Innleveringsoppgave 4

Innledning

I denne oppgaven skal du lage et sammensatt system for leger, legemidler, resepter og pasienter ved hjelp av klassene du har skrevet i tidligere innleveringer.

For å gjøre dette på en effektiv måte, kommer du til å trenge noen nye klasser. I tillegg vil oppgaven kreve at du gjør noen utvidelser i klassene du allerede har laget. Avhengig av hvordan du har løst tidligere innleveringer kan det også hende du må gjøre andre endringer i klassene dine for at de skal fungere slik dette oppgavesettet ber om.

Oppgave 1 – Itererbare lister

For å enkelt kunne løpe gjennom listene våre skal vi sørge for at de er *itererbare*. Dette skal gjøres ved å modifisere grensesnittet `Liste<E>` slik at det utvider Java-grensesnittet `Iterable<E>`.

(a) Sørg for at grensesnittet `Liste<E>` utvider `Iterable<E>`.

(b) Skriv klassen `LenkelisteIterator` som implementerer `Iterator<E>` og metodene `boolean hasNext()` og `E next()`.

Hint: Hvis Node-klassen er en indre klasse i `Lenkeliste<E>`, bør `LenkelisteIterator` også være det. Da trenger den heller ikke noen typeparameter.

(c) Utvid klassen `Lenkeliste<E>` med metoden `Iterator<E> iterator()` som returnerer et nytt `LenkelisteIterator`-objekt.

Relevante Trix-oppgaver
7.08 & *7.10*

Oppgave 2 – Klassen Pasient

(a) Skriv klassen `Pasient`.

En `Pasient` er en typisk bruker av resepter. Pasienten har et navn og et fødselsnummer (en tekststreng i vårt program). Når en ny pasient registreres, skal denne i tillegg få en unik ID. Pasienter har også en liste over reseptene de har fått utskrevet. Det skal være mulig å legge til nye resepter.

(b) Endre klassene som tar inn en `int pasientid` til å ta inn en `Pasient pasient`.

Oppgave 3 – Klassen Lege

Senere i oppgaven ønsker vi å kunne *sortere* leger.

(a) Utvid klassen `Lege` slik at den implementerer grensesnittet `Comparable<Lege>`. Leger skal kunne sorteres alfabetisk etter navn, slik at en lege ved navn "Dr. Paus" vil forekomme før "Dr. Ueland" etter sortering.

(b) Klassen `Lege` skal også kunne holde styr på hvilke resepter den har skrevet ut. Utvid klassen med en referanse `IndeksertListe<Resept> utskrevneResepter` og funksjonalitet for å hente ut denne listen av resepter.

(c) Klassen `Lege` skal ha metoder for å opprette instanser av de fire `Resept`-klassene man kan lage instanser av (hvit resept, p-resept, millitærresept og blå resept). Når et reseptobjekt opprettes, skal det legges inn i listen over legens utskrevne resepter før en referanse til objektet returneres.

Metodesignaturene skal se slik ut:

```
HvitResept skrivHvitResept (Legemiddel legemiddel, Pasient
 ↪  pasient, int reit) throws UlovligUtskrift { /* ··· */ }

MilResept skrivMilResept (Legemiddel legemiddel, Pasient
 ↪  pasient) throws UlovligUtskrift { /* ··· */ }

PResept skrivPResept (Legemiddel legemiddel, Pasient pasient,
 ↪  int reit) throws UlovligUtskrift { /* ··· */ }

BlaaResept skrivBlaaResept (Legemiddel legemiddel, Pasient
 ↪  pasient, int reit) throws UlovligUtskrift { /* ··· */ }
```

Om en vanlig lege prøver å skrive ut et narkotisk legemiddel, kastes unntaket `UlovligUtskrift`:

```
class UlovligUtskrift extends Exception {
    UlovligUtskrift(Lege l, Legemiddel lm) {
        super("Legen "+ l.hentNavn() + " har ikke lov til å
         ↪  skrive ut " + lm.hentNavn());
    }
}
```

Spesialister kan alltid skrive ut narkotiske legemidler, men bare på blå resept.

