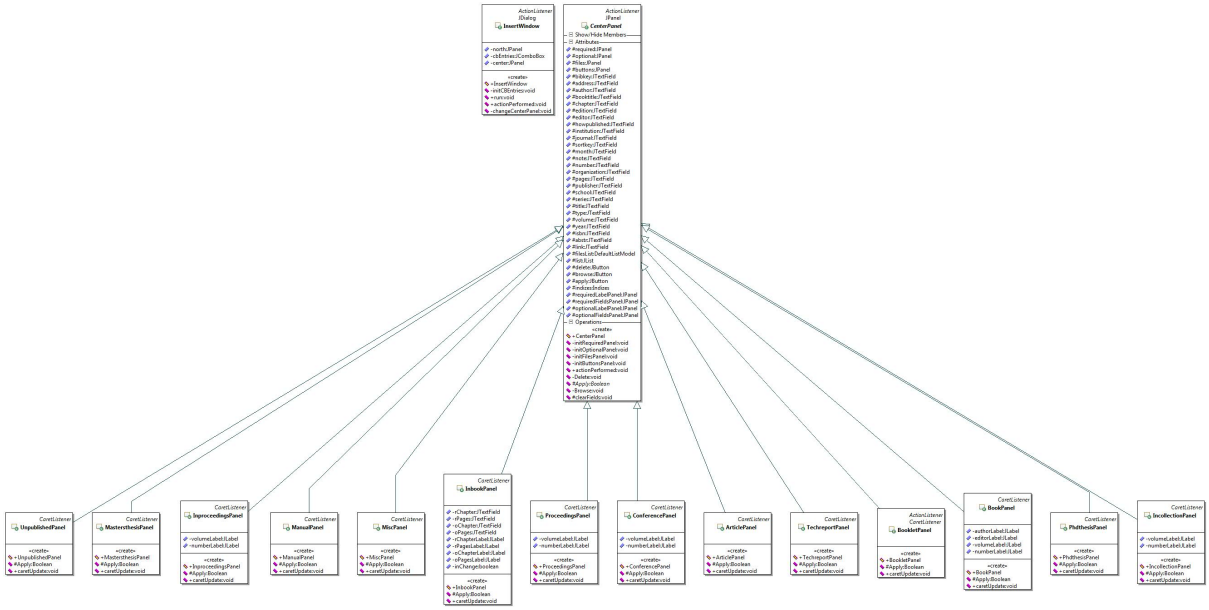

Abbildung 6: Klassendiagramm Package *de.BibTexMaker.GUI.InsertWindow*

3.2.6.1. Klasse InsertWindow

Die Klasse *InsertWindow* stellt der GUI ein Fenster zum Hinzufügen von Publikation zur Verfügung. Die Klasse ist von *JDialog* abgeleitet und implementiert den *ActionListener*.

Mit der Methode *run* wird das Fesnter aufgerufen. Die Methode stellt anhand der Bildschirmauflösung die Größe des Fensters ein und positioniert es mittig auf dem Bildschirm.

Mit der Methode initCBEntries wird die Auswahlliste der Publikationstypen initialisiert.

Die Methode *actionPerformed* reagiert auf ActionEvents. Dabei prüft es ob das Kommando *comboBoxChanged* übergeben wurde. Ist dies der Fall wird der Wechsel des CenterPanels ausgelöst. Das Kommando *comboBoxChanged* wird ausgelöst, wenn in der Auswahlliste ein anderer Publikationstyp ausgewählt wird. Da der Publikationstyp gewechselt wurde, muss auch das entsprechende Panel in das CenterPanel geladen werden.

Mit der Methode *changeCenterPanel* wird das Panel in das CenterPanel geladen, dass zu dem Publikationstyp gehört, welche in der Auswahlliste ausgewählt wurde.

3.2.6.2. Klasse CenterPanel

Die Klasse *CenterPanel* ist von *JPanel* abgeleitet und implementiert den *ActionListener*. Die Klasse wird nicht direkt von *InsertWindow* verwendet, sondern dient für die jeweiligen CenterPanels (je Publikationstyp) als Oberklasse von der abgeleitet wird.

Die Methode *initRequiredPanel* initialisiert das Panel der Pflichtfelder.

Die Methode *initOptionalPanel* initialisiert das Panel der optionalen Felder.

Die Methode *initFilesPanel* initialisiert das Panel mit der Liste der BibTex-Dateien.

Die Methode *initButtonsPanel* initialisiert das Panel mit den Schaltflächen.

Die Methode *actionPerformed* reagiert auf *ActioneEvents*. Entsprechend des übergebenen Kommandos, wird eine entsprechende Methode ausgeführt.

Wenn die Schaltfläche Entfernen betätigt wird, wird die Methode *Delete* ausgeführt. Die Methode entfernt die ausgewählten BibTex-Dateien aus der Liste der der Publikation zugeordneten BibTex-Dateien.

Die Methode *Apply* ist eine abstrakte Methode und wird ausgeführt wenn die Schaltfläche Speichern betätigt wurde. Der eigentliche Quellcode wird durch die ableitende Klasse gestellt.

Die Methode *Browse* öffnet ein Auswahlfenster mit denen man BibTex-Dateien der Liste hinzufügen kann. Sollte diese Datei noch nicht exisiteiren, wird Sie zwar nicht angelegt, aber trotzdem in die Liste aufgenommen.

Mit der Methode *clearFields*, wird in allen Editierfeldern der Inhalt gelöscht und alle Einträge in der Liste der BibTex-Dateien entfernt.

3.2.6.3. abgeleitete Klassen von der Klasse CenterPanel

Zu den abgeleiteten Klassen von der Klasse CenterPanel gehören folgende Klassen:

- ArticlePanel
- BookPanel
- BookletPanel
- ConferencePanel
- InbookPanel
- IncollectionPanel
- InproceedingsPanel
- ManualPanel
- MastersthesisPanel
- MiscPanel
- PhdthesisPanel
- ProceedingsPanel
- TechreportPanel
- UnpublishedPanel

Alle zuvor aufgezählten Klassen besitzen den gleichen Aufbau. Sie unterscheiden sich lediglich darin, welche Pflichtfelder und optionalen Felder im Panel zur Anwendung kommen. Dies ist abhängig vom Publikationstyp der hinzuzufügenden Publikation. In der Liste kann man erkennen, dass der Publikationstyp sich im Namen der jeweiligen Klasse wiederspiegelt. Im folgenden werden daher die Methode nicht explizieht für jede Klasse vorgestellt, sondern allgemein.

