Technische Dokumentation Weinbau

Die technische Dokumentation erklärt die wichtigsten Funktionalitäten und Konzepte hinter dem Programm dem Weinbauprogramm.

Die ausführliche Dokumentation der einzelnenn Klassen, Methoden und Variablen ist unter https://monaKac.github.io/ zu finden.

Das Programm wurde in Java geschrieben.
In dem Programm wurden Inhalt und Darstellung getrennt.
Dabei sind die Klassen GUI und Feldarbeiter für die Darstellung zuständig.

GUI

Kümmert sich dabei um die Darstellung des Hauptprogramms, dass durch den Winzer bedient wird.
Es erstellt die Hauptseite mithilfe eines JFrame mit jeweils einem JPanel pro Weinberg des Winzers.

Durch einen JButton kann der/die Anwender/in einen weiteren JFrame öffnen mit weiteren Informationen z.B. Anzeige von Wetterdaten.

Feldarbeiter

Ist eine Nebenanwendung, die von einem Feldarbeiter im Außendienst bedient wird.
Auch diese besteht aus einem JFrame.

SchnittstelleFeldanwendung

Dient als Schnittstelle in weiteren Releases zur Anbindung eines vollautomatischen Weinberges.
Die Bedienung dieses erfolgt in der Hauptanwendung, durch drücken eines Buttons wird z.B. die Bewässerung gestartet.

Datenbank

Die Klasse Datenbank simuliert eine Datenbankanbindung. Dabei werden zu beginn der Programmausführung alle benötigten Daten initialisiert und während der Laufzeit kann über Methoden dieser Klasse auf die Daten zugegriffen werden

Main

In dieser Klasse befindet sich die main Methode. Hier werden die Methoden zur Datenbank Initialisierung in der richtigen Reihenfolge aufgerufen (Erst initialiserung des Winzers danach des Wetters und der Benutzeroberflächen)

Wetter

Die Klasse Wetter dient vor allem als Datenstruktur.

Ein Wetterobjekt besteht aus einem Datum, Temperatur, Sonnenstunden, Niederschlagsmenge und Regenwahrscheinlichkeit, einer Bewölkung(Enum) die abhängig der anderen Eigenschaften berechnet wird, es ist einem Weinbergobjektes zugeordnet und besitzt eine Id die zur eindeutigen Bestimmung des Objektes dient (Primärschlüssel).

Die Id eines Wetterobjektes wird über die Klassenvariable nextId während dem Konstruktor Aufrufes bestimmt. Die Variable wird bei jedem Konstruktoraufrufs um eins incrementiert.

Bewoelkung(Enum)

Das Enum Bewoelkung hält alle in unserer Programlogik vorgesehenen Bewoelkungszustände fest. Die Methode toString() wurde überschrieben und gibt den jeweiligen Namen des Enums zurück. Außerdem gibt die Methode getIcon() ein zu dem Zustand passendes BufferedImage zurück. Dass für die Darstellung in der GUI genutzt wird.

Winzer

Die Klasse Winzer dient vor allem als Datenstruktur.

Ein Winzerobjekt besteht aus einem Vornamen, Nachnamen, einer ArrayList der Klasse Weinberg und einer Id. Auch hier wird die Id über die Klassenvariable nextId bestimmt.
ArrayList wird verwendet da die Eigenschaft der dynamischen Größe benötigt wird.

Weinberg

Die Klasse Weinberg dient größtenteils als Datenstruktur.
Ein Weinbergobjekt besitzt die Werte Alter (da jede Pflanze eines Weinberges zum gleichen Zeitpunkt eingepflanzt wird, wurde dieses Attribut nicht der Pflanze zugeordnet), Bodenfeuchtigkeit und Mineraliengehalt, sowie einer ArrayList der Klasse Pflanzen, einem zugeordneten Winzerobjektes, Statusobjektes, sowie einem Namen und einem Kommentar, der zur Kommunikation zwischen Winzer und Feldarbeiter dient.
Auch ein Weinbergobjekt besitzt eine Id und die Klasse Weinberg die Klassenvariable nextId.

Die Klasse besitzt Methoden um Objekte zu der ArrayList hinzuzufügen und zu entfernen und die Methoden:

Methode	Funktion
getPflanzenGroesse()	Berechnet die durchschnittliche Groesse der Pflanzen des Weinberges (arith. Mittel)
getZuckergehalt()	Berechnet den durchschnittlichen Zuckergehalt der Pflanzen des Weinberges
isKrank()	Gibt true zurück wenn keine Pflanze krank ist, Gibt False zuruück wenn mind 1 Pflanze krank ist.

Pflanzen

Die Klasse Pflanzen dient größtenteils als Datenstruktur.

