Gruppe 7: Marcus, Esben og Sofie

Modulopgave 2 - Krydsord

Krav:
Vi skal lave et program, der kan finde krydsord med tre ord hen og tre ord ned, hvor alle ni bogstaver skal være unikke. Programmet skal udskrive tid og antal løsninger. Vi itererer over vores algoritmekode for at finde den hurtigste løsning.

Vi har besluttet at standardisere ordene, så alle ord bliver stavet med småt, der må ikke være tal i ordet, og alle ord skal i forhold til kravene indeholde unikke bogstaver. Vi har været nødt til at slette et enkelt ord manuelt, olé. MySQL læste bogstavet 'é' som et 'e' uanset om tegnsættet var sat til latin1 eller utf8.

UC#1 Casual:

Property	**Description**
Name	Algoritme optimering.
Goal	At optimere vores algoritme inden for tid og antal krydsord.
Success Scenario	Algoritmen giver alle samtlige krydsord under x minutter.
Precondition	Java og MySQL environment.
Story	Programmet kører en af følgende algoritmer: 1) Med ForEach(:) loops. 1.1) Med ForEach(:) loops under løbende betingelse, at bogstaverne i de fundne ord er unikke. 2) Der tages udgangspunkt i ét ord som der løbende joines flere ord på med metoden substring(). Til sidst undersøges om alle bogstaver er unikke. Brugeren får tal, på antal krydsord dannet samt hvor langt tid programmet tog. Se længere nede i rapport for uddybende forklaring.

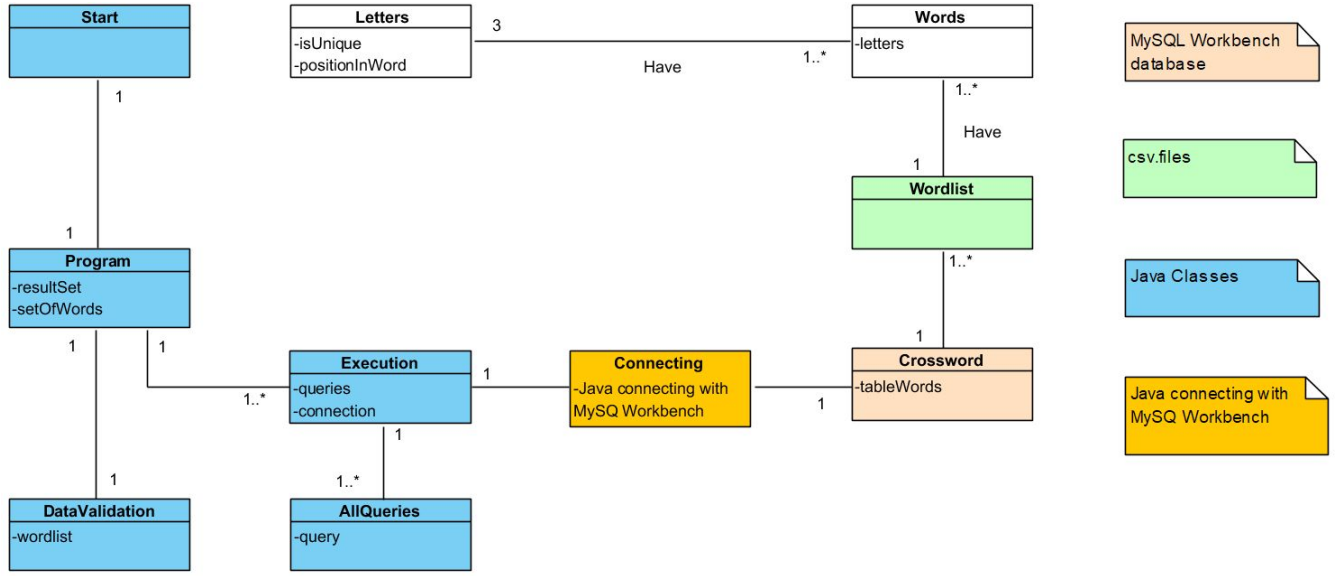
UC#2 Brief - Data Validation:
Vi opretter en tabel i MySQL Workbench og lægger alle vores ord ind.
Ordene læses ind i Java og valideres, efter følgende krav:

1. Ingen store bogstaver.
2. Alle bogstaver skal være unikke.
3. Ingen tegn.
4. Ingen tal.

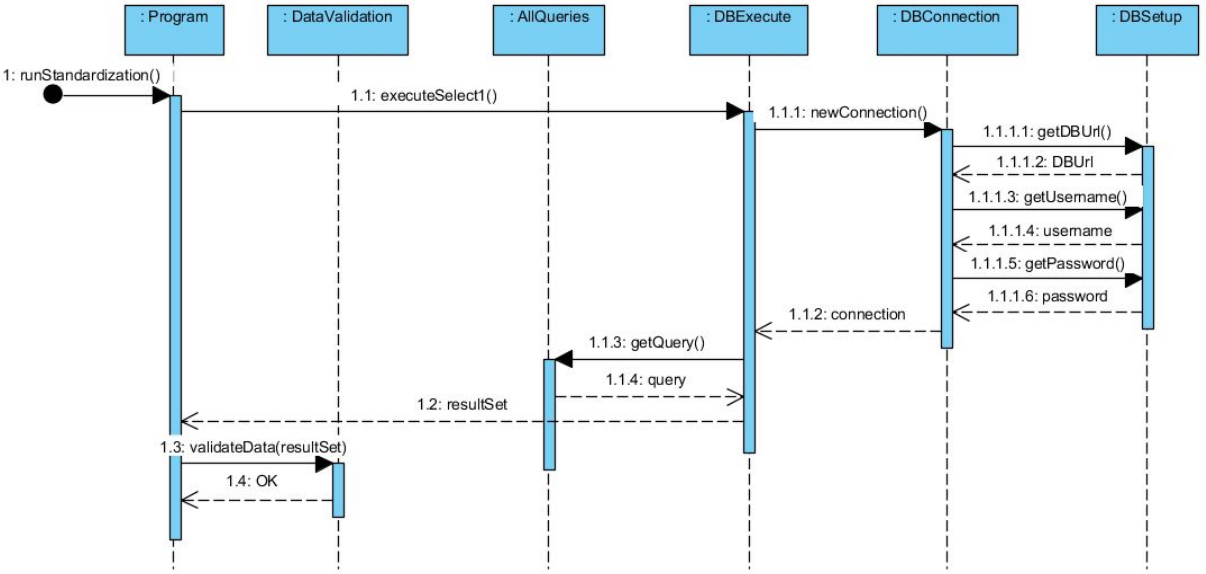
De sorterede ord sættes ind i et TreeSet.

UC#3 Brief - Table generation:
Vi opretter en ny tabel i MySQL Workbench gennem Java.
Vi sætter derefter vores validerede datasæt ind i denne tabel.

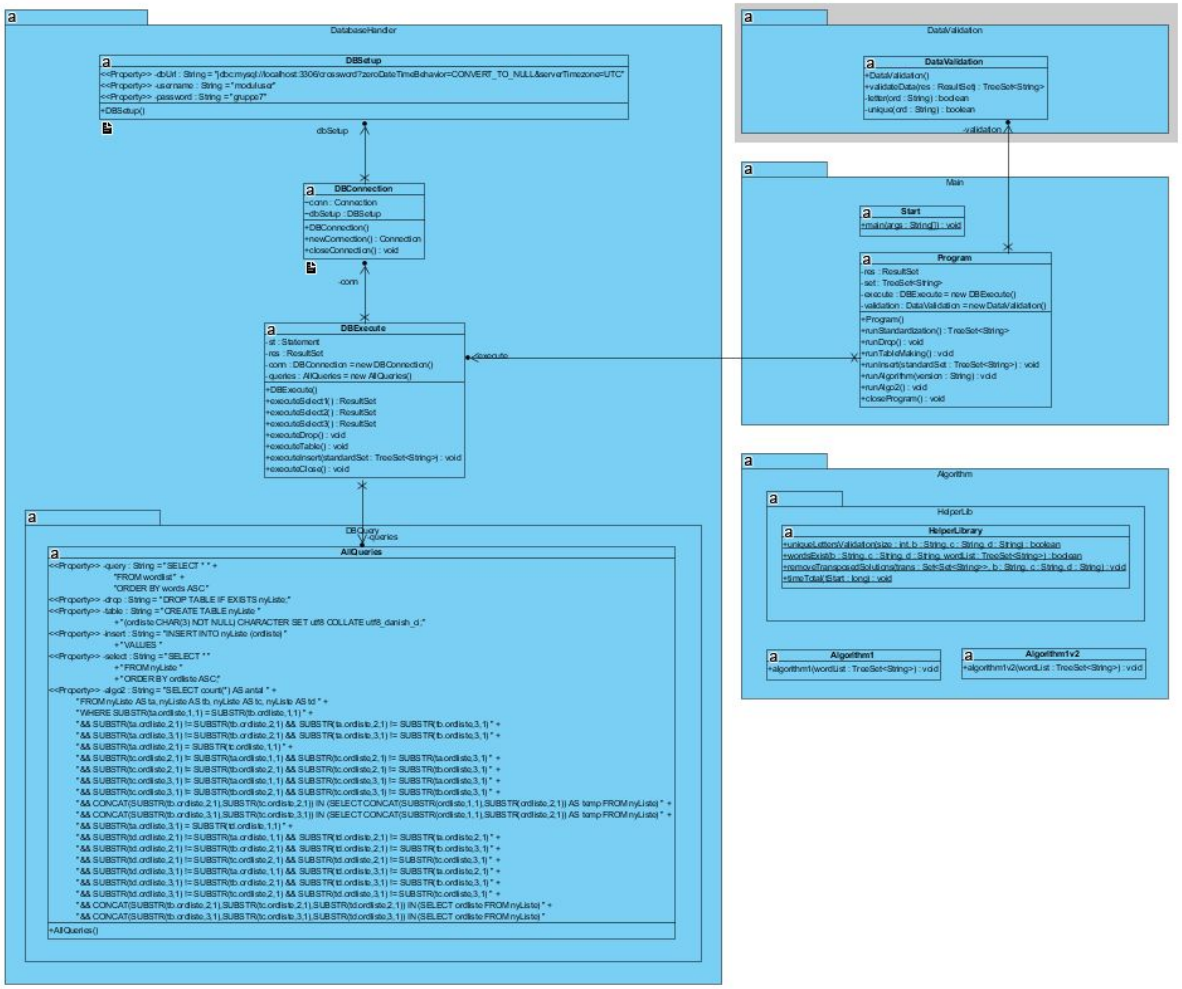
Domænemodel:



SD over datastandardisering:



Klasse Diagram - Se SD-DomainModel-ClassDiagram-Final.VPP fil



Algoritme afvikling:
Alle algoritmer er kørt på samme grunddata, tabellen nyListe, der indeholder 1133 unikke ord. Alle ord er sat til lowercase og udgøres af tre bogstaver, der er unikke i ordet. Tal og andre specialtegn er fjernet.
For at mindske CPU'ens indflydelse på forskelle i kørselstiderne, er samme PC benyttet. Strømmen har været tilsluttet den bærbare PC under kørslerne, hvilket har haft markant indflydelse på kørselstiderne.
Den transponerede løsning, der modsvarer enhver unik løsning, tælles ikke med i resultatet.

Algoritme 1 (JAVA):
Første udgave af vores Java baserede algoritme er en simpel brute force metode. Alle kombinationer af tre ord i én dimension køres igennem. For hver kombination udføres to tjek:

1. De tre ord der dannes i anden dimension skal eksistere i vores tabel af mulige ord.
2. Alle ni bogstaver, der udgør den mulige løsning, skal være unikke.

Hvis begge tjek overholdes, accepteres de seks ord som en gyldig løsning.

- *Kørselstid: 6 min. og 32 sek.*
- *Løsninger: 141.238*

Algoritme 1 Version 2 (JAVA):
Videreudviklingen af vores brute force algoritme skærer kørselstiden ned til det halve. Der er indført et tjek efter andet ord er fundet. Hvis de seks bogstaver, der udgør ord ét og to, ikke er unikke, kasseres andet ord og der forsøges med næste ord fra listen.

- *Kørselstid: 3 min. og 56 sek.*
- *Løsninger: 141.238*

Algoritme 2 (SQL):
Der laves et self join på tabellen nyListe, under betingelse at første bogstav i ord A matcher første bogstav i ord B. Der testes for om bogstaverne der udgør begge ord er unikke (med undtagelse af join bogstavet).

A1,B1	A2,C1	A3,D1
E1,B2	E2,C2	E3,D2
F1,B3	F2,C3	F3,D3

Samme type join udføres med ord C på andet bogstav i A. Der laves igen et tjek for om alle bogstaver er unikke. Dernæst undersøges vores oprindelige tabel for, om de to første bogstaver i et eller flere ord er identiske med andet bogstav i B konkateneret med andet bogstav i C. Samme tjek udføres for tredje bogstav i B og C.
Igen laves et join på tredje bogstav i A og første bogstav i D. Samme tjek som tidligere udføres og hvis de overholdes gemmes de fundne ord, som en godkendt løsning.

- *Kørselstid: 1 min. og 50 sek.*
- *Løsninger: 141.238*

Eksempler på løsninger

ordA	ordB	ordC	ordD	ordE	ordF
gab	gør	ajo	bed	øje	rod
gab	gøs	ajo	bed	øje	sod
gab	gås	ali	bed	åle	sid
dab	døv	ajo	ben	øje	von
gab	gød	ajo	ben	øje	don
gab	gås	ali	ben	åle	sin
lab	liw	adt	beo	ide	wto
gab	gui	adv	beo	ude	ivo
dab	dus	agn	beo	uge	sno
dab	døs	agn	beo	øge	sno
lab	luk	agn	beo	uge	kno
lab	lus	agn	beo	uge	sno
lab	lås	agn	beo	åge	sno
gab	går	ali	beo	åle	rio
dab	dis	alk	beo	ile	sko

Konklusion:
Vi har ikke fået implementeret projektopgave del 3. Da vi har været i tidsmangel og fokuseret på vores algoritme optimering.
Vi startede på en tredje algoritme i SQL der tager udgangspunkt i to midlertidige tabeller, der hver indeholder en hjørneløsning i krydsordstabellen. De joines under forudsætning af at ordet i midten (lodret og vandret) eksisterer i vores tabel af mulige ord. Algoritmen tog over ti minutter at køre og vi opgav derfor ideén.