# Blumenthaler Aue

## Bericht zur Anwendungsentwicklung

“De Blomendaler Au is en Beek, de von rechts in de Werser münnt”

Ein Projekt von
**Simon Bullik**

08. März 2020
Fachlehrer: Jürgen Wolkenhauer

# Inhaltsverzeichnis



# Allgemeines

Das erarbeitete Programm samt Klassen, Methoden, Variablennamen und Kommentaren ist auf Englisch geschrieben, da dies dem internationalen Standard von Software entspricht und Problemen wie z. B. der Verwendung von Umlauten vorbeugt.

Zur Bearbeitung dieses Projekts wurde uns Zeit in der Schule zur Verfügung gestellt, die wir genutzt haben. Ein Teil der Arbeit geschah aber außerhalb des Unterrichts.

# Aufgabenstellung

Es soll eine Dokumentation zum Entwurf der Applikation, die objektorientiert und modularisiert implementiert werden soll, erstellt werden. Diese soll schriftlich abgegeben werden und beinhaltet ein Klassendiagramm. Das Programmfenster soll eine Menüleiste, mit der das Programm beendet werden kann und Informationen über den Verfasser beinhaltet, enthalten. Die Verbindung zur Datenbank soll über einen Menüeintrag hergestellt werden. Eine erfolgreiche Verbindung soll in der Statuszeile ausgewiesen werden. Über ein geeignetes Control und die entsprechende Programmlogik in der Anwendung, soll ein Pflanzenschutzmittel (z.B. Altrazin oder Simazin) für die Filterung bei der Datenbankabfrage ausgewählt werden können. Ein Control „Daten lesen“ soll es dem Anwender ermöglichen, die Messwerte aus der Datenbank auszulesen. Die Messwerte sollen nach der Datenbankspalte „Index“ aufsteigend sortiert gelesen und daraufhin in geeigneten Controls als Zahlenwerte und in einem Diagramm angezeigt werden. Die x-Achse des Diagramms soll der Index und die y-Achse die Konzentration des Pflanzenschutzmittels (Messwert) sein.

Die Applikation soll die Möglichkeit bieten, den Pfad zu der Datenbank auszuwählen. Beim Beenden der Applikation sollen einige Einstellungen in einer Datei gespeichert (ini, Property, json ...) werden. Sollte es diese Datei nicht geben, so soll das Programm mit Standardwerten starten. Beim Beenden des Programms soll die Einstellungsdatei erstellt werden. Diese Einstellungen sollen beim nächsten Start berücksichtigt werden. Die Einstellungen sollen den Pfad zur DB, Fenstergröße und Position enthalten.

# Technische Ausstattung

Das Programm wurde in der IDE IntelliJ 2019 und auf Basis von Java 11 entwickelt. Um das Programm korrekt ausführen zu können, muss eine kompatible Java-Version (11 oder neuer) auf dem ausführenden Rechner vorhanden sein.

