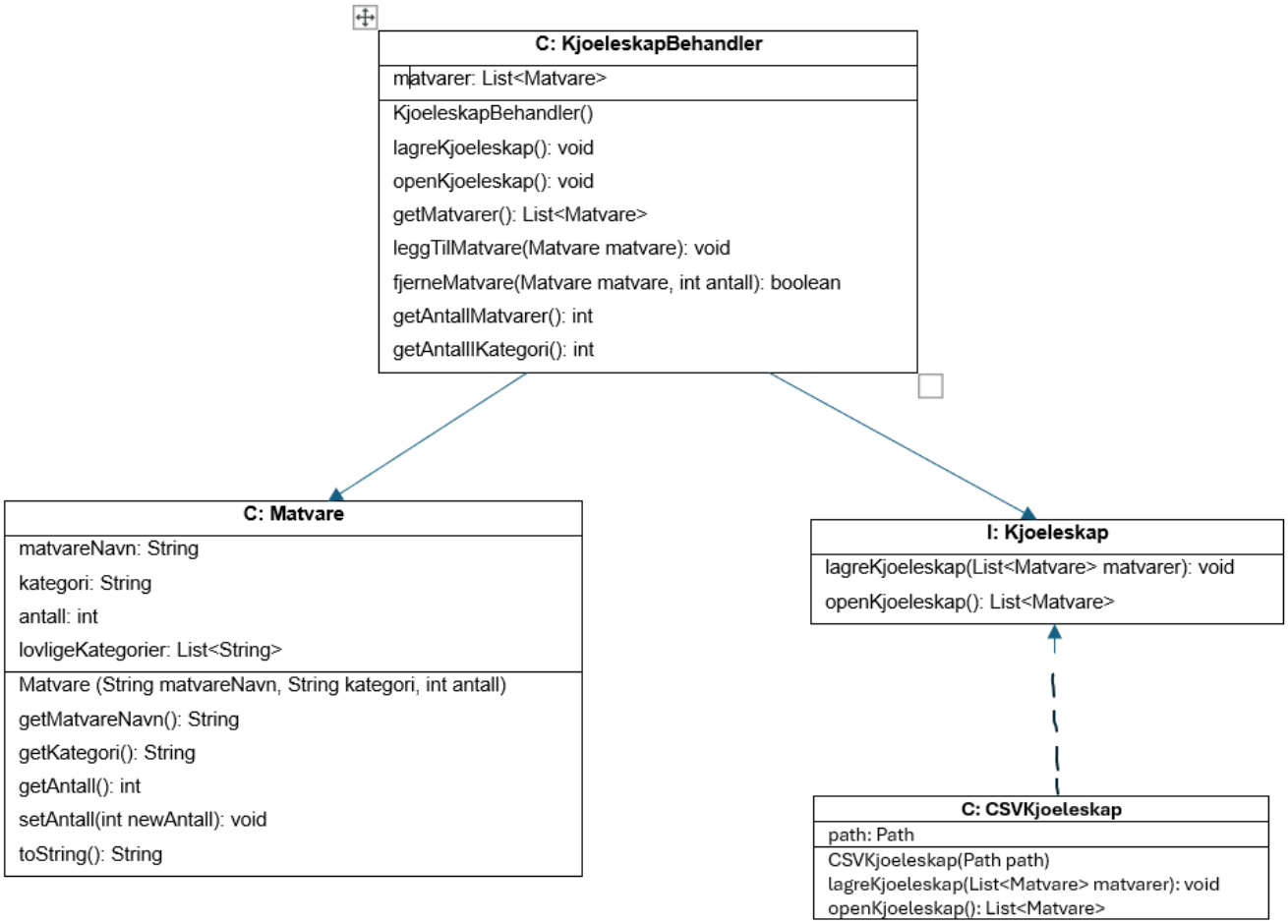
Beskrivelse av appen

Appen jeg har laget er en kjøleskap app som lar brukere få en oversikt over matvarer i kjøleskapet. Brukeren har mulighet å legge til matvarer etter navn, kategori og antall samt minke antallet eller fjerne hele matvaren. Når brukeren legger til en matvare kan den velge mellom kategoriene "Pålegg", "Drikke", "Grønnsaker", "Saus" og "Annet". Dersom brukeren legger til en matvare som allerede er i kjøleskapet og under samme kategori, vil antallet til denne matvaren øke. Brukeren har så mulighet til å lagre de nye matvarene i kjøleskapet og i tillegg laste inn matvarene som er lagt til i kjøleskapet fra før av. Dersom brukeren utfører en ugyldig handling som f.eks. å ikke legge til kategori, laste inn et tomt kjøleskap eller prøver å fjerne for mye av en matvare, vil dette også bli varslet om. I kjøleskapet har brukeren også en oversikt over antall matvarer i kjøleskapet samt hvor mange matvarer det ligger under de ulike kategoriene. Appen har et enkelt design som gir en oversiktlig og enkel oversikt over matvarer.