Der Konstruktor intialisiert die Panels mit den Labeln und Editierfeldern der Pflicht- und optionalen Felder sowie der BibTex-Datei-Liste. Dem Konstruktor ist die Breite des Panels zu übergeben.

Die Methode *Apply* implementiert den Quellcode in die abstrakte Methode der Oberklasse *CenterPanel*. Die Methode wird ausgeführt, wenn die Schaltfläche SPEICHERN betätigt wurde. Die Methode liest die Daten aus den Editierfeldern aus und übergibt diese der Datenbank.

Die Methode *caretUpdate* prüft ob alle Pflichtfelder ausgefüllt sind. Ist dies der Fall so wird die Schaltfläche SPEICHERN freigegeben, ansonsten gesperrt. Die Methode wird aufgerufen, wenn ein *CaretEvent* ausgelöst wird. Der *CaretEvent* wird ausgelöst, wenn sich die Cursor-Position in einem der Editierfelder der Pflichtfelder ändert.

3.2.7. Package EditWindow

Das Package *de.BibTexMaker.GUI.EditWindow* stellt die Klassen für das Fenster zum Bearbeiten einer Publikation zur Verfügung.

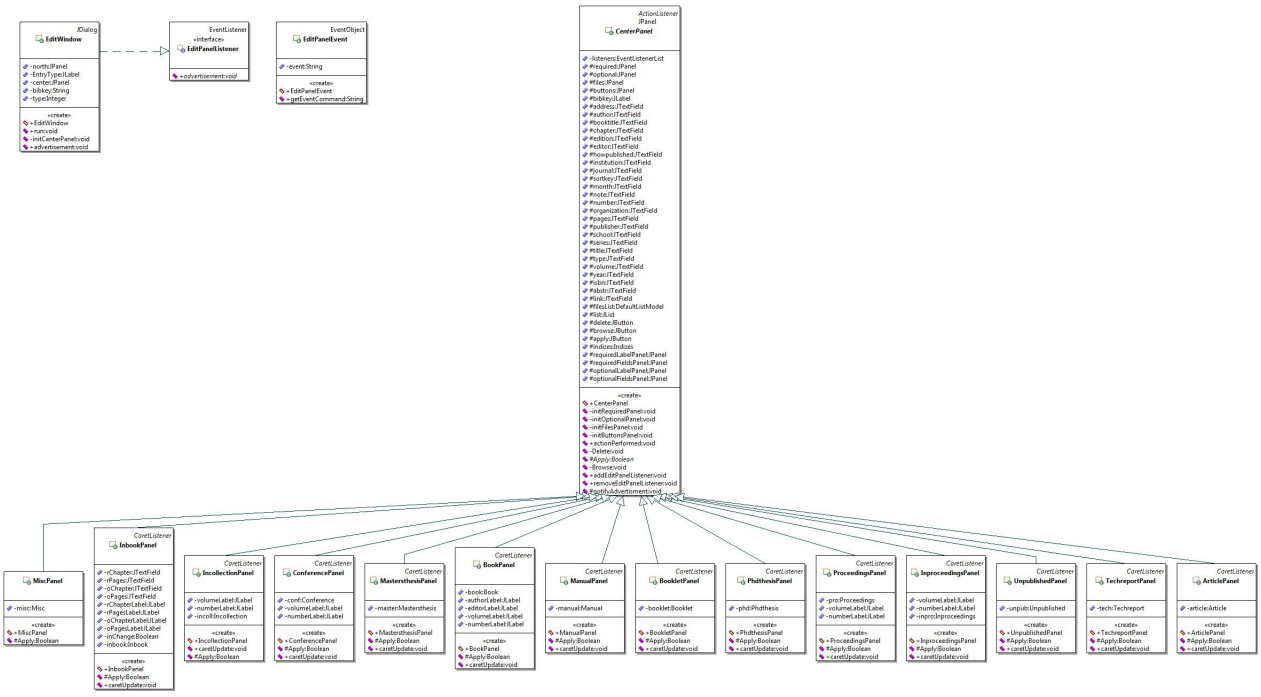

Abbildung 7: Klassendiagramm Package *de.BibTexMaker.GUI.EditWindow*

3.2.7.1. Klasse EditWindow

Die Klasse *EditWindow* stellt der GUI ein Fenster zur Bearbeitung der Informationen zu einer Publikation zur Verfügung. Die Klasse ist von *JDialog* abgeleitet und implementiert den *EditPanelListener*.

Der Konstruktor initialisert das Bearbeitungsfenster. Dazu ist ihm über die Paramter *bibkey* und *type* der Index und der Typ der Publikation zu übergeben.

Mit der Methode *run* wird das Bearbeitungsfenster aufgerufen. Die Methode stellt anhand der Bildschirmauflösung und des jeweiligen Publikationstypen die Größe des Fensters ein und positioniert es mittig auf dem Bildschirm.

Die Methode *initCenterPanel* intialisiert das CenterPanel (center). Zunächst entfernt die Methode alle vorhandenen Elemente aus dem CenterPanel und prüft dann den Publikationstyp und lädt das entsprechende Panel in das CenterPanel. Zum Schluss wird das Centerpanel noch aufgefordert sich neu aufzubauen, damit es die Aktualisierung zur Anzeige bringt.

Die Mehode *advertisement* reagiert auf *EditPanelEvents*. Wurde ein *EditPanelEvent* mit dem Kommando *closeWindow* ausgelöst, veranlasst die Methode das Schließen des Bearbeitungsfensters.

3.2.7.2. Klasse CenterPanel

Die Klasse *CenterPanel* ist von JPanel abgeleitet und implementiert den *ActionListener*. Die Klasse wird nicht direkt von *EditWindow* verwendet, sondern dient für die jeweiligen CenterPanels (je Publikationstyp) als Oberklasse von der abgeleitet wird.

Der Konstruktor initialisiert das CenterPanel. Dem Konstruktor ist der Index (bibkey) der zu bearbeitenden Publikation zu übergeben.

