

IN1010 V19, Obligatorisk oppgave 4

Innleveringsfrist: Tirsdag 26.03. kl 23:59

Sist modifisert av [Hilmar Elverhøy](#).

Innledning

I denne oppgaven skal du lage et sammensatt system for leger, legemidler, resepter og pasienter ved hjelp av klassene du har skrevet i obligatorisk oppgave 2 og 3.

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 oblig 2 og 3 kan det også hende du må gjøre andre endringer i klassene dine for at de skal fungere som oppgaven ber om.

Spesielt i del D vil det være mange elementer som skal henge sammen. Her bør du bruke tid på å se for deg designet av hovedprogrammet og stille deg selv spørsmål om de neste stegene, for eksempel: Kan denne delen av programmet forenkles ved å plassere den i en metode (for eksempel når vi skal finne ut om en lege eksisterer)? Hvordan kan man enklest sjekke om brukerens input er gyldig?

Dersom du gjør antakelser om oppgaveteksten forventes det at du forklarer disse nøye under retting. Dersom du får rettet oppgaven tradisjonelt skal disse antakelsene være nøye dokumentert som kommentarer.

Del A: Klassen Pasient

A1: Skriv klassen Pasient.

En Pasient er en typisk bruker av resepter. Pasienten har et navn og et fødselsnummer-tekststreng. Når en ny pasient registreres skal denne i tillegg få en unik ID. Pasienter har også en liste over reseptene de har fått utskrevet. Siden pasienten ofte vil bruke en resept kort tid etter at den er utskrevet, bruker vi en *Stabel<Resept>* til å lagre pasientens resepter. Det skal både være mulig å legge til nye resepter og hente ut hele reseptlisten.

A2: Endre klassene som tar inn en *int pasientid* til å ta inn en *Pasient pasient*.

Del B: Itererbare lister

For å enkelt kunne løpe gjennom listene våre skal vi sørge for at de er itererbare. Dette skal gjøres "fra toppen" ved å modifisere grensesnittet fra *Liste<T>* slik at det utvider java-grensesnittet *Iterable<T>*. Det er her viktig å skille mellom grensesnittene *Iterable* og *Iterator*.

Iterable er et grensesnitt i Java som brukes av listene våre for å gjøre dem itererbare. Denne implementasjonen gjør blant annet at vi får lov til å skrive en *for each*-løkke som løper gjennom listen vår. Grensesnittet ser slik ut:

```
interface Iterable<T> {  
    Iterator<T> iterator();  
}
```

Iterator er et annet grensesnitt som beskriver selve *iterator-objektet* vi bruker for å gå gjennom listen. Siden alle lister ikke er like trengs det en egen *Iterator*-klasse som implementerer *Iterator<T>*, spesialtilpasset våre lister. Dette grensesnittet ser slik ut:

```
interface Iterator<T> {  
    boolean hasNext();  
    T next();  
    void remove();  
}
```

En fordel med implementasjonen av dette er at vi får tilgang til å bruke *for-each*-notasjon:

```
for (E e : elementliste) {  
    //Gjør noe med e ...  
}
```

... som egentlig er kortform for “så lenge iteratoren finner et neste element i listen (*hasNext()* returnerer true), hent det ut (*next()*).”

B1: Sørg for at grensesnittet *Liste<T>* utvider *Iterable<T>*.

B2: Skriv klassen *LenkelisteIterator* (*hint: Denne trenger ikke eget typeparameter*) som implementerer *Iterator<T>* og dermed metodene *boolean hasNext* og *T next*. Metoden *void remove()* er frivillig og trenger ikke å implementeres.

Hint: Hvis *Node*-klassen er en indre klasse i *Lenkeliste<T>* bør *iterator*-klassen også være det!

B3: Utvid klassen *Lenkeliste<T>* med metoden *Iterator iterator*, som returnerer et nytt *LenkelisteIterator*-objekt.

Relevante Trix-oppgaver: [7.06](#).

Del C: Klassen Lege

Senere i oppgaven ønsker vi å kunne sortere leger.

C1: Utvid klassen *Lege* slik at den implementerer grensesnittet *Comparable<Lege>* og dermed også metoden *compareTo*. Leger skal kunne sorteres alfabetisk etter navn, slik at en lege ved navn "Dr. Paus" kommer før (altså er mindre enn) "Dr. Ueland".

C2: Klassen *Lege* skal også kunne holde styr på hvilke resepter den har skrevet. Utvid klassen med en instans *Lenkeliste<Resept> utskrevdeResepter* og funksjonalitet for å hente ut denne listen av resepter.

C3: Modifiser *skrivResept()* slik at den tar inn et *Pasient* objekt i stedet for en pasientid, og når en resept opprettes, legges den til i *utskrevdeResepter* i legeobjektet som skriver den ut, og i pasientens stabel av resepter.

C4 (Frivillig): Erstatt *skrivResept()* med metodene *skrivHvitResept()*, *skrivBlaaResept()*, *skrivMillitærResept()*, og *skrivPResept()* som tar inn de relevante parameterene, og skriver ut en resept av riktig type på samme måte som *skrivResept()*

Relevante Trix-oppgaver: [7.01](#) & [9.01](#).