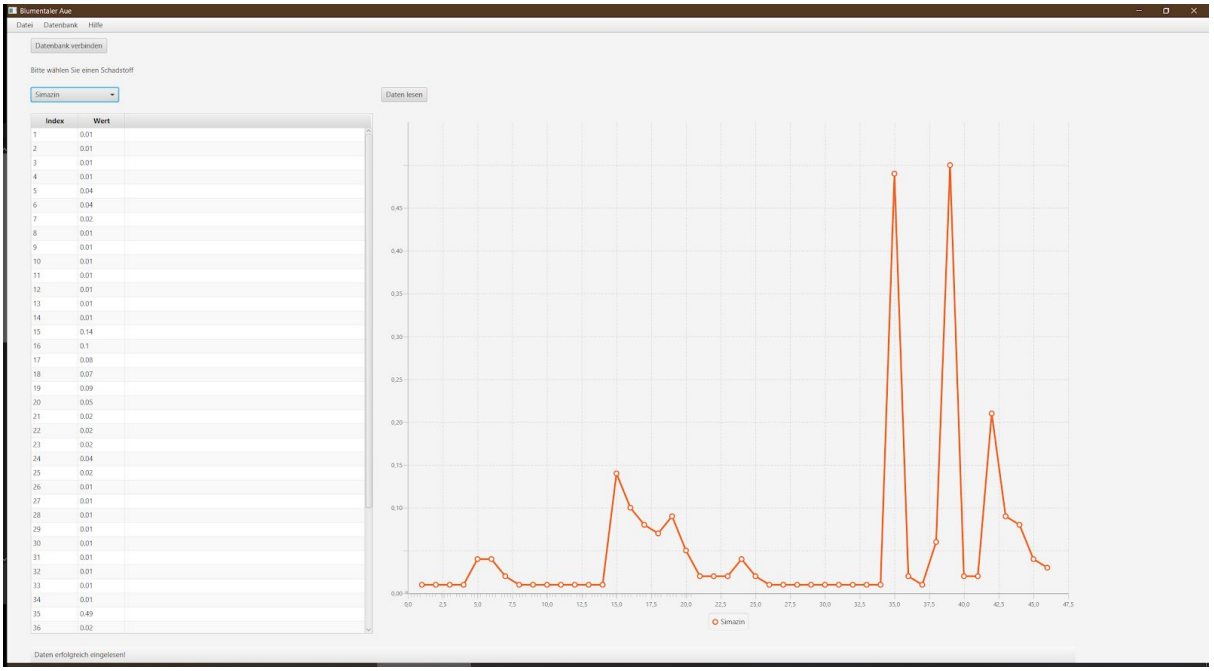
Die Programmabhängigkeiten werden mithilfe des Build-Management-Automatisierungs-Tools Gradle durch die build.gradle-Datei aufgelöst. Der Benutzer kommt hierbei nicht in direkten Kontakt mit den benötigten Bibliotheken.

Es wird empfohlen, die GUI auf einem möglichst großen Bildschirm darzustellen, damit genug Platz ist, um alle Schaltflächen anzuzeigen und somit die volle Funktionalität des Programms genutzt werden kann.

# Benutzeranleitung

Zur Ausführung des Programms empfehlen ich die IDE *IntelliJ*. Zunächst müssen alle Abhängigkeiten korrekt aufgelöst werden - unter *Java* macht *Gradle* dies selbstständig.

Um die Anwendung zu starten, kann die mitgelieferte Startkonfiguration in IntelliJ genutzt oder einer der beiden Befehle aus der Readme ausgeführt werden.
Der Start der GUI kann einige Sekunden dauern. Sobald die Benutzeroberfläche dargestellt wird, kann die Benutzung des Programms beginnen. Die Anordnung der GUI-Elemente ändert sich im Verlauf der Benutzung nicht; es stehen also alle Funktionalitäten auf einen Blick zur Verfügung, was im folgenden genauer erklärt ist. Die Oberfläche ist an das Beispiel angelehnt, welche uns bereitgestellt worden ist und zur Referenz im *docs* zu finden ist.




In der Menüleiste befinden sich drei Menüs. Im “Datei” Menü ist aktuell nur die Option zum Beenden des Programms. Im Reiter Datenbank gibt es die Option zum erstellen einer neuen Datenbank. Dafür öffnet sich ein Dialog um den Ort sowie den Namen der Datenbank auszuwählen. Darunter befindet sich die Option zum Auswählen einer existierenden Datenbank. Auch hierfür öffnet sich ein Dialog um die entsprechende Datei zu suchen. Die nächsten zwei Optionen sind zum Verbinden mit der Datenbank und um diese Verbindung zu beenden. Als letztes findet man die Option Beispieldaten in die Datenbank hinzuzufügen. “Hilfe” ist der letzte Menüreiter. Hier findet man den Dialog mit den Informationen zum Programm.

