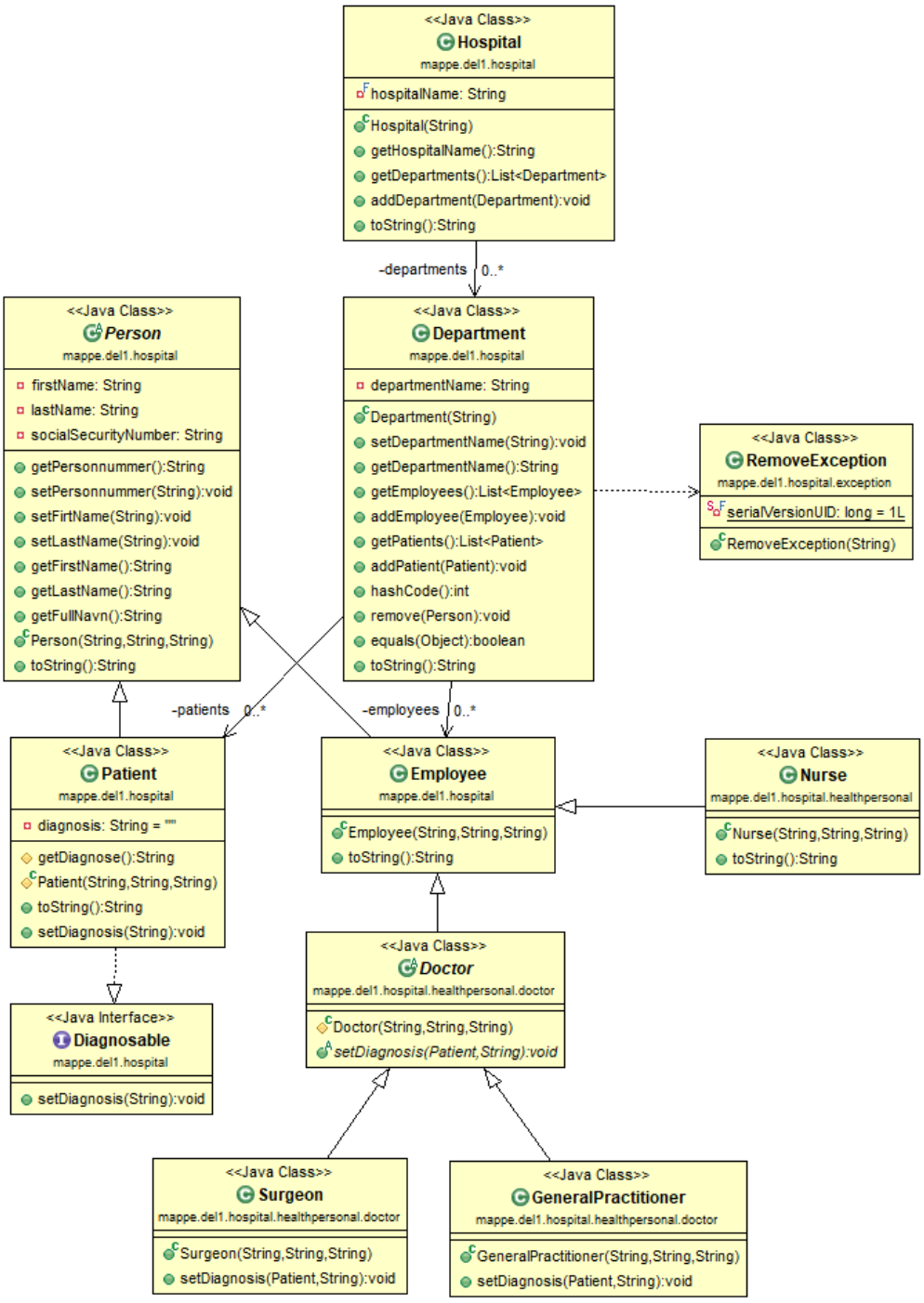
Problemstilling

Inspirasjon til problemstillingen er hentet fra hjemmesiden til St. Olavs Hospital (https://stolav.no/). Modellen vi skal jobbe med i denne oppgaven er forenklet for å tilpasses pensum og tidsrammen i denne innleveringen.

