Problemstilling

I denne oppgaven skal du utvikle en pasient-register applikasjon for et sykehus. Applikasjonen skal være komplett med et grafisk brukergrensesnitt (GUI) og med lagring av pasienter til fil (persistens). Oppgaven baserer seg på problemstillingen fra "Mappe - Del 1", med følgende krav og avgrensninger:

- Pasientregistret skal kun lagre informasjon om pasienter.
- Klassen for Pasient skal **ikke** arve fra Person (du lager **én** klasse for Pasient).
- Vi ser bort fra hvilken avdeling en pasient er innlagt.
- Vi ser bort ifra resepter og medisiner til en pasient
- Det skal være mulig å lagre pasientdata til fil og lese disse i etterkant
- Registret skal ha et grafisk brukergrensesnitt (GUI)
- Applikasjonen skal ha enhetstester for den delen av koden som er koblet mot forretningskritisk funksjonalitet
- Det skal brukes unntakshåndtering der det er nødvendig for å gjøre applikasjonen robust med tanke på feilsituasjoner som kan oppstå.

Følgende informasjon skal registreres om en pasient:

- Personnummer 11 siffer, som tekst (engelsk: Social Security Number)
- Fornavn (eng: first name)
- Etternavn (eng: last name)
- Diagnose (som tekst) (eng: diagnosis)
- Fastlege (navn på lege som tekst) (eng: general practitioner)

Oppgave 1: Maven og Git

Opprett et tomt Maven prosjekt og gi prosjektet en fornuftig groupId og artefactId. Prosjektet skal følge JDK v11 eller høyere. Når du svarer på kodeoppgavene under skal filer lagres iht standard Maven-oppsett: enhetstester legges i katalogen "test/java", eventuelle ressursfiler (bilder, konfigurasjon osv) legges i "main/resources", mens resten av koden hører hjemme i katalogen "main/java". Det skal være mulig å bygge og kjøre både applikasjon og tester med Maven uten feil.

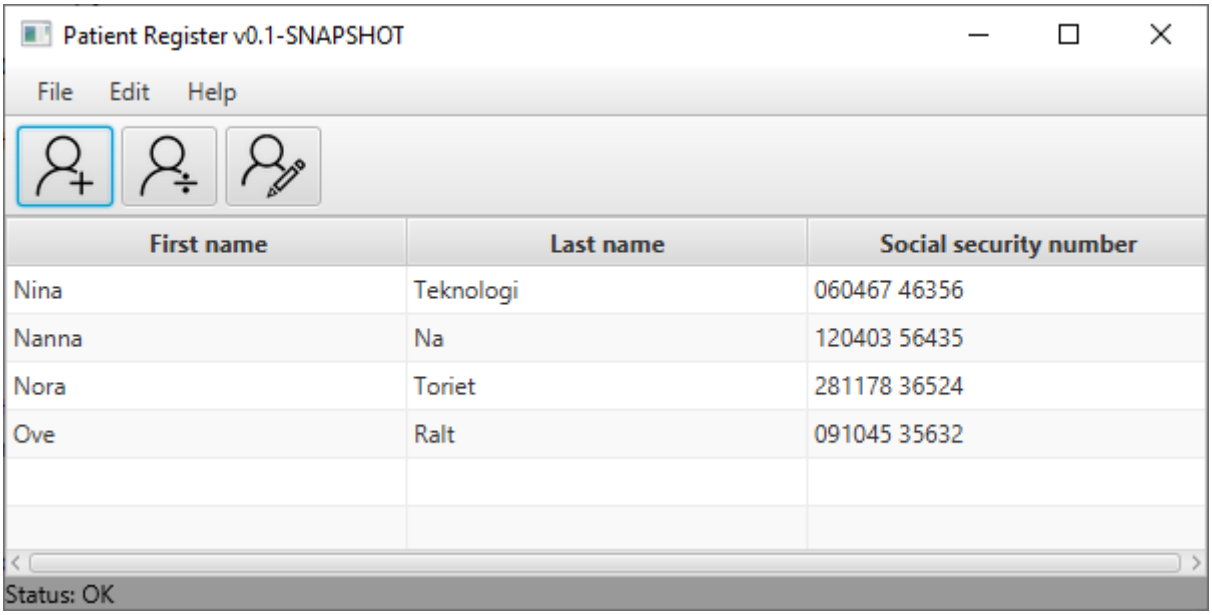
Legg så prosjektet under versjonskontroll:

- Først legger du prosjektet under lokal versjonskontroll
- Deretter oppretter du et nytt sentralt repo (tomt prosjekt) med samme navn på GitLab (for Ålesund: lag remote repo fra GitHub Classroom)
- Til slutt kobler du lokalt repo mot sentralt repo

For hver av oppgavene under skal du gjøre minst én innsjekk (commit) i lokal versjonskontroll, før du til slutt laster opp alle endringene til sentralt repo (push).

Oppgave 2: GUI

I denne oppgaven skal du designe og implementere det grafiske grensesnittet til applikasjonen. Du velger selv om du vil bruke FXML, eller kode GUI manuelt. Et forslag kan for eksempel se slik ut:

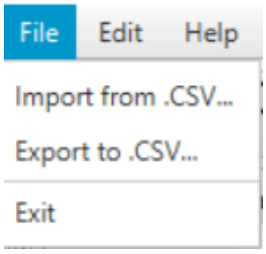


Du står fritt til å bestemme hvordan layoutene skal se ut, men vi stiller følgende krav til funksjonalitet:

1) Pasientene vises i en liste (TableView). Minimum informasjon som vises på skjermen skal være *First name, Last name og Social security number*. Status feltet nederst er frivillig, og du kan implementere det dersom du får tid. Den viser meldinger til brukeren. F.eks. etter import av CSV fil, kan en skrive «Import successful».

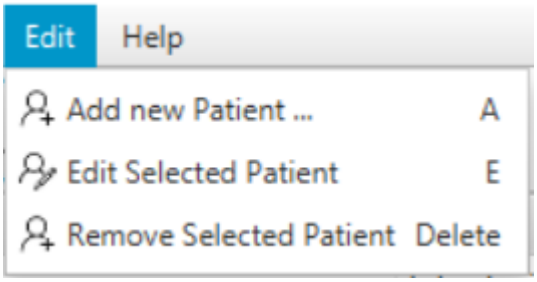
2) Applikasjonen skal ha en meny-linje (menu bar) med følgende valg (minimum): **File, Edit** og **Help**.

    a. **File** viser *Import from CSV*, *Export to CSV* og *Exit*. Disse funksjonene skal lese og skrive tilbake data i csv format (kolonne separator settes til «;») (Se Oppgave 4). Menyen kan se slik ut:

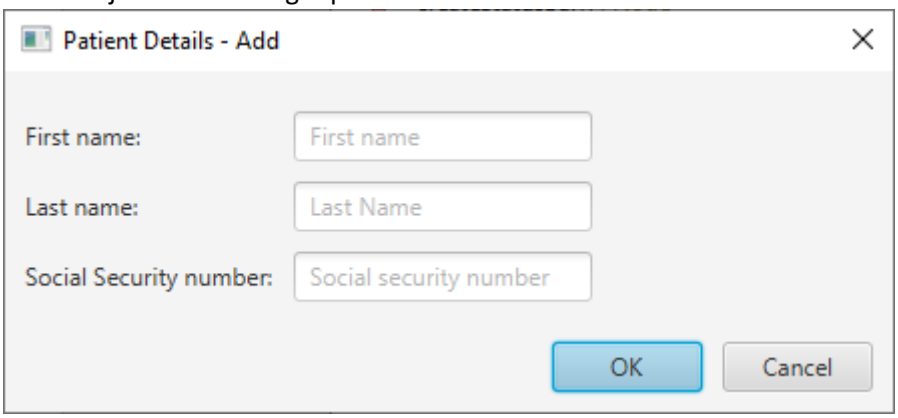


Exit-valget i menyen er frivillig å implementere.