Ein Objekt der Klasse Pflanzen besitzt eine Variable Zuckergehalt, Groesse und Krankheit (boolean). Außerdem eine Id die über die Klassenvariable nextId im Konstruktor festgelegt wird.

Status

Die Klasse Status dient als Datenstruktur und verbindet das Enum Weinbergstatus mit einem dazugehörigen Prozentsatz, der den Fortschritt anzeigt.

Weinbergstatus (Enum)

Das Enum Weinbergstatus beschreib den Status eines Weinberges.

Es besitzt eigene Variablen, die für die personalisierte Konfiguration der Empfehlung dienen. Festgelegt werden die minimale und maximale Bodenfeuchtigkeit, ein minimaler Mineraliengehalt,

minimaler und maximaler Niederschlag, minimale Sonnenstunden, eine minimale und maximale Temperatur, eine maximale Windgeschwindigkeit, ein minimaler Zuckergehalt und eine minimale Größe.
Die Variablen sind an die einzelnen Phasen angepasst.

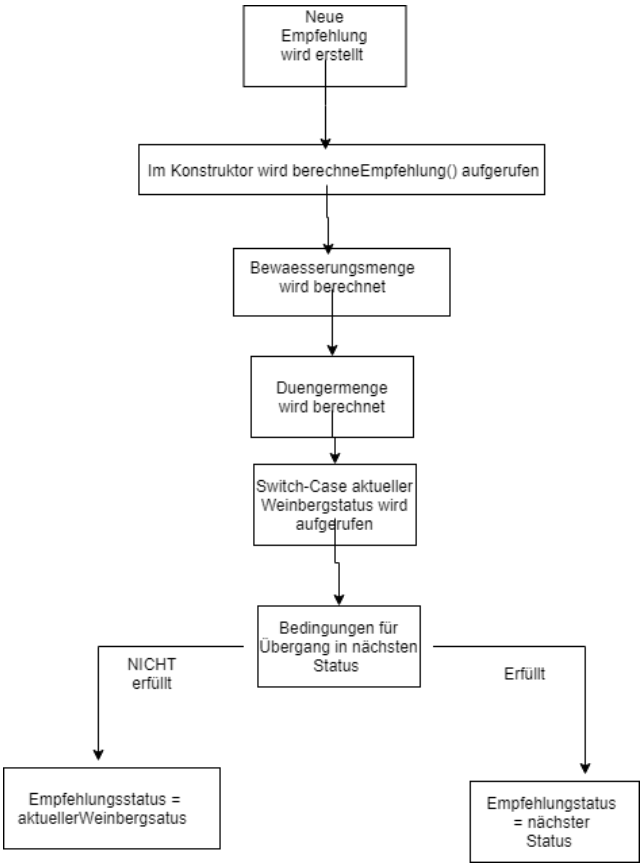
Standardwerte sind gegeben, die Grenzwerte können jedoch durch setter und getter angepasst werden. Eine Benutzeroberfläche dafür steht jedoch in diesem Entwicklungsstand nicht zur Verfügung.

Empfehlung

Die Klasse Empfehlung berechnet die Empfehlung für Bewässerung, Düngen und Übergang in den nächsten Weinbergstatus.

Die Klasse verwendet folgende Variablen :

Name	Datentyp	Nutzen
Weinberg	Weinberg	Aktueller Weinberg für den die Empfehlung erstellt wird
Wettervorhersage	Wetter[] (Array Länge 7)	Zieht sich die wettervorhersage aus der Datenbank der nächsten 7 Tage
Warnung	Boolean	Zeigt an ob die Empfehlung so dringend ist, dass sie als Warnung angezeigt wird.
Empfehlungsstatus	Status	Zeigt den Status an der Empfohlen wird (siehe Diagramm)
Datum	Int	Datum des Tages an dem die Empfehlung erstellt wurde
Text	String	Empfehlungstext (für Übergang in neuen Weinbergstatus)
TextBewaesserung	String	Empfehlungstext für Bewässerungsmenge
TextDuenger	String	Empfehlungstext für Düngermenge



Die Empfehlung ist abhängig von dem aktuellen Status in dem sich der Weinberg im Moment befindet und deshalb gibt es für jede Empfehlung in einen Übergang in einen neuen Status eine eigene Methode. (Alle Stati treten in einen Jahr nur einmal auf und jedes Jahr in der gleichen Reihenfolge).