Die Methode *initRequiredPanel* intialisiert das Panel mit den Pflichtfelder. Dazu wird das Layout und die Border eingestellt.

Die Methode *initOptionalPanel* intialisiert das Panel mit den optionalen Feldern. Dazu wird das Layout und die Border eingestellt.

Die Methode *initFilesPanel* intialisiert das Panel mit den BibTex-Dateien.

Die Methode *initButtonPanel* intialisiert das Panel mit den Schaltflächen.

Die Methode *actionPerformed* reagiert auf ActionEvents. Dabei prüft die Methode welche Schaltfläche betätigt wurde. Entsprechend wird eine weitere Methode aufgerufen.

Wenn die Schaltfläche Entfernen betätigt wird, wird die Methode *Delete* ausgeführt. Die Methode entfernt die ausgewählten BibTex-Dateien aus der Liste der der Publikation zugeordneten BibTex-Dateien.

Die Methode *Apply* ist eine abstrakte Methode und wird ausgeführt wenn die Schaltfläche Speichern betätigt wurde. Der eigentliche Quellcode wird durch die ableitende Klasse gestellt.

Die Methode *Browse* öffnet ein Auswahlfenster mit denen man BibTex-Dateien der Liste hinzufügen kann. Sollte diese Datei noch nicht exisiteiren, wird Sie zwar nicht angelegt, aber trotzdem in die Liste aufgenommen.

Die Methode *addEditPanelListener* dient anderen Klassen bzw. Methoden dazu sich an den EditPanelListener einzutragen, um bei entsprechenden Events informiert zu werden.

Die Methode *removeEditPanelListener* dient anderen Klassen bzw. Methoden dazu sich aus den EditPanelListener auszutragen.

Soll ein EditPanelEvent ausgelöst werden, so muss die Methode *notifyAdvertisment* aufgerufen werden. Diese geht dann die Liste der Klassen bzw. Methoden durch welche sich am *EditPanelListener* angemeldet haben und informiert diese, dass ein *EditPanelEvent* ausgelöst wurde.

3.2.7.3. abgeleitete Klassen von der Klasse CenterPanel

Zu den abgeleiteten Klassen von der Klasse CenterPanel gehören folgende Klassen:

- ArticlePanel
- BookletPanel
- BookPanel
- ConferencePanel
- InbookPanel
- IncollectionPanel
- InproceedingsPanel
- ManualPanel
- MastersthesisPanel
- MiscPanel
- PhdthesisPanel
- ProceedingsPanel
- TechreportPanel
- UnpublishedPanel

Alle zuvor aufgezählten Klassen besitzen den gleichen Aufbau. Sie unterscheiden sich lediglich darin, welche Pflichtfelder und optionalen Felder im Panel zur Anwendung kommen. Dies ist abhängig vom Publikationstyp der hinzuzufügenden Publikation. In der Liste kann man erkennen, dass der Publikationstyp sich im Namen der jeweiligen Klasse wiederspiegelt. Im folgenden werden daher die Methode nicht explizieht für jede Klasse vorgestellt, sondern

allgemein.

Der Konstruktor befüllt die in CenterPanel intialisierten Panels mit den Labeln und Editierfeldern der Pflicht- und optionalen Felder sowie der BibTex-Datei-Liste. Dem Konstruktor ist der Index der Publikation und die Breite des Panels zu übergeben.

Die Methode *Apply* implementiert den Quellcode in die abstrakte Methode der Oberklasse *CenterPanel*. Die Methode wird ausgeführt, wenn die Schaltfläche SPEICHERN betätigt wurde. Die Methode liest die Daten aus den Editierfeldern aus und übergibt diese der Datenbank.

Die Methode *caretUpdate* prüft ob alle Pflichtfelder ausgefüllt sind. Ist dies der Fall so wird die Schaltfläche SPEICHERN freigegeben, ansonsten gesperrt. Die Methode wird aufgerufen, wenn ein *CaretEvent* ausgelöst wird. Der *CaretEvent* wird ausgelöst, wenn sich die Cursor-Position in einem der Pflichtfelder ändert.

3.2.7.4. Klasse EditPanelListener

Das Interface *EditPanelListener* wird von den Klassen eingebunden, welche auf *EditPanelEvents* hören. Es ist von *EventListener* abgeleitet und führt die Methode *advertisement* ein. Diese Methode wird in den entsprechenden Klassen ausgeführt, wenn ein *EditPanelEvent* ausgelöst wurde und die Klassen sich am Listener angemeldet haben.

3.2.7.5. Klasse EditPanelEvent

Die Klasse *EditPanelEvent*, wird dazu genutzt, um bei einem auslösen eines *EditPanelEvents* ein Objekt dieser Klasse zu erzeugen. Die Klasse ist von *EventObject* abgeleitet.

Die Methode *getEventCommand* gibt der aufrufenden Methode das Kommando des Events zurück.

3.2.8. Package createWindow

Das Package *de.BibTexMaker.GUI.createWindow* stellt eine Klasse mit entsprechenden Methoden der GUI zur Verfügung, welche ein Fenster erzeuegen, dass eine Auswahl der BibTex-Dateien bereitstellt. Aus dieser Liste kann dann der Anwender eine BibTex-Datei auswählen, welche erstellt werden soll.

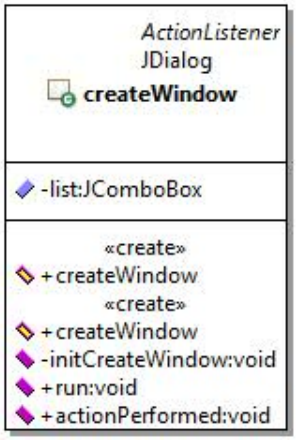

Abbildung 8: Klassendiagramm Package *de.BibTexMaker.GUI.createWindow*

3.2.8.1. Klasse createWindow

Die Klasse *createWindow* stellt einen Dialog zum erstellen einer BibTex-Datei bereit. Die Klasse ist von *JDialog* abgeleitet.

Ein Objekt der Klasse kann über zwei verschiedene Konstruktoren initialisiert werden.

Dem ersten Konstruktor sind keine Parameter zu übergeben. Er führt die Methode *initCreateWindow* aus, welche den Dialog initialisiert.