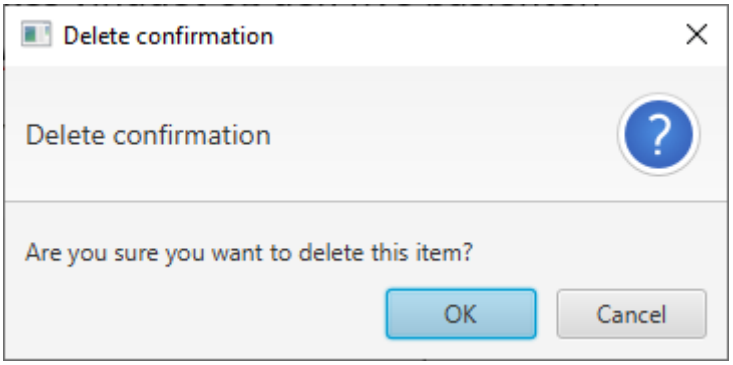
    b. **Edit** viser valgene *Add new patient*, *Edit selected patient*, og *Remove selected patient*. Menyen kan se slik ut (det er ingen krav til ikoner og short-keys som vist i eksempelet):



Add new patient viser et nytt vindu, der bruker kan skrive inn First name, Last name og Social Security number. Når en klikker på «OK», lukkes vinduet og den nye pasienten vises i listen i hovedvinduet. Når *Edit selected patient* velges, skal samme vindu åpnes og informasjon om den valgte pasienten vises der.

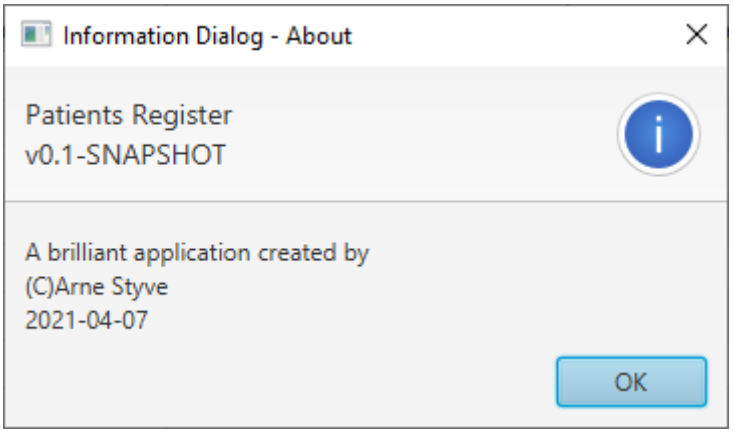


Dersom «*Delete selected patient*» velges, skal en dialogboks vises der bruker blir bedt om å bekrefte sletting. Klikk på «OK» fjerner valgt pasient fra hovedvinduet:



Hurtig-tastene i Edit-menyen er frivillig å implementere.

c. Når bruker klikker på «Help», skal det vises *About*. Velger man det, skal det komme informasjon om bla versjon av programmet. Du står fritt til å skrive ønsket tekst her:



3) I denne oppgaven skal du nå lage verktøy-menyen (toolbar). Du kan selv velge iconer for verktøyene:

Funksjonaliteten bak disse knappene skal være det samme som i forrige oppgave, altså *Add new patient*, *Edit selected patient*, og *Remove selected patient*. Du kan finne ikoner på nettet eller lage dine egne. Disse kan f.eks. være jpeg filer.