Allgemeine Methoden

Methodenname	Funktion
`Void berechneEmpfehlung()`	Entscheidet mithilfe eines Switch-Cases und dem Status in dem sich der Weinberg im Moment befindet welche Methode zur Berechnung der Empfehlung aufgerufen werden soll. Es ruft außerdem die Methoden `empfehlungBewaesserung()` und `empfehlungDuengen()auf.`
`Void empfehlungBewaesserung()`	Berechnet die benötigte Bewässerungsmenge für den Weinberg. Abhängig von dem aktuellen Status, dem Wetter der nächsten 7 Tage, der aktuellen Bodenfeuchtigkeit und dem Alter der Pflanzen.
`Void empfehlungDuengen()`	Berechnet die benötigte Düngermenge für den Weinberg. In Abhängigkeit von dem aktuellen Status und Mineralgehalt des Bodens und der Anzahl der Pflanzen des Weinberges
`int durchschnittNiederschlag(int tage) throws Exception`	Berechnet die durchschnittliche Niederschlagsmenge in Abhängigkeit der

	Regenwahrscheinlichkeit für die nächsten tage Tage. Es wird eine Exception geworfen wenn tage kleiner 1 oder größer 7 ist. Da nur die Wettervorhersage für die nächsten 7 Tage vorliegt.
`int durchschnittWind(int tage) throws Exception`	Berechnet den durchschnittlichen Wind für die nächsten tage Tage. Es wird eine Exception geworfen wenn tage kleiner 1 oder größer 7 ist. Da nur die Wettervorhersage für die nächsten 7 Tage vorliegt.
`int durchschnittTemp(int tage) throws Exception`	Berechnet die durchschnittliche Temperatur für die nächsten tage Tage. Es wird eine Exception geworfen wenn tage kleiner 1 oder größer 7 ist. Da nur die Wettervorhersage für die nächsten 7 Tage vorliegt.
`int durchschnittSonnenstunden(int tage) throws Exception`	Berechnet die durchschnittlichen Sonnenstunden für die nächsten tage Tage. Es wird eine Exception geworfen wenn tage kleiner 1 oder größer 7 ist. Da nur die Wettervorhersage für die nächsten 7 Tage vorliegt.

Des Weiteren gibt es noch einige Getter und Setter die nur geringe Funktionalität ausführen.

Empfehlungsmethoden

Alle Empfehlungen verlaufen nach dem gleichen Schema.
Wenn die Bedingungen für den nächsten Status erfüllt sind wird der empfehlungsStatus = dem nächsten Status gesetzt.
Wenn die Bedingungen nicht erfüllt sind wird der empfehlungsStatus auf den Status des Weinberges gesetzt.
Die Soll-Werte für die Bedingungen sind abhängig von dem aktuellen Weinbergstatus, können jedoch konfiguriert werden und somit an die Bedürfnisse des Winzers angepasst werden.
Eine Bedingung ist immer, dass der aktuelle Status bereits abgeschlossen ist (^= 100%)

Methodenname	Funktionalität
`Void empfehlungWinterruhe()`	Berechnet die Empfehlung abhängig von der Temperatur der nächsten 7 Tage. Entscheidend ist der Beginn des Austriebs der bei einer durchschnittlichen Temperatur von 8-10 C beginnt.
`void empfehlungRebschnitt()`	Berechnet die Empfehlung abhängig von dem Niederschlag der nächsten 7 Tage. Damit es feucht genug ist damit die Reben beim biegen nicht brechen.
`void empfehlungReberziehung()`	Berechnet die Empfehlung abhängig von der Bodenfeuchtigkeit und ob es in den nächsten 7

	Tagen Starkregen geben soll. Um abrutschen der Hänge zu verhindern.
`void empfehlungBodenarbeit()`	Berechnet die Empfehlung abhängig von dem Niederschlag, Sonnenstunden und Temperatur der nächsten 7 Tage. Es muss warm und sonnig genug sein und genug regnen.
`void empfehlungPflanzenschutz()`	Berechnet eine Warnung vor Pilzkrankheiten, abhängig von der Bodenfeuchtigkeit, Niederschlag, Sonnenstunden und Temperatur. Übergang in den nächsten Status findet ab Beginn der Befruchtung statt und muss anhand der Entwicklung der Pflanze manuell entschieden werden.
`void empfehlungBefruchtung()`	Berechnet die Empfehlung abhängig von Niederschlag und Sonnenstunden.
`void empfehlungLaubarbeit()`	Berechnet die Empfehlung abhängig von der große der Pflanze und dem Zuckergehalt.
`void empfehlungErnte()`	Berechnet die Empfehlung abhängig von der Temperatur.

JUnit5 Tests

Die Klassen TestEmpfehlung, TestWeinberg, TestWetter testen mithilfe von JUnit5 Tests vollautomatisch die Klassen Empfehlung, Weinberg und Wetter bedarfsgerecht.