Diagram



Spørsmål

1. *Hvilke deler av pensum i emnet dekkes i prosjektet, og på hvilken måte? (For eksempel bruk av arv, interface, delegering osv.)*

   I dette prosjektet dekkes flere aspekter av pensumet i objektorientert programmering. Prosjektet mitt tar bla. i bruk prinsippene i objektorientert programmering ved å organisere koden i klasser og objekter som f.eks. "Matvare" klassen representerer en matvare med attributter og metoder. Det brukes også interface i form av "Kjoeleskap" som har metoder som lagrer og åpner kjøleskapet som klassen "CSVKjoeleskap" implementerer. "CSVKjoeleskap" klassen er også ansvarlig for filhåndtering som lagrer data til og laster data fra "kjoeleskap.csv" filen. I dette prosjektet er unntakshåndtering også en viktig del da den tar hensyn til feilsituasjoner som kan oppstå når brukeren bruker appen. I tillegg er det også brukt flere enhetstester for å sjekke at appen fungerer som den skal. FXML er også brukt i dette prosjektet for å definere brukergrensesnittet i appen gjennom "App.fxml". Gjennom prosjektet mitt har jeg også brukt lambda flere steder for å definere handlinger som skal utføres som svar på ulike hendelser og gir dermed enklere håndtering av hendelser i JavaFX-applikasjonen.

2. *Dersom deler av pensum ikke er dekket i prosjektet deres, hvordan kunne dere brukt disse delene av pensum i appen?*

   I appen dekkes flere spekter av pensum, men ikke alt. En av tingene som ikke blir brukt i mitt prosjekt er comparator/comparable. Dette kunne blitt lagt til for å sortere matvarer basert på navn, kategori og antall. Jeg kunne også brukt iterator/iterable for å tillate ekstern iterering over matvarene i kjøleskapet. En annen del av pensum jeg ikke har brukt er abstrakte klasser. Dette kunne jeg lagt til i prosjektet mitt for å gjøre appen lettere å vedlikeholde dersom jeg senere skulle utvide funksjonaliteten til appen. Dette kunne også vært aktuelt dersom jeg skulle definere felles funksjonalitet som brukes av flere relaterte klasser.

3. *Hvordan forholder koden deres seg til Model-View-Controller-prinsippet? (Merk: det er ikke nødvendig at koden er helt perfekt i forhold til Model-View-Controller standarder. Det er mulig (og bra) å reflektere rundt svakheter i egen kode)*

   Med hensyn til Model-View-Controller-prinsippet følger prosjektet mitt dette til en viss grad, men kunne også vært forbedret. Med hensyn til Model representerer "Matvare" klassen dette og inneholder dataene om matvarer i kjøleskapet. "KjoeleskapBehandler" klassen fungerer også som en del av Model da den håndterer mesteparten av logikken for å legge til, fjerne, lagre og laste matvarer. "App.fxml" filen definerer brukergrensesnittet for appen og hvordan de ulike elementene vises og interageres med brukeren og dermed også View. Med hensyn til Controller har "KjoeleskapController" ansvaret for kontrolleren i applikasjonen. Klassen kobler sammen modellen og visningen gjennom å håndtere brukerens interaksjoner og oppdatere modellen/visningen etter det i tillegg til oppdatering av brukergrensesnittet. En svakhet i prosjektet mitt er at deler av logikken ligger i "KjoeleskapController" klassen. Dette kan skape en mindre tydelig separasjon mellom Model og View og gjøre det vanskeligere å oppdatere/vedlikeholde senere.

4. *Hvordan har dere gått frem når dere skulle teste appen deres, og hvorfor har dere valgt de testene dere har? Har dere testet alle deler av koden? Hvis ikke, hvordan har dere prioritert hvilke deler som testes og ikke? (Her er tanken at dere skal reflektere rundt egen bruk av tester)*

   I testene mine er flere ulike scenarioer tatt hensyn til for å sjekke at koden er korrekt og at appen skal fungere på riktig måte. Når jeg skulle gå frem for å lage tester til appen, har jeg prioritert testing av de grunnleggende funksjonalitetene til appen. Dette innebar å teste konstruktøren i "Matvare" klassen, om matvarer ble lagt til og fjernet på riktig måte, sjekke at riktige unntak kastes når ugyldige verdier blir brukt samt om dataene lagres og lastes inn på riktig måte. Controller, og app-klassene er ikke testet for her, men de ulike funksjonene i "Matvare" og "KjoeleskapBehandler" er testet for da disse inneholder de grunnleggende

funksjonene i appen. Selv om testene dekker det viktigste, kan det legges til flere tester for å sjekke ulike scenarioer.