**Aufgabe „Datenbankanbindung II“**

Zielsetzung der Aufgabe

Diese Aufgabe zeigt die Architektur eines datenbankgestützten Java-Programms mit den wesentlichen Elementen GUI, Anwendungslogik und Datenbankzugriffsschicht. Sie ist dieser Hinsicht die Fortsetzung der Aufgabe zur Notizenverwaltung (die GUI, Anwendungslogik und Persistenzschicht besaß, letztere allerdings ohne Datenbank).

Mögliche Erweiterungen der Aufgabe:

* Singleton-Pattern (als Enum-Singleton) für die DBVerbindung
* Erweiterung der GUI um eine Liste aller Termine
* DB-Konfiguration über Properties

Vorbereitung

Legen Sie auf Ihrem lokalen PostgreSQL-Datenbankserver eine leere Datenbank namens "termine" an. Diese Datenbank wird im Rahmen der Aufgabe 2 als Datenspeicher für das Programm dienen.

Sie besteht aus einer Tabelle nach folgendem Relationenschema:

termin = ( id, start, ende, thema, ort);

id ist dabei vom Typ SERIAL

start und ende vom Typ TIMESTAMP

thema und ort vom Typ TEXT

Die Anwendung

Es wird ein Ausschnitt aus einer Terminverwaltung programmiert. Dazu gehören:

Daten-Objekte für Termine (Termin), ein GUI-Element zur Darstellung dieser Objekte (TerminEditor), ein Fenster zur Kontrolle des GUI-Elements (FrameHolder mit JFrame) und einem "Speicher-Objekt"(=DAO für Data Access Objekt) für Termine (TerminDAO) sowie einer Klasse für das Herstellen und Halten der Datenbankverbindung (DBVerbindung).

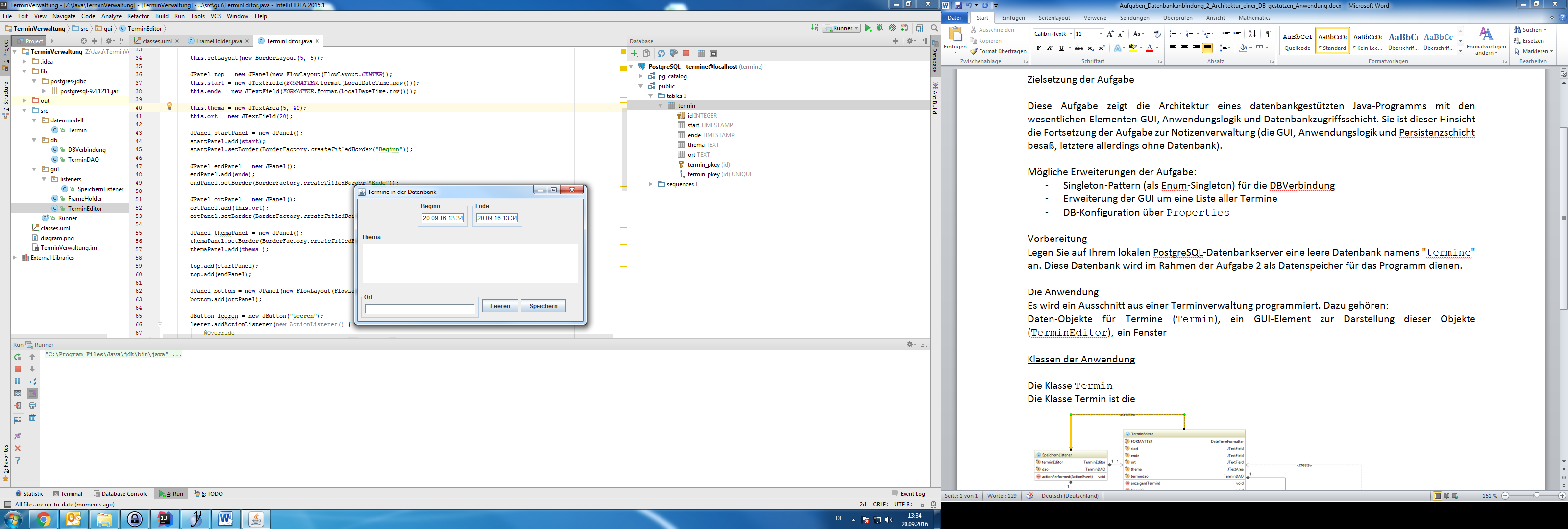
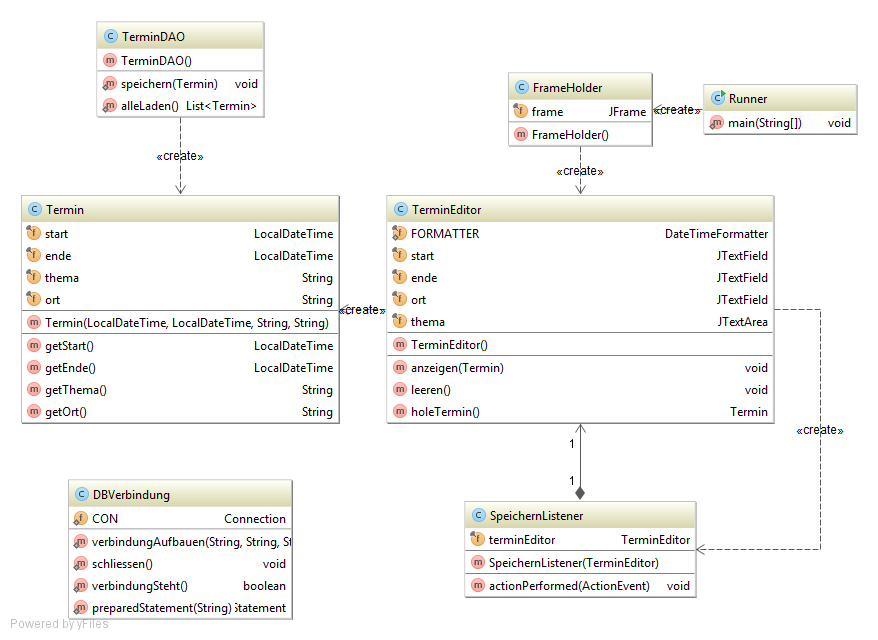