Dem zweiten Konstruktor ist mittels des Parameters *file* der Dateiname der BibTex-Datei zu übergeben, welche nach der Initialisierung in der Auswahlliste selektiert, also vorausgewählt, sein soll. Dazu führt der Konstruktor zunächst die Methode *initCreateWindow* aus, welche den Dialog initialisiert. Anschließend wird die entsprechende BibTex-Datei in der Auswahlliste selektiert.

Mit der Methode *initCreateWindow* wird der Dialog initialisert.

Mit der Methode *run* wird der Dialog aufgerufen. Die Methode stellt dabei die Größe des Dialogfensters ein und positioniert das Fenster auf dem Bildschirm.

Die Methode *actionPerformed* reagiert auf *ActionEvents*. Die Methode prüft dann ob die Schaltfläche ERSTELLEN betätigt wurde. Ist dies der Fall wird der Serialisierer aufgerufen und die BibTex-Datei übergeben, welche erzeugt werden soll. Zum Schluss wird der createWindow-Dialog geschlossen.

3.3. Package de.BibTexMaker.Database

Das Package *de.BibTexMaker.Database* beinhaltet alle Klassen und Packages der Datenbank.

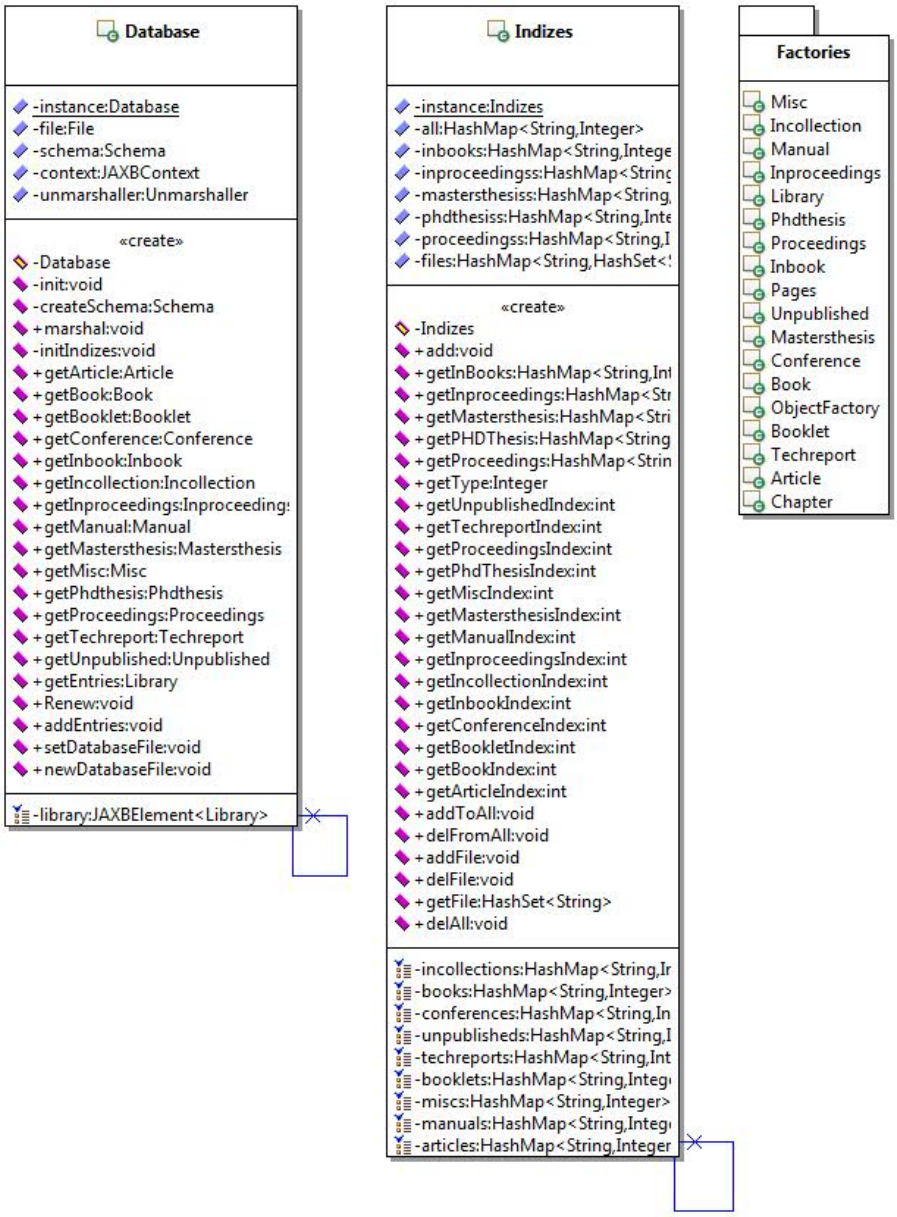

Abbildung 9: Klassendiagramm Package *de.BibTexMaker.Database*

3.3.1. Klasse Database

Die Klasse *Database* nutzt die Programmschnittstelle *Java Architecture for XML Binding (JAXB)* in der Version 2.1. Damit wird ermöglicht Daten aus einer XML-Datei automatish an Java-Klassen zu binden und aus einem XML-Schema heraus zu generieren. Weiterhin stellt die Datenbank Methoden zum Marshallen und Unmarshallen zur Verfügung. Sowohl beim Marshallen als auch beim Unmarshallen werden die Daten validiert und auf ihre Korrektheit und Konsistenz geprüft. Es werden auch Getter und Setter Methoden bereit gestellt, um Daten in die Datenbank zu schreiben bzw. aus der Datenbank zu lesen. Für die diversen Zugriffe auf die Daten werden Indizierungen bereitgestellt. Besser gesagt die Indizierungen werden hier erstellt und über die Klasse *Indizes* können diese abgefragt werden.

Die Klasse ist nach dem Prinzip des Singleton-Pattern geschrieben. Dies dient dazu sicherzustellen, dass innerhalb des Programms nur eine Instanz der Klasse existiert, so dass immer auf die Richtige Instanz zugegriffen wird. Der Zugriff erfolgt über die Methode *getInstance*. Diese prüft ob bereits eine Instanz der Klasse angelegt wurde und legt eine an, wenn dies nicht der Fall ist. Anschließend gibt die Methode die Instanz an die aufrufende Methode zurück.