Definisjon hentet fra Store Medisinske Leksikon (https://sml.snl.no/sykehus): *Sykehus er en helseinstitusjon som er godkjent for innleggelse, undersøkelse og behandling av pasienter og fødende kvinner som trenger spesialisert helsetjeneste. Til sykehusene hører sengeavdelinger, poliklinikker og laboratorier. Tradisjonelt har man skjelnet mellom somatiske sykehus, som hovedsakelig behandler kroppens sykdommer, og psykiatriske sykehus, som behandler psykiske lidelser. Dette skillet er av mindre betydning nå i og med at psykiatriske avdelinger ofte inngår som en del av det totale sykehustilbudet.*

I denne oppgaven skal du kun jobbe med ett sykehus. Ett sykehusobjekt består av en liste med avdelinger og er identifisert med et navn. Hver avdeling består videre av en liste med ansatte og en liste med pasienter. Ansatte kan videre kategoriseres som helsepersonell og øvrige ansatte. Helsepersonell begrenses til allmennlege, kirurg og sykepleier. Det er kun allmennlege og kirurg som kan sette diagnose på en pasient. Du skal ikke lage en fullstendig klient i denne oppgaven, men en forenklet versjon som er spesifisert i oppgave 5.



Figur 1: Klassediagram for en forenklet sykehus-modell

Oppgave 1: Maven og Git

Det første du må gjøre er å opprette et tomt Maven prosjekt. Gi prosjektet en fornuftig groupId og artefactId. Prosjektet skal følge JDK v11 eller høyere. Når du svarer på kodeoppgavene under skal enhetstester legges i katalogen "test/java", mens resten av koden hører hjemme i katalogen "main/java". Det skal være mulig å bygge prosjektet og kjøre 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/GitHub (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).

Vi er nå klare for å skrive litt kode :-)

Oppgave 2: Implementasjon av personell-klassene

I denne oppgaven skal du programmere klassene for personer knyttet til sykehuset. Klassetreet skal altså bestå av klassene Person, Employee, Doctor, GeneralPractitioner, Surgeon, Nurse og Patient.

Klassene «Person» og «Doctor» defineres som abstract da det ikke gir mening å opprette objekter av disse klassene.

Det er kun klassene «GeneralPractitioner» og «Surgeon» som kan sette pasient diagnose. Sykepleiere eller øvrige ansatte skal altså ikke kunne sette diagnose på en pasient.

Interface-klassen «Diagnosable» har metoden setDiagnose som skal implementeres av «Patient»-klassen siden det er en pasient som kan få diagnose.

Når du har løst oppgave 2 sjekker du inn de relevante filene i lokal versjonskontroll. Commit-meldingen skal på en kort og konsis måte beskrive endringene du har gjort til nå i prosjektet (gjelder for alle commits). Merk: hvis du sjekker inn på slutten av en oppgave, men senere trenger å gjøre endringer i koden og sjekke inn på nytt, så er det selvfølgelig greit. Kravet er at du skal sjekke inn minst en gang per kodeoppgave.

Oppgave 3: Implementasjon av avdelingene for et sykehus

Et sykehus har et navn og en liste med Avdelinger (Du kan bruke *HashMap*, *ArrayList*… til å lage et register). Implementer klassene *Hospital* og *Department*. En ansatt eller pasient hører til en avdeling.

Når du har løst oppgaven sjekker du inn endringene i lokal versjonskontroll.

Oppgave 4: Implementasjon av remove-metoden i *Department*

I oppgave 4 skal du implementere en metode kalt *remove* i klassen *Department* som fjerner et objekt av typen *Patient* eller *Employee*.

Dersom objektet som sendes inn ikke finnes i registrene (pasienter eller ansatte), skal det kastes et unntak. Lag en *exception*-klasse kalt *RemoveException* som blir kastet av metoden *remove*. Unntaket skal være av typen "checked". Klassen som kaller *remove*-metoden skal fange opp dette og vise feilmeldingen.

Når du har implementert *remove*-metoden skal du skrive enhetstester som både verifiserer at metoden fungerer som forventet (positiv testing) og at uønsket input og uforutsigbar oppførsel håndteres på en god måte (negativ testing).