Del D: Legesystem

Du skal nå programmere selve legesystemet vårt. 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 listene du skrev i oblig 3. 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.

D1: Bruk kodeskjelettet [Legesystem.java](#). Implementer filinnlesning, og følg instruksjonene i filen for implementering. Merk at det kan hende du må gjøre endringer for å få dette til å passe med dine klasser fra oblig 2.

Om du allerede har implementert filinnlesning, står du fritt til å bruke denne, men det må modifiseres til å bruke *skrivResept()* metoden på legen, og bruke egne lister.

Filformatet er gitt av **vedlegg 2**. Merk at det ikke lenger tas inn hvilken type resept som skal opprettes.

Merk: For at filformatet skal stemme er det viktig at tellerne dine for unike ID-er starter på 0.

D2: Sørg for at brukeren får presentert en kommandoløkke som kjører frem til brukeren selv velger å avslutte programmet. Kommandoløkken skal presentere følgende valgmuligheter:

- Skrive ut en fullstendig oversikt over pasienter, leger, legemidler og resepter (deloppgave D3).
- Opprette og legge til nye elementer i systemet (deloppgave D4).
- Bruke en gitt resept fra listen til en pasient (deloppgave D5).
- Skrive ut forskjellige former for statistikk (deloppgave D6).
- Skrive alle data til fil (frivillig, deloppgave D8).

D3: Implementer funksjonalitet for å skrive ut en ryddig oversikt om alle elementer i legesystemet. Leger **skal** skrives ut i *ordnet rekkefølge*.

D4: Legg til funksjonalitet for å la bruker legge til en lege, pasient, resept eller legemiddel. Resepter skal opprettes via *Lege* sin *skrivResept()*. 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 oppgitt data er gyldig bør vi ta i bruk iteratoren vi har laget og lete etter dem 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!

D5: Legg til mulighet for å bruke en resept. Illustrasjon av foreslått interaksjon med bruker (fra bruker har indikert at de ønsker å bruke en resept) finner du nederst i oppgaven (*vedlegg 1*).

D6: 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 statistiske 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.

D7 (frivillig, krever C3): Legg til funksjonalitet til å velge hvilken type resept som blir laget under kommandoløkken

D8 (frivillig): Gi brukeren mulighet til å skrive alle elementer i det nåværende systemet til fil. Filen skal formateres på samme måte som innfil-eksempelet fra forrige deloppgave. Du trenger ikke å lagre elementene sortert på ID, men merk at dersom du velger å gjøre det kan du *lese fra samme fil som du skriver til*.

Relevante Trix-oppgaver: [9.01](#).

Oppsummering

Du skal levere klassene som utgjør hovedprogrammet ditt samt alle klasser som skal til for at hovedprogrammet skal fungere (inkludert både endrede og uendrede klasser fra obligatorisk innlevering 2 og 3).

Alle delene av programmet må kompilere og kjøre på Ifi-maskiner for å kunne få oppgaven godkjent. Unngå bruk av packages (spesielt relevant ved bruk av IDE-er som IntelliJ). *Ikke levér zip-filer!* Det går an å laste opp flere filer samtidig i Devilry.

Vedlegg

1) Foreslått interaksjon for bruk av resept:

```
Hvilken pasient vil du se resepter for?  
0: Anne (fnr 12121212121)  
1: Johnny (fnr 32323232323)  
> 1
```

```
Valgt pasient: Johnny (fnr 32323232323).  
Hvilken resept vil du bruke?  
0: Prozac (3 reit)  
1: Ibux (2 reit)  
2: Paracet (0 reit)  
> 1
```

```
Brukte resept paa Ibux. Antall gjenvaerende reit: 1
```

```
Hovedmeny:  
[...]
```

```
Hvilken pasient vil du se resepter for?  
0: Anne (fnr 12121212121)  
1: Johnny (fnr 32323232323)  
> 1
```

```
Valgt pasient: Johnny (fnr 32323232323).
```

```
Hvilken resept vil du bruke?  
0: Prozac (3 reit)  
1: Ibux (1 reit)  
2: Paracet (0 reit)  
> 2
```

```
Kunne ikke bruke resept paa Paracet (ingen gjenvaerende reit).
```

```
Hovedmeny:  
[...]
```

2) Filformat for innlesing av fil:

```
# Pasienter (navn, fnr)  
Jens Hans Olsen, 11111143521  
Petrolina Swiq, 24120099343  
Sven Svendsen, 10111224244  
Juni Olsen, 21049563451  
# Legemidler (navn, type, pris, virkestoff [, styrke])  
Predizol, a, 450, 75, 8  
Paralgin Forte, b, 65, 400, 5  
Placebo Pianissimo, c, 10, 0  
Ibux, c, 240, 200  
# Leger (navn, kontrollid / 0 hvis vanlig lege)  
Dr. Cox, 0  
Dr. Wilson, 0  
Dr. House, 12345  
Dr. Hillestad Lovold, 0  
# Resepter (legemiddelNummer, legeNavn, persID, reit)  
1, Dr. Cox, 2, 3  
2, Dr. Hillestad Lovold, 3, 10000  
0, Dr. House, 1, 1337  
3, Dr. Hillestad Lovold, 3, 2
```