Oppgave 3: Design Patterns

Factory pattern kan brukes til å opprette instanser av objekter sentralt. I denne applikasjonen bygger du GUI gjennom å opprette forskjellige instanser av GUI-elementer som meny, status felt, verktøy linje, tabell osv. Anvend factory design pattern i applikasjonen slik at GUI elementer opprettes sentralt via en factory klasse.

Oppgave 4: Lagring/lesing av register data

1. I denne oppgaven må dere lese filer (fra meny - File-> Load Data). Skulle en bruker velge en annen filtype, må dere vise et vindu med en klar beskjed - “The chosen file type is not valid. Please choose another file by clicking on Select File or cancel the operation”.
2. Når en skal lagre data til en csv fil (fra meny - File->Save Data), skal dere be om et filnavn og plassering. Dersom det er en eksisterende fil med samme navn, skal dere vise en varsel som viser melding “A file with same name exists. Do you want to overwrite it?”. Med å trykke på “Yes” skal dere fjerne tidligere fil og lage en fil med samme navn. Når en velger “No”, skal dere be om et nytt fil navn.

Oppgave 6: Ekstra poeng – Database

Denne oppgaven skal gi dere ekstra poeng. Dersom dere ønsker å forbedre resultatet fra mappe-del-1, vil ekstra poeng fra denne oppgaven overføres til mappe-del-1 ;-)

1. I denne oppgaven skal dere lage en Derby Embedded database som skal lagre data i en tabell database. Persistence-unit skal ha navn “st-olavs-register”. Dere kan bruke “persistence.xml” gitt sammen med den oppgaven.
2. Databasen skal ha følgende informasjon
    a. Personnummer 11 siffer, som teskt (engelsk: Social Security Number) - Brukes som primær nøkkel
    b. Fornavn (eng: first name)
    c. Etternavn (eng: last name)
    d. Diagnose (som tekst) (eng: diagnosis)
    e. Fastlege (navn på lege som tekst) (eng: general practitioner)
3. Databasen skal være passord beskyttet. Se eksempelet i gitt persistence.xml
4. Hente ut alle pasienter med JPQL query. JPQL query dere skal bruke: “SELECT c FROM Patient c”

Vurderingskriterier

Når vi vurderer arbeidskravet “Mappe - Del 2” vektlegger vi følgende momenter:

- Maven:
    - Er prosjektet et Maven-prosjekt med fornuftige prosjekt-verdier og gyldig katalogstruktur?
    - Kan vi kjøre Maven-kommandoer for å bygge, installere, kjøre og teste uten at det feiler?
- Versjonskontroll med git:
    - Er prosjektet underlagt versjonskontroll med lokalt repo?
    - Er det lokale repoet koblet mot et sentralt repo?
    - Finnes det minst én commit per kodeoppgave?
    - Beskriver commit-meldingene endringene på en kort og konsis måte?
    - Har alle endringer blitt lastet opp til sentralt repo?
- Oppfylles kravene til det grafiske brukergrensesnittet (GUI)?
- Er filhåndtering (lesing og skriving av CSV-filer) implementert iht oppgavebeskrivelsen?
- Anvendes Factory-pattern til å opprette GUI-elementer, og er designmønstret kodet på en fornuftig måte?
- Er databaseløsningen satt opp iht oppgavebeskrivelsen og fungerer den som forventet?
- Enhetstesting:
    - Har forretningskritisk kode egne enhetstester?
    - Har enhetstestene beskrivende navn som dokumenterer hva testen gjør?
    - Følger de mønstret Arrange-Act-Assert?
    - Tas det hensyn til både positive og negative tilfeller?
- Kodekvalitet:
    - Er koden godt dokumentert med JavaDoc og kommentarer der det er fornuftig?
    - Er koden robust (verifiseres parametere før de brukes, håndteres unntak og feil på en rimelig måte mm)?
    - Har koden en fornuftig struktur og oppdeling, og er det lett å gjøre endringer og utvidelser i kodebasen?
    - Har variabler, metoder og klasser gode beskrivende navn?