NTNU
Kunnskap for en bedre verden

Mappeoppgave
IDATx1003 Programmering 1 - Del 1

Høst 2024

Innhold

Dette dokumentet inneholder 3 deler:

1. Om mappevurdering - her finner du en detaljert beskrivelse av hvordan mappevurdering gjennomføres i emnet, hvilke deler mappen består av, og hvordan mappen skal leveres
2. Beskrivelse av prosjektet i sin ***helhet*** (kravspek)
3. Hva som skal gjøres i del 1 av 3 - **NB! LES DENNE GRUNDIG FØR DU STARTER MED KODINGEN!!**

Innholdsfortegnelse

Om mappevurdering i IDATx1003

I dette emnet er det ingen eksamen (skriftlig eller muntlig). I stedet for benytter vi **Mappe** som vurderingsform.

Mappen leveres ut ca uke 40 og går parallelt med øvrige øvingsaktiviteter ut semesteret.

Detaljert prosjektbeskrivelse blir publisert i 3 deler:

- **Del 1 - ca Uke 40**: Fokuserer på design av klasse, kodestil og dokumentasjon samt versjonskontroll og enhetstesting.
- **Del 2 - ca Uke 43**: Fokuserer på samling av objekter (ArrayList, HashMap osv), håndtering av samlinger og søk i samlinger (Lambda-uttrykk og streams/filter)
- **Del 3 - ca Uke 45**: Fokuserer på ferdigstillelse av applikasjon med et tekstbasert brukergrensesnitt med komplette enhetstester. I tillegg skal du utvide løsningen med egne forslag/ideer.

Hver ny del bygger på foregående del. Når ny del publiseres, vil du få mulighet til **muntlig tilbakemelding** på arbeidet du har gjort så langt, med mulighet å justere/endre/forbedre din løsning. Du vil få i alt 3 slike tilbakemeldinger, der den siste tilbakemeldingen gis i uke 47.

Endelig **fungerende** løsning/applikasjon leveres i GitHub/GitLab samt som ZIP-fil **sammen med en rapport** i **Inspera** innen fristen. (Se Blackboard for detaljer om GitHub/GitLab)

Vekting mellom rapport og prosjekt mot endelig karakter:

- Rapport - 30%
- Prosjekt - 70%

Kilder:

- Hva Forskriften sier om mappe som vurderingsform: https://lovdata.no/dokument/SF/forskrift/2015-12-08-1449/KAPITTEL_5#%C2%A75-11

Prosjektoppgaven - Matsvinn

NB! Her følger en beskrivelse av den endelige løsningen som skal leveres til slutt i prosjektet. Senere i dokumentet er det en detaljert beskrivelse av hva dere skal utvikle i hver av de 3 delene. Ikke start på utviklingen/kodingen før dere har lest hva som kreves i hver del!

Innledning

Figur 1 Illustrasjon

> *Matproduksjon står for over én tredjedel av menneskeskapte klimagassutslipp i verden.*
>
> *Dette er i stor grad fordi matproduksjon legger beslag på store landområder, bruker klimaintensive innsatsfaktorer som kunstgjødsel og kraftfôr, og fordi husdyrhold medfører store klimagassutslipp.*
>
> *Bare andelen mat som går i søpla står for 8-10 prosent av totale menneskeskapte klimagassutslipp! FNs bærekraftsmål sier derfor at vi trenger en halvering av matsvinnet før 2030 som et viktig tiltak for å begrense global oppvarming.*
>
> *En halvering av matsvinnet er faktisk vurdert som et av de billigste og mest effektive tiltakene vi har for å redusere våre klimagassutslipp.*
>
> *Sitat fra «Fremtiden i våre hender» 31/5-2022.*
> *https://www.framtiden.no/artikler/hvorfor-er-matsvinn-et-problem*

Du skal utvikle en applikasjon som bidrar til å gjøre forbrukerne litt mer bevisst på sitt eget matsvinn. Et godt sted å starte er å få oversikt over hvilke matvarer man har i eget hus, og hvordan man best kan benytte disse matvarene i egen matlaging før varene går ut på dato.

Avgrensninger

Applikasjonen som skal utvikles skal ha et **tekstbasert brukergrensesnitt** (TUI) i form av en meny eller kommandoer og kjøres i konsollet/terminalvinduet på din datamaskin.

Funksjonelle krav

Ønsket funksjonalitet i applikasjonen er inndelt i 2 nivå:

- **Nivå 1:** Et «Minimum Viable Product» nivå (MVP). Det er forventet at alle skal kunne klare å implementere dette.
- **Nivå 2:** Utvidet funksjonalitet. Her vil det gis noen utfordringer (fortsatt innenfor pensum) som kanskje ikke alle vil klare å implementere.