Hint: Du kan sjekke om et legemiddel er `Narkotisk` ved å bruke `instanceof`.

Relevant Trix-oppgave:
5.05

Oppgave 4 – Legesystem

Du skal nå programmere selve legesystemet. Programmet skal holde styr på flere lister med informasjon om legemidler, resepter, leger og pasienter. Det betyr at du må tenke gjennom hva som skjer når nye objekter som er avhengige av andre objekter, legges til.

Legesystemet skal benytte seg av listeklassene du skrev i en tidligere innlevering, så `ArrayList` og lignende fra Java-biblioteket skal ikke brukes. Du velger selv struktur for legesystemet så lenge det oppfyller kravene i deloppgavene. Der objektene kan identifiseres både med unik ID og navn (for eksempel når vi skal finne et legemiddel for å opprette en resept), velger du selv hva som er mest hensiktsmessig.

(a) Skriv en metode for å lese inn objekter fra fil. Følg filformatet i `legedata.txt` (publisert på semestersiden). Bruk metodene for å skrive resept i `Lege`-objektene for å opprette `Resept`-objekter. Dersom et objekt er ugyldig eller ikke følger filformatet i `legedata.txt`, skal det ikke legges inn i systemet.

Alle unntak som kastes må håndteres.

Merk at for at filformatet skal stemme må tellerne for unike ID-er starter på 0.

(b) Sørg for at brukeren får presentert en kommandoløkke som kjører frem til brukeren selv velger å avslutte programmet. Et eksempel på en brukerinteraksjon finnes i `interaksjon.txt` (publisert på semestersiden). Kommandoløkken skal presentere følgende valgmuligheter:

- Skrive ut en fullstendig oversikt over pasienter, leger, legemidler og resepter.
- Opprette og legge til nye elementer i systemet.
- Bruke en gitt resept fra listen til en pasient.
- Skrive ut forskjellige former for statistikk.
- Skrive alle data til fil.

(c) Implementer funksjonalitet for å skrive ut en ryddig oversikt over alle elementer i legesystemet. Leger skal skrives ut *sortert*.

(d) Legg til funksjonalitet for å la bruker legge til en lege, pasient, resept eller legemiddel. Resepter skal opprettes via en `Lege` sine metoder for å skrive resepter. Pass på at du sjekker om det er mulig å lage det ønskede objektet *før* det opprettes – for eksempel skal det ikke være tillatt å lage en resept uten en gyldig utskrivende lege. Dersom brukeren oppgir ugyldig informasjon, skal de informeres om dette.

Hint: For å finne ut om oppgitte data er gyldig, bør vi ta i bruk iteratoren vi har laget og lete i de relevante listene!

Du *bør* også gjøre fornuftige typesjekker underveis – for eksempel bør programmet gi en feilmelding og gå tilbake til hovedmenyen dersom en bruker forsøker å oppgi noe annet enn et tall som mengde virkestoff – men dette er ikke et krav.

Hint: Fang opp `NumberFormatException` ved behov!

(e) Legg til mulighet for å bruke en resept.

(f) Opprett funksjonalitet for å vise statistikk om elementene i systemet. Dette kan for eksempel presenteres som en «undermeny» av brukermenyen. Brukeren skal kunne se følgende informasjon:

- Totalt antall utskrevne resepter på vanedannende legemidler.
- Totalt antall utskrevne resepter på narkotiske legemidler.
- Statistikk om mulig misbruk av narkotika skal vises på følgende måte:
  - List opp navnene på alle leger (i alfabetisk rekkefølge) som har skrevet ut minst en resept på narkotiske legemidler, og antallet slike resepter per lege.
  - List opp navnene på alle pasienter som har minst en gyldig resept på narkotiske legemidler og, for disse, skriv ut antallet per pasient.

(g) Gi brukeren mulighet til å skrive alle elementer i det nåværende systemet til fil. Filen skal formateres på samme måte som filene som skal kunne leses. Du trenger ikke å lagre elementene sortert på ID, men merk at dersom du velger å gjøre det, kan du senere *lese fra samme fil som du skriver til*.

Relevant Trix-oppgave:
5.06