Abbildung 1:Screenshot

Klassen der Anwendung

Abbildung 2: Klassendiagramm

*Die Klasse Termin*

Die Klasse Termin ist die "fachliche" Datenklasse, um die es eigentlich in dem Programm geht. In der wirklichen Welt besteht der "fachliche" Anteil eines Programms i.a. aus deutlich mehr als einer Klasse!

Die Instanzen der Klasse Termin kennt einen Anfangs- und einen End-Zeitpunkt (java.time.LocalDateTime), ein Thema und einen Ort (beide String). Wenn der Anfangszeitpunkt nicht vor dem Endzeitpunkt liegt, wird eine IllegalArgumentException mit einem aussagekräftigen Text geworfen.

Alle Attribute sind unveränderlich und müssen bei der Objekterzeugung angegeben werden. Für alle Attribute gibt es passende Getter-Methoden.

*Die Klasse TerminEditor*

Die Klasse TerminEditor ist ein GUI-Element für die Eingabe von Termin-Daten. Sie ist von javax.swing.JPanel abgeleitet und verfügt über javax.swing.JTextFields für die Eingabe von Anfangs- und Endzeitpunkt (im Format tt.mm.jj ss:mm) und eine javax.swing.JTextArea für das Thema des Termins.

Der richtige java.time.format.DateTimeFormatter kann mit DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT) erzeugt werden.

Sie verfügt über folgende Methoden:

* void leeren() : alle Anzeigeelemente leeren,
* void anzeigen( Termin t ) : alle Anzeigeelemente mit den Daten aus dem übergebenen Termin-Objekt füllen
* Termin holeTermin() : aus den Inhalten der Anzeigeelemente ein Termin-Objekt bauen (Fehlerbehandlung selbst durchführen und Nutzer informieren!)

*Die Klasse DBVerbindung*

Die Klasse DBVerbindung verwaltet das java.sql.Connection-Objekt, über das die Datenbankverbindung hergestellt wird. Daher soll es in dem Programm höchstens eine Instanz dieser Klasse geben (Singleton-Eigenschaft, stellen wir im Moment sicher, indem der Konstruktor privat gemacht wird).