Über die Methode *init* wird der Datenbank mittels des Parameters *file* die Datenbankdatei (XML-Datei) übergeben. Daraufhin wird JAXB initialisiert und die Daten der XML-Datei an die entsprechenden Java-Klassen aus dem Package *Factories* gebunden.

Die Methode *Schema* lädt das XML-Schema für die BibTexMaker-Datenbank und gibt dies der aufrufenden Methode zurück.

Die Methode *marshal* wird aufgerufen, um die Daten aus den Java Objekten in eine XML-Datei zu generieren.

Die Methode *initIndizes* legt Listen der einzelnen Publikationen zu den entsprechenden Publikationstypen an. Diese Listen werden z.B. von der GUI für die Auswahllisten der Publikationen verwendet.

Die verschiedenen Getter-Methode zu den Publikationstypen (z.B. getArticle, getBook usw.) geben die Daten zu einer über den Parameter *bibkey* bestimmten Publikation an die aufrufende Methode zurück.

Über die Methode *getLibrary* kann die gesamte Datenbank abgefragt werden.

Die Methode *getEntries* übergibt der aufrufenden Methode alle Datensätze der Datenbank.

Mit der Methode *Renew* kann eine externe Methode veranlassen, dass die Indizierungen neu erstellt werden.

Um Publikationen in die Datenbank aufzunehmen, muss die Methode *addEntries* aufgerufen werden. Über den Parameter *entries* sind dabei die Datensätze der Methode zu übergeben.

Soll die Datenbank gewechselt werden, dann ist die Methode *setDatabaseFile* aufzurufen. Über dem Parameter *file* ist die zu ladende Datenbankdatei zu übergeben.

Mithilfe der Methode *newDatabaseFile* kann eine neue Datenbankdatei angelegt werden. Der Methode ist über den Parameter *file* die Datenbankdatei zu übergeben.

3.3.1.1. Klasse Indizes

Die Klasse *Indizes* beinhaltet Methode um Indizierungslisten zu erstellen und zu bearbeiten. Die Indizierungslisten werden von der GUI verwendet, um damit die Auswahlliste der Publikationen zu befüllen. Die Klasse wurde nach dem Prinzip des Singleton-Pattern geschrieben, um sicherzustellen, dass immer auf die aktuellen Indizierungslisten zugegriffen wird.

Die Methode *getInstance* übergibt die einzige Instanz der Klasse *Indizes* an die aufrufende Methode. Dabei wird vorher geprüft ob die Instanz schon angelegt wurde. Ist dies nicht der Fall, so wird dies noch erledigt.

Über die Methode *add* können einer Indizierungsliste eines bestimmten Publikationstypen weitere Einträge hinzugefügt werden.

Die Methode getAll gibt der aufrufenden Methode die Indizierungsliste aller Publikationstypen zurück.

Die Getter-Methode zu den Publikationstypen (z.B. getArticles, getBooks, usw.) geben der aufrufenden Methode die Indizierungsliste des entsprechenden Publikationstypen zurück.

Mit der Methode *getType* kann aus der Indizierungsliste aller Publikationen der Publikationstyp extrahiert werden. Als Parameter muss der entsprechende BibKey der Publikation übergeben werden.

Die Getter-Methoden zu den Publikationen (z.B. getTechreportIndex, getUnpublishedIndex, usw.) geben der aufrufenden Methode den Index einer bestimmten Publikation zurück. Die Publikation wird über den Parameter bibkey referenziert.

Mit der Methode *addToAll* kann der Indizierungsliste aller Publikationen ein neuer Eintrag hinzugefügt werden.

Mit der Methode *delFromAll* kann ein Eintrag aus der Indizierungsliste aller Publikationenen entfernt werden.

Mit der Methode *addFile* kann der Liste der Zuordnungen von BibTex-Dateien und Publikationen ein neuer Eintrag hinzugefügt werden.

Mit der Methode *delFile* kann aus der Liste der Zuordnungen von BibTex-Dateien und Publikationen ein Eintrag entfernt werden.

Die Methode *getFile* gibt der aufrufenden Methode eine Liste der Publikationen zurück, welche einer bestimmten BibTex-Datei zugeordnet sind. Die Bibtex-Datei wird mit dem Parameter *filename* der Methode übergeben.

Die Methode *getFiles* gibt eine Liste aller in der Datenbank eingetragenen BibTex-Dateien zurück.

Mit der Methode *delAll* können alle Einträge aus allen Indizierungslisten entfernt werden.

3.3.1.2. Packages de.BibTexMAker.Database.Factories

Das Package *Factories* enthält die Klassen, welche die Java-Objekte der XML-Elemente entsprechen. Sie dienen als Schnittstelle zwischen XML und Java.

Die Klassen sind nach dem Prinzip des Factory Pattern geschrieben und wurden über das Framework JAXB automatisch erstellt. Dazu wurde eine XML-Schemata-Datei erstellt und diese dem ANT Compiler mittels einer build.xml-Datei übergeben. In der build.xml stehen die Verarbeitungsregeln für den ANT Compiler und in der XML-Schemata-Datei (library.xsd) die Regeln zu den XML-Elementen. Aus diesen Dateien und deren darin befindlichen Angaben hat der ANT Compiler die im Package *de.BibTexMAker.Database.Factorie* enthaltenen Klassen erstellt.

Aufgrund der automatischen Erzeugung der Klassen und der Nutzung des Frameworks JAXB wird hier auf eine Dokumentation der einzelnen Klassen verzichtet und auf die Kommentare in den Klassen bzw. den diversen Dokumentationen zu JAXB verwiesen.