Matvarer er som regel lagret ulike steder i et hjem; kjøleskap, kjøkkenskap, hyller og skuffer. For å forenkle oppgaven noe, tenker vi oss at alle varer i denne applikasjonen oppbevares i et kjøleskap. Alternativt kan du velge å tenke "matlager" (engelsk: "FoodStorage") for en felels representasjon av alle lagerplassene i heimen.

Nivå 1: Minimum Viable Product (MVP)

Som bruker må jeg kunne:

- Opprette en ny vare/ingrediens (engelsk: *Grocery* eller *Ingredient*)
- Legge varen i kjøleskapet/varelageret (engelsk: *Fridge/FoodStorage*). Her må man kunne angi **mengden** av varen, f.eks. 4 liter med melk, 12 stk egg osv.
- Søke etter en vare (for å se om du har varen i kjøleskapet)
- Ta ut/fjerne en mengde av en vare fra kjøleskapet.
- Skrive ut en oversikt over varer i kjøleskapet
- Skrive ut en oversikt over varer som har gått ut på dato, og samlet verdi (i kroner) på disse varene.
- Beregne samlet verdi på samtlige varer i kjøleskapet.

Nivå 2: Utvidet funksjonalitet

Applikasjonen utvides fra Nivå 1 med følgende funksjonlalitet:

Som bruker må jeg kunne:

- Opprette en **matoppskrift** (for en matrett). En matoppskrift (engelsk: *Recipe*) består typisk av følgende elementer: Et **navn** på oppskriften, en kort **beskrivelse** av hva oppskriften lager, en **fremgangsmåte** og en liste av **ingredienser** (inkludert mengde).
- Sjekke om kjøleskapet inneholder nok varer/ingredienser til å lage matretten.
- Legge oppskriften inn i en **kokebok** for senere bruk.
- Få forslag til hvilke retter som kan lages fra rettene i kokeboken med varene/ingrediensene som finnes i kjøleskapet. (Avansert!)

Detaljer om vare/ingrediens

En vare/ingrediens kan bestå av følgende informasjon:

- Navn på vare, f.eks. «Lettmelk»
- Mengde av varen, f.eks. 1,5
- Måleenhet, f.eks. «liter»
- Best-før-dato (her kan du se litt på klassen java.util.Date, og java.text.SimpleDateFormater)
- Pris/kostnad i norske kroner pr enhet.

Noen ord om "mengde" og enhet

I denne applikasjonen må vi forholde oss til begrepene *mengde* og *enhet*. F.eks. 1.5 liter. En utfordring her er at oppskrifter ofte opererer med ulike enheter. F.eks.:

- 6 stk egg
- 4 dl melk
- 80 g smør
- ..osv

For å forenkle løsningen anbefaler vi at du tenker SI-enheter hele veien, både i forhold til mengde du har på lageret av en vare, og mengde som kreves i en oppskrift.

Typiske SI-enheter:

- Vekt: gram (g)
- Volum: liter (l)

For oppskrifter, vil det derfor være lurt å operere med liter i stedenfor desiliter, teskje, spiseskje osv. En spiseskje tilsvarer 15 ml, en teskje tilsvarer 5 ml f.eks.

Bruk av Kunstig Intelligens (KI)

Du er oppfordret til å benytte kunstig intelligens som verktøy, på lik linje med øvrige verktøy som hjelper deg i programmeringen. Verktøy som GitHub CoPilot, Jetbrains AI Assistant, ChatGPT osv kan være til stor hjelp og sparringpartner **når verktøyene brukes riktig!** IKKE bruk AI-verktøyene til å generere kode du ikke selv forstår.

Du **skal** dokumentere i rapporten hvordan du har benyttet deg av AI-verktøy, hvilke verktøy du har brukt, og gi eksempler på bruk i rapporten. Har du brukt "pronpting" for å få hjelp av AI-verktøyet, skal du skrive i rapporten prompten(e) du benyttet.

Mappen Del 1 (av 3)

I første del av mappen skal du fokusere på å:

- forstå oppgaven/prosjektet
- få opprettet prosjektet som et Maven-prosjekt
- sette prosjektet under versjonskontroll koblet til GitHub/GitLab
- implementere **entitetsklassen** (klassen som representerer en vare/ingrediens)
- implementere **testklasse** for å teste entitetsklassen.
- starte på rapporten

Detaljene følger under.