Til slutt sjekker du inn endringene i lokal versjonskontroll.

Oppgave 5: Implementer klienten *HospitalClient*

Du skal nå lage en enkel klient-klasse for å teste applikasjonen. Implementer disse funksjonene:

- Lag klienten *HospitalClient* som bruker *HospitalTestData.fillRegisterWithTestData(hospital)* til å fylle registrene med data (metoden *fillRegisterWithTestData* er gjengitt under).
- Kall *remove* metoden på *Department* for å fjerne en ansatt
- Kall *remove* metoden på *Department* for å fjerne en pasient som ikke finnes i listen. Bruk try-catch exception blokk for å håndtere situasjonen.

Når du har løst oppgave 5 sjekker du inn endringene i lokal versjonskontroll.
Til slutt pusher du alle lokale endringer til det sentrale repositoriet.

HospitalTestData:

```
import mappe.del1.hospital.healthpersonal.Nurse;
import mappe.del1.hospital.healthpersonal.doctor.GeneralPractitioner;
import mappe.del1.hospital.healthpersonal.doctor.Surgeon;

public final class HospitalTestData {

    private HospitalTestData() {
        // not called
    }

    /**
     * @param hospital
     */
    public static void fillRegisterWithTestData(final Hospital hospital) {

        // Add some departments
        Department emergency = new Department("Akutten");
        emergency.getEmployees().add(new Employee("Odd Even", "Primtallet", ""));
        emergency.getEmployees().add(new Employee("Huppasahn", "DelFinito", ""));
        emergency.getEmployees().add(new Employee("Rigmor", "Mortis", ""));
        emergency.getEmployees().add(new GeneralPractitioner("Inco", "Gnito", ""));
        emergency.getEmployees().add(new Surgeon("Inco", "Gnito", ""));
        emergency.getEmployees().add(new Nurse("Nina", "Teknologi", ""));
        emergency.getEmployees().add(new Nurse("Ove", "Ralt", ""));
        emergency.getPatients().add(new Patient("Inga", "Lykke", ""));
        emergency.getPatients().add(new Patient("Ulrik", "Smål", ""));
        hospital.getDepartments().add(emergency);

        Department childrenPolyclinic = new Department("Barn poliklinikk");
        childrenPolyclinic.getEmployees().add(new Employee("Salti", "Kaffen", ""));
        childrenPolyclinic.getEmployees().add(new Employee("Nidel V.", "Elvefølger", ""));
        childrenPolyclinic.getEmployees().add(new Employee("Anton", "Nym", ""));
        childrenPolyclinic.getEmployees().add(new GeneralPractitioner("Gene", "Sis", ""));
        childrenPolyclinic.getEmployees().add(new Surgeon("Nanna", "Na", ""));
        childrenPolyclinic.getEmployees().add(new Nurse("Nora", "Toriet", ""));
        childrenPolyclinic.getPatients().add(new Patient("Hans", "Omvar", ""));
        childrenPolyclinic.getPatients().add(new Patient("Laila", "La", ""));
        childrenPolyclinic.getPatients().add(new Patient("Jøran", "Drebli", ""));
        hospital.getDepartments().add(childrenPolyclinic);
    }
}
```

Vurderingskriterier

Når vi vurderer arbeidskravet "Mappe - Del 1" 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 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?
- Er personell-klassene implementert iht oppgavebeskrivelsen?
- Er klassene Hospital og Departments implementert iht oppgavebeskrivelsen?
- Fungerer remove-metoden i Hospital som forventet?
- Enhetstesting:
  - Har enhetstestene beskrivende navn som dokumenterer hva testen gjør?
  - Følger de mønstret Arrange-Act-Assert?
  - Tas det hensyn til både positive og negative tilfeller?
- Er klassen HospitalClient implementert iht oppgavebeskrivelsen?
- Kodekvalitet:
  - Er koden godt dokumentert med JavaDoc og kommentarer der det er fornuftig?
  - Er koden robust (verifiseres parametere før de brukes mm)?
  - Har variabler, metoder og klasser gode beskrivende navn?