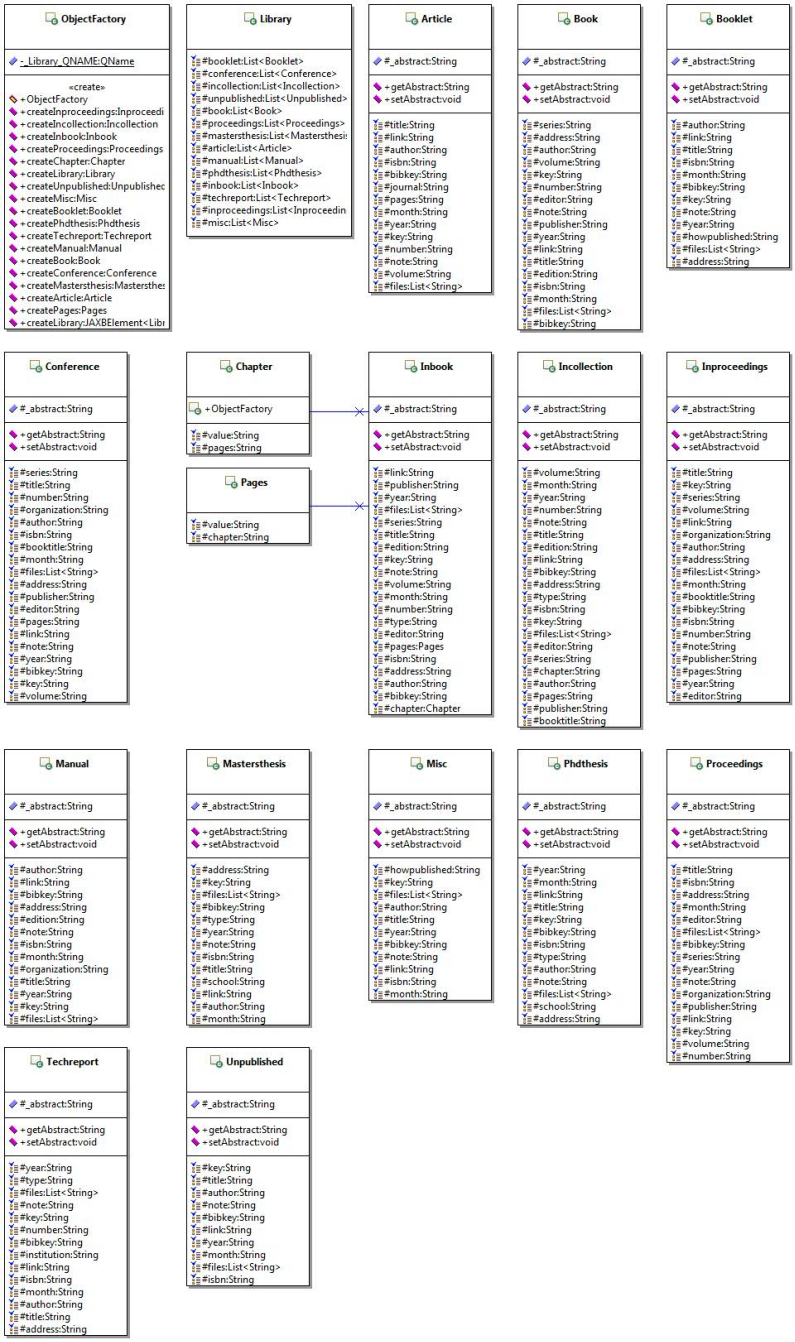

Abbildung 10: Klassendiagramm Package *de.BibTexMaker.Database.Factories*

3.4. Package de.BibTexMaker.BibTex

Das Package *de.BibTexMaker.Bibtex* beinhaltet die Klassen *Parser*, *Serializer* und *Validator*.

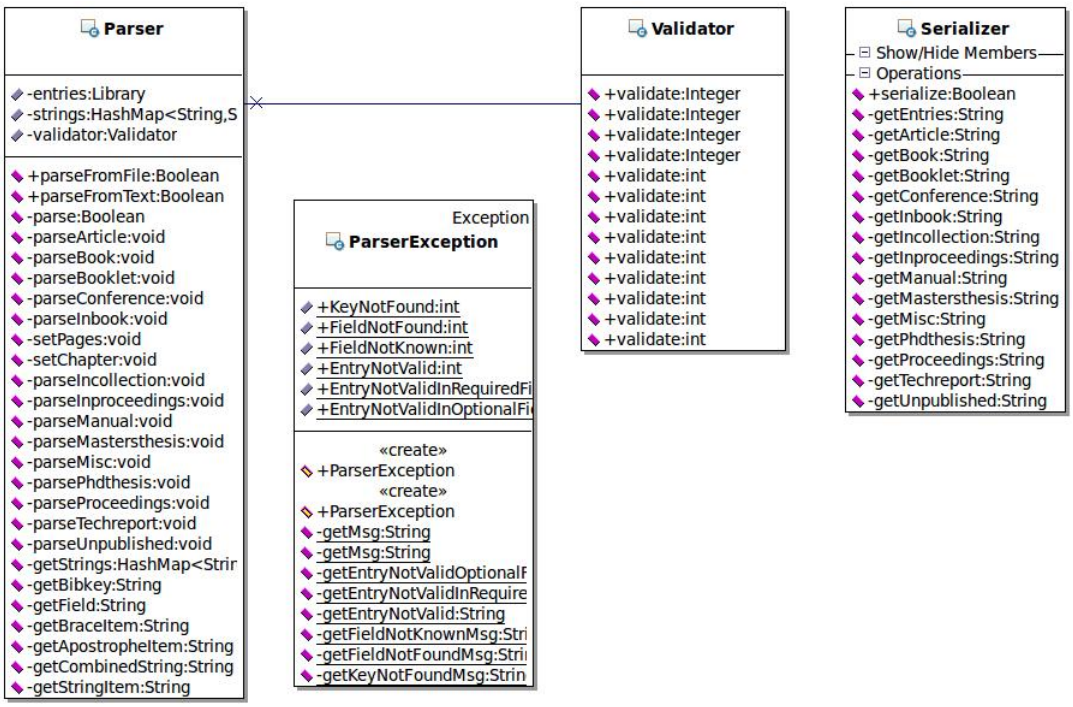

Abbildung 11: Klassendiagramm Package *de.BibTexMaker.BibTex*

Diese Klassen sind dafür zuständig den Datenaustausch zwischen BibTeX-Dateien und der BibTeXMaker-Datenbank sicherzustellen. Dabei sind die Daten in die entsprechenden Formate zu wandeln und beim Parsen zu validieren.