Kodeprosjekt

Du får tilgang til et ferdig oppsatt prosjekt via **GitHub Classroom** for mappen. Prosjektet er satt opp som et **Maven** prosjekt. For å få tilgang til dette prosjektet og få opprettet ditt eget repository for Mappen på GitHub, se BlackBoard og siden "Opprette Mappe-prosjektet fra GitHub" under menyen "Mappevurdering" i Black Board.

Når du har fått opprettet prosjektet og **klonet** prosjektet til din egen datamaskin, åpner du prosjektet i din IDE (IntelliJ, VSCode el.l.), og følger videre instrukser under (NB! DU kan ikke bruke BlueJ på dette prosjektet):

- Lage **entitetsklasse** for varetype/ingrediens (engelsk: "*Grocery*" eller "*Ingredient*")
- Dokumenter klassen grundig, inkludert
    - Rolle/ansvar
    - Hvilke informasjon klassen holder på og hvilke datatyper du har valgt for hver info og hvorfor (begrunnelse)
    - En vurdering av hvilke informasjon skal kun settes ved opprettelse av instans, og hvilken informasjon må kunne endres etter at instansen er opprettet
    - Hvordan klassen responderer på ugyldige data - hvilken strategi følger klassen (kaster unntak? setter dummy-verdi?)
- Implementer nødvendige felt i klassen, med egnede **datatyper**.
- Lag **enhetstester** for å teste at klassen for vare/ingrediens fungerer som den skal og er robust. Husk både *positive*- og *negative* tester.
- Kjør **CheckStyle** på koden din med **Google-reglene**.
- Opprett en klasse som på sikt skal bli brukergrensesnittet og dermed ta seg av all brukerinteraksjon. Opprett to metoder; **start()** og **init()**.
- Bruk start-metoden til å skrive **enkel testkode** for å teste at vare-klassen fungerer iht spek ved f.eks. å opprette 3-4 instanser av klassen, og skrive ut objektene (hint: Kan lønne seg å lage en egen metode for å skrive ut detaljene til en vare...).
- Opprett til slutt en klasse som skal være selve applikasjonen og som inneholder **public static void main(String[] args)**-metoden.
  Opprett main()-metoden. La denne metoden lage en instans av UI-klassen og kall deretter metodene **init()** og **start()** til UI-objektet.

Husk å utfør **commit (innsjekk)** til lokalt Git-repository jevnlig/hver gang du gjør endringer i koden. Husk å alltid legge til en kommentar til innsjekken. Husk også å utføre *push* til GitHub ved jevne mellomrom slik at du har koden din oppbevart trygt i skyen skulle du få problemer med datamaskinen din.

Rapport

Begynn på rapporten allerede nå:

- Sett deg inn i rapportmalen
- Fyll ut forside, og skriv innledningen
- Lag et **UseCase-diagram** som viser hvilken funksjonalitet løsningen skal tilby en bruker.
- Start på kapitlene om **teori** og **metode**.
- I **resultat**-kapitelet kan du allerede nå gi en kort beskrivelse av hvilke klasser du ser for deg du vil lage i prosjektet, og vise et klassediagram som viser avhengigheten mellom de.

Viktige sjekkpunkter

Når du løser oppgaven, bør du dobbeltsjekke følgende:

- Kompilering/bygging og prosjektstruktur:
  - Er prosjektet et Maven-prosjekt med en ryddig og riktig katalogstruktur?
  - Kan prosjektet **kompilere** uten feil (*mvn compile*)?
  - Er prosjektet plassert på fornuftig sted på din harddisk (finner du lett tilbake til prosjektet)?
- Versjonskontroll med git:
  - Er prosjektet underlagt versjonskontroll med sentralt repository i GitHub eller GitLab?
  - Finnes det flere innsjekkinger (commits) ?
  - Beskriver commit-meldingene endringene på en kort og konsis måte?
- Enhetstester:
  - Har enhetstestene beskrivende navn som dokumenterer hva testene gjør?
  - Følger de mønstret Arrange-Act-Assert?
  - Tas det hensyn til både positive og negative tilfeller?
  - Er testdekningen god nok?
  - Kjører samtlige tester uten feil?
- Er klassene for vare/ingrediens, brukergrensesnitt og applikasjon implementert iht oppgavebeskrivelsen?
- Kodekvalitet:
  - Er koden godt dokumentert iht JavaDoc-standard?
  - Er koden robust (validering mm)?
  - Har variabler, metoder og klasser beskrivende navn?
  - Er klassene gruppert i en logisk pakkestruktur?