Über den Button “Datenbank verbinden” kann man sich mit der Datenbank verbinden. Über den Button “Daten lesen” werden die Pflanzenschutzmittel aus der Datenbank ausgelesen und in das Dropdown eingetragen. Wenn man nun ein Mittel im Dropdown auswählt, werden die entsprechende Werte in der Tabelle angezeigt und grafisch im Diagramm visualisiert.

# Erläuterung des Programms

## Abhängigkeiten

Das Programm ist von wenigen Bibliotheken abhängig, die im folgenden näher erläutert sind.

### JavaFX

*JavaFX* ist ein Framework zur Erstellung von graphischen Oberflächen mit *Java*. Seit der Version 11 ist es Teil vom *OpenJDK* und wird unter dem Projektnamen *OpenJFX* weiterentwickelt.

### sqlite-jdbc

Die *SQLite JDBC* Bibliothek wird benötigt, um mit der SQLite Datenbank interagieren zu können.

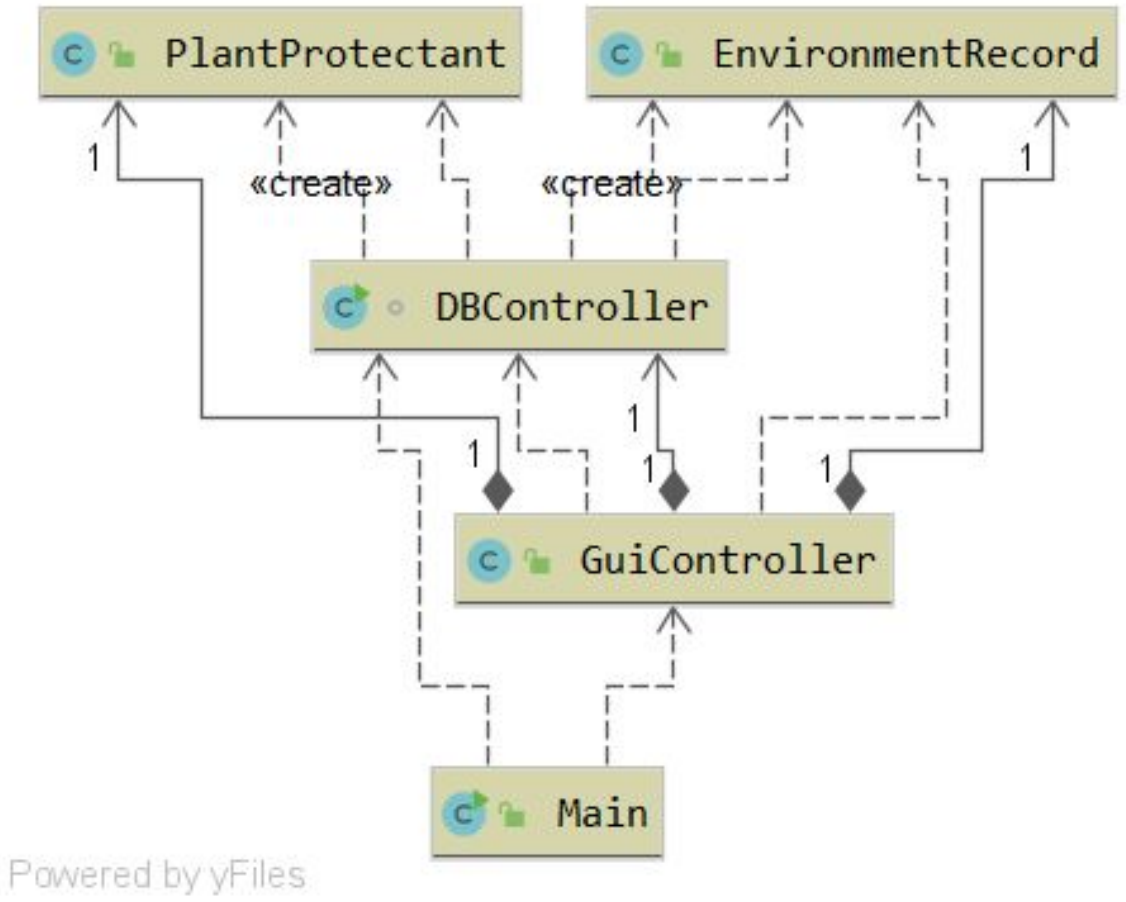
### commons-io

*Commons IO* ist eine Hilfsbibliothek um ohne viel *Boilerplate* Ein- und Ausgabe Operationen wie das Einlesen einer Datei zu schreiben.

### annotations

Mit Hilfe dieser Bibliothek von *JetBrains* haben wir *Annotations* hinzugefügt, welche anzeigen, ob Variablen *Null* sein können oder nicht. Durch die Integration der IDE wurden uns so Warnungen angezeigt, wenn wir keine *Null*-Checks machen aber der Wert *Null* sein kann und vice versa. So wurden *NullPointerExceptions* vermieden.

## UML-Klassendiagramm




Das obige Diagramm verdeutlicht die interne Struktur der Anwendung.

Die “Main” Klasse startet das Programm. Über sie wird auch die JavaFX Oberfläche gestartet. Das Speichern und Laden der Einstellungen geschieht auch hier.

Der “GuiController” führt entsprechende Aktionen aus, welche zum Benutzer gewünscht sind. Dabei delegiert er entweder zum “DBController” oder arbeitet mit den vorhandenen Daten. Außerdem können natürlich auch Aktionen wie das Schließen des Fensters oder das Öffnen des Dialogs hier asugeführt.