3.4.1. Klasse Parser

Die Klasse *Parser* stellt einen Parser zur Verfügung, welcher Datensätze aus BibTeX-Dateien in Datensätze der der Datenbank Database umwandelt. Bei dieser Umwandlung werden die Daten validiert und somit auf die Einhaltung der BibTeX-Regeln geprüft. Findet der Parser bzw. Validator einen Fehler wird das Parsen abgebrochen und eine entsprechende Exception ausgelöst.

Mit der Methode *parseFromFile* wird das Parsen einer Datei ausgelöst. Die zu parsende Datei ist über den Parameter *file* zu übergeben. War das Parsen erfolgreich wird der aufrufenden Methode true zurückgegeben, ansonsten false.

Mit der Methode *parseFromText* wird das Parsen eines Strings ausgelöst. Der zu parsende String ist über den Parameter *text* zu übergeben. War das Parsen erfolgreich wird der aufrufenden Methode true zurückgegeben, ansonsten false.

Die Methode *parse* wandelt BibTeX-Datensätze in Datensätze der BibTeXMaker-Datenbank um. Die BibTeX-Datensätze sind der Methode über den Parameter *text* zu übergeben. Des Weiteren ist über den Parameter *filename* die Pfad- und Dateinamenangabe zur BibTeX-Datei zu übergeben, aus der die Daten stammen. Stammen die Daten aus keiner Datei, so ist der Parameter *filename* mit einem Null-String zu übergeben.

Die Methoden zu den jeweiligen Publikationstypen, wie z.B. *parseArticle*, extrahieren aus dem String *bibtex*, welcher als Parameter der Methode übergeben wurde, die Daten der Pflichtfelder und der optionalen Felder. Die Daten werden dann in ein neues Objekt zum Publikationstyp gehörenden Klasse (z.B. *Article*) geschrieben. Dieses Objekt wird dann dem Validator zugeführt, welcher prüft, ob alle Pflichtfelder befüllt sind und alle Regeln für den entsprechenden Publikationstyp eingehalten wurden. Ist dies der Fall, so wird der Datensatz in das Attribut entries übernommen.

Die Methode *getStrings* sucht nach Abkürzungen in dem mit dem Paramter *text* übergebenen String. Die Methode baut aus den gefundenen Abkürzungen über eine HashMap ein Abkürzungsverzeichnis auf, welches die Methode am Ende zurückgibt.

Die Methode *getBibkey* extrahiert aus dem mit dem Parameter *bibtex* übergebenen String den BibKey der entsprechenden Publikation und gibt ihn der aufrufenden Methode zurück.

Mit der Methode *getField* kann aus dem String, der mit dem Parameter *bibtex* übergeben wurde, das Datum zum jeweiligen Feld, welches über den Parameter *field* übergeben wurde, ermittelt werden.

Die Methode *getBrace* wird von der Methode *parse* eingesetzt, um Daten eines Feldes zu ermitteln welche in geschweifte Klammern gesetzt ist.

Die Methode *getApostropheItem* wird von der Methode *parse* eingesetzt, um Daten eines Feldes zu ermitteln, welches in Anführungszeichen gesetzt wurde.

Die Methode *getCombinedString* wird von der Methode *getApostropheItem* eingesetzt, um

die Daten eines Feldes zu ermitteln, welche aus Kombinationen von Abkürzungen und in Anführungszeichen gesetzte Daten bestehen.

Die Methode *getStringItem* wird dazu eingesetzt, um aus dem String *tmp* der als Parameter übergeben wurde die Abkürzung zu extrahieren und mittels des erstellten Abkürzungsverzeichnisses (strings) zu übersetzen.

3.4.2. Klasse ParserException

Die Klasse *ParserException* stellt Methoden bereit, um auf Exception zu reagieren, welche vom Parser ausgelöst werden. Die Klasse ist von *Exception* abgeleitet.

Die Klasse kann über zwei Konstruktoren aufgerufen werden. Dem ersten Konstruktor ist lediglich die Fehlernummer zu übergeben und dem zweiten Konstruktor ist zusätzlich zur Fehlernummer noch eine Fehlermeldung zu übergeben.

Die Methode *getMsg* gibt die Fehlermeldung zur mit dem Parameter *errno* übergebenen Fehlermeldung zurück.

Die Methode *getMsg* gibt die Fehlermeldung zur mit dem Parameter *errno* übergebenen Fehlermeldung zurück. Konnte zur Fehlermeldung keine Fehlermeldung gefunden werden, so wird die über den Parameter *msg* übergebene Meldung zurückgegeben.

Die Methode *getEntryNotValidOptionalField* gibt entsprechend der Message, welche mit dem Parameter *msg* übergeben wird, eine Fehlermeldung zurück.

Die Methode *getEntryNotValidInRequiredField* gibt entsprechend der Message, welche mit dem Parameter *msg* übergeben wird, eine Fehlermeldung zurück.

Die Methode *getEntryNotValid* gibt entsprechend der Message, welche mit dem Parameter *msg* übergeben wird, eine Fehlermeldung zurück.

Die Methode *getFieldNotKnownMsg* gibt entsprechend der Message, welche mit dem Parameter *msg* übergeben wird, eine Fehlermeldung zurück.

Die Methode *getFieldNotFoundMsg* gibt entsprechend der Message, welche mit dem Parameter *msg* übergeben wird, eine Fehlermeldung zurück.

Die Methode *getKeyNotFoundMsg* gibt entsprechend der Message, welche mit dem Parame-

ter *msg* übergeben wird, eine Fehlermeldung zurück.

3.4.3. Klasse Serializer

Die Klasse *Serializer* stellt Methoden zur Verfügung, um Datensätze aus der BibTeXMaker-Datenbank in BibTeX-Daten zu konvertieren und diese Regelkonform in einer BibTeX-Datei abzulegen.

Mit der Methode *serialize* wird das Serialisieren gestartet. Dazu ist der Methode über den Parameter *filename* der absolute Pfad zur zu erstellenden BibTeX-Datei zu übergeben. Die Methode gibt true zurück, wenn das Serialisieren ohne Fehler abgeschlossen werden konnte, ansonsten wird false zurückgegeben.

Die Methode *getEntries* gibt einen vollständigen Datensatz im BibTeX-Format zurück. Dazu ist der Methode der entsprechende BibKey des Datensatzes zu übergeben.