DBVerbindung besitzt folgende statische[[1]](#footnote-1) Methoden:

* boolean verbindungAufbauen(String url, String user, String pass) : Baut die Datenbank-Verbindung auf; falls dabei ein Fehler auftritt, soll das Programm mit einer Meldung auf der Konsole beendet werden (System.exit(int status)); gibt true zurück falls die Verbindung erfolgreich aufgebaut wurde.
* boolean verbindungSteht() : überprüft die DB-Verbindnung, falls es ein Connection-Objekt gibt (Timeout 0)
* void schliessen() : schließt die Datenbankverbindung, wenn es eine gibt
* PreparedStatement prepareStatement(String sql) : holt ein PreparedStatement-Objekt von der Connection und gibt es an den Aufrufer weiter. Diese Methode soll ihre Fehlerbehandlung nicht selbst durchführen!

Die schliessen()-Methode muss auf jeden Fall ausgeführt werden, wenn das GUI-Programm beendet wird. Dies kann mit einem java.awt.event.WindowListener an dem JFrame der Anwendung erreicht werden, der die windowClosing()-Methode für diesen Aufruf nutzt.

*Die Klasse TerminDAO*

Das TerminDAO (Data Access Object) greift auf die statischen Methoden der DBVerbindung zu, über die es seine Datenbankoperationen abwickelt. Es besitzt statische[[2]](#footnote-2) öffentliche Methoden für alle Datenbankoperationen, die mit Terminen zu tun haben und vom Java-Programm ausführbar sein sollen.

In jeder dieser Methoden steht das SQL-Statement in einem String, die Vorbereitung und Konfiguration des Statements (setXYZ()), die Ausführung (executeUpdate() oder executeQuery()) und die Interpretation der Ergebnisse (z.B. ResultSet von Termindaten -> ArrayList<Termin>). Jede Methode führt ihre eigene Fehlerbehandlung durch.

Für die Speicherung der Zeitpunkte ist die Postgres-JDBC-Dokumentation eine nützliche Quelle!

Es gibt eine Methode void speichern (Termin t), die den übergebenen Termin in der Datenbank speichert und eine Methode List<Termin> alleLaden(), die eine Liste aller in der Datenbank gespeicherten Termine holt.

*Die Klassen FrameHolder, SpeichernListener und Runner*

Der FrameHolder ist – wie üblich – Besitzer des zur Anzeige genutzten JFrames (WindowListener nicht vergessen!).

Der SpeichernListener hängt an dem Speichern-Button im TerminEditor. Er holt bei Auslösung das Termin-Objekt vom TerminEditor und speichert mittels der Methoden des TerminDAO.

Die main()-Methode im Runner muss erst die DBVerbindung dazu veranlassen, eine Verbindung herzustellen, und dann die Erzeugung der GUI anstoßen.

Überlegen Sie sich eine sinnvolle Package-Struktur für die Anwendung!

Zusammenfassung der in dieser Aufgabe gezeigten projektrelevanten Inhalte

* "Schichtarchitektur" für datenbankgestützten Anwendungen
* DAO-Muster zur Organisation der Datenbankoperationen einer Anwendung
* Klasse zur Verwaltung der Verbindung
* WindowListener zum Schließen der DB-Verbindung beim Programmende
* Nutzung der Java-8 Datum/Zeit-Klassen in Verbindung mit PostgreSQL

Abbildung 3 : Schichtarchitektur

GUI

Anwendungslogik/Geschäftslogik

Datenmodell ("fachliche Java-Objekte")

Persistenzschicht (Datenbank)

Listener

DAOs

1. Die sind statisch, damit sie von überall aus dem Programm erreichbar sind, ohne dass eine Objekt-Referenz "herumgereicht" werden muss. [↑](#footnote-ref-1)
2. Auch die sind statisch: Jeder Programmteil, der eine Termin-bezogene Datenbank-Operation ausführen muss, kann das tun, ohne dafür eine TerminDAO-Referenz besitzen zu müssen [↑](#footnote-ref-2)