Der “DBController” ist die Instanz zwischen der Software und der Datenbank. Er öffnet und schließt die Verbindungen, führt SQL Befehle aus und erstellt aus den zurückgelieferten Daten die entsprechende Objekte.

“PlantProtectant” und “EnvironmentRecord” sind pure Datenhalter. Sie enthalten die Daten, welche aus der Datenbank ausgelesen worden sind.

## Einzelheiten zur Software

Das zugehörige Dokumentation der Klassen ist im *docs*-Ordner zu finden und wurde automatisch mithilfe von *Javadoc* erstellt. Bei den erstellten *Javadoc*-Beschreibungen wurde darauf geachtet, kurz und prägnant die Aufgabe der einzelnen Klassen und Methoden zu schildern.

Die Namen der Klassen und Methoden, die Ordnerstruktur unseres Projekts sowie ergänzende Kommentare zu wichtigen Stellen im Code tragen dazu bei, dass die von uns erarbeitete Software übersichtlich und (hoffentlich auch für Außenstehende) einfach verständlich ist.

# Erkenntnisgewinn

Dieses Projekt hat das Wissen in verschiedenen Bereichen erweitert, was im Folgenden näher aufgeschlüsselt ist:

1. Mit Graphen, Dropdowns, Menüs und Tabellen haben wir die Funktionalitäten von **JavaFX** ausgiebig genutzt. Besonders der Graph war eine schöne neue Erfahrung, die ich machen durfte.
2. Das Projekt war das erste Projekt in Java in welchem ich mit **SQLite** gearbeitet habe. Es war interessant Mal wieder SQL zu schreiben, wenn man sonst nur mit Frameworks wie Hibernate arbeitet.
3. In diesem Projekt habe ich das Build-Management-Automatisierungs-Tool **Gradle** verwendet. Wenn man die Konfiguration mit der pom.xml von Maven vergleicht, sieht man wie schön aufgeräumt Gradle sein kann.

Es hat Spaß gemacht dieses kleine Projekt als Prüfungsvorbereitung mit Hinsicht auf die ITA Prüfung zu machen.

**- SimonIT**