Die jeweiligen Methoden zu den Publikationstypen, wie z.B. *getArticle*, wandeln einen Datensatz des entsprechenden Publikationstyps vom BibTeXMaker-Datenbank-Format in das BibTeX-Format um und giben diesen zurück. Dazu ist der Methode der Bibkey des entsprechenden Datensatzes zu übergeben.

3.4.4. Klasse Validator

Die Klasse *Validator* dient dem Parser zur Validierung der zu parsenden Daten. Der Validierer prüft allerdings lediglich das zu einem Datensatz alle Pflichfelder vorhanden sind, also es zu den entsprechenden Pflichtfelder Daten exisitieren und diese Regelkonform vorliegen. Dabei werden auch optionale Felder bei denen es eine Regel gibt in die Prüfung einbezogen.

Die Methoden *validate* validieren eine Publikation je nach Publikationstyp. Dabei wird die ID-Nummer des Feldes zurückgegeben welches nicht valide ist. Sind alle Felder valide, so wird eine 0 zurückgegeben.

3.5. Package de.BibTexMaker.Utiles

Das Package *de.BibTexMaker.Utiles* beinhaltet Hilfsklassen deren Methode meist statisch sind, da Sie zur Unterstützung der anderen Klassen dienen.

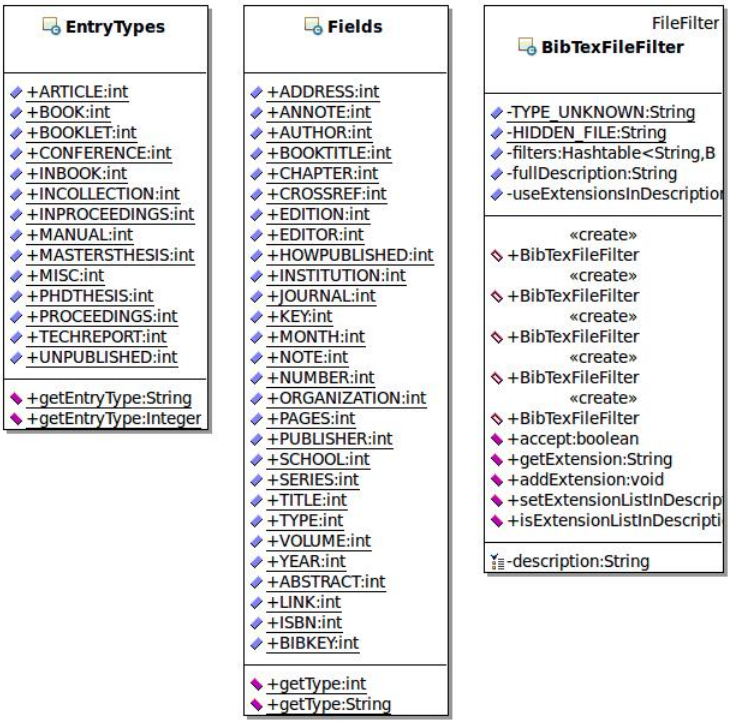

Abbildung 12: Klassendiagramm Package *de.BibTexMaker.Utiles*

3.5.1. Klasse EntryTypes

Die Klasse *EntryTypes* stellt statische Methoden zur Verfügung, um über die ID-Nummer eines Publikationstypen den Titel von diesem abzufragen und umgekehrt.

Es gibt zwei Methode *getEntryType*. Die Eine erwartet als Parameter die ID-Nummer des Publikationstyps dessen Bezeichnung zurückgeliefert werden soll. Bei der zweiten Methode *getEntryType* ist es genau umgekehrt. Diese erwartet als Parameter die Bezeichnung des Publikationstyps dessen ID-Nummer zurückgegebene werden soll.

3.5.2. Klasse Fields

Die Klasse *Fields* stellt statische Methoden zur Verfügung, um über die ID-Nummer eines Feldes den Titel von diesem abzufragen und umgekehrt.

Es gibt zwei Methode *getType*. Die Eine erwartet als Parameter die ID-Nummer des Feldes dessen Bezeichnung zurückgeliefert werden soll. Bei der zweiten Methode *getType* ist es genau umgekehrt. Diese erwartet als Parameter die Bezeichnung des Feldes dessen ID-Nummer zurückgegebene werden soll.

3.5.3. Klasse BibTexFileFilter

Die Klasse *BibtexFileFilter* wird hier hauptsächlich von der GUI eingesetzt. Die Klasse stammt in der Ursprünglichen Version von *Jeff Dinkins*[1] und wurde hier nicht weiter verändert, daher werde ich hier den Originalkommentar[2] zur Klasse wiedergeben.

> A convenience implementation of FileFilter that filters out
> all files except for those type extensions that it knows about.
>
> Extensions are of the type ".foo", which is typically found on
> Windows and Unix boxes, but not on Macinthosh. Case is ignored.
>
> Example - create a new filter that filerts out all files
> but gif and jpg image files:

```
JFileChooser chooser = new JFileChooser();
ExampleFileFilter filter = new ExampleFileFilter(
        new String{"gif", "jpg"}, "JPEG & GIF Images");
chooser.addChoosableFileFilter(filter);
chooser.showOpenDialog(this);
```

[1] Jeff Dinkins has been working on the Swing GUI Tookit since it's start in 1996, and is currently the engineering manager of the Swing & AWT teams at Sun.
Quelle: http://www.java.net/pub/au/219

[2] Jeff Dinkins; Version 1.16 07/26/04

Literatur

Frank Mittelbach, M. G., *Der LATEX Begleiter*, 2. ed., Pearson Education Deutschland GmbH, 2005.

Guido Krüger, T. S., *Handbuch der Java Programmierung*, 6. ed., Addison-Wesley Verlag, 2009.

Kecher, C., *UML 2.0 - Das umfassende Handbuch*, 2. ed., Galileo Press, 2006.

Michaelis, S., *JAXB 2.0 - Ein Programmiertutorial für die Java Architecture for XML Binding*, Carl Hanser Verlag, 2007.

Thomas Grechenig, R. B. K. K., Mario Bernhart, *Softwaretechnik*, Pearson Education Deutschland GmbH, 2010.

Ullenboom, C., *Java ist auch nur eine Insel*, 8. ed., Galileo Press, 2009.