IN1010 V20, Obligatorisk oppgave 4

Innleveringsfrist: Tirsdag 24.03. kl 23:59

Sist modifisert av Hilmar Elverhøy. V1.1

Gruppeoppgave

Denne obligatoriske oppgaven skal leveres som gruppeoppgave. Man kan melde inn en gruppe i devilry, eller fylle ut nettskjema for å bli satt på en gruppe. Vi anbefaler ikke å gjøre denne oppgaven alene, siden arbeidsmengden ar tiltenkt grupper på 3-4 medlemmer.

Innledning

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

For å gjøre dette på en effektiv måte kommer dere til å trenge noen nye klasser. I tillegg vil oppgaven kreve at dere gjør noen utvidelser i klassene dere allerede har laget. Avhengig av hvordan dere har løst oblig 2 og 3 kan det også hende dere 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 dere 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 dere gjør antakelser om oppgaveteksten forventes det at dere forklarer disse nøye under retting. Dersom dere får rettet oppgaven tradisjonelt skal disse antakelsene være nøye dokumentert som kommentarer.

Del A: Gruppearbeid

Denne deloppgaven trenger ikke å gjøres av de som jobber alene.

I denne deloppgaven skal dere bli enige om hvordan gruppearbeidet på deres gruppe skal foregå. For hver del skal dere levere et notat om hva dere kom frem til. Denne delen må besvares for å få godkjent, men innholdet vil ikke påvirke bedømmelsen

A1: Snakk sammen innad i gruppen om hva dere forventer å bidra med, og hva dere forventer av de andre på gruppen Diskuter også ambisjonsnivå for oppgaven. Noter hva dere kommer frem til på stikkordsform.
Hint: Se på presentasjon om smidig utvikling for å se hvordan dette kan gjøres

A2: Diskuter samarbeidsform. Vil dere sitte sammen og jobbe, eller fordele oppgaver og jobbe individuelt? Sett krav til når hver del av oppgaven skal være ferdig, og husk å beregne ekstra tid til uforutsette hendelser. Noter på stikkordsform hva dere kommer frem til.

Diskuter også hva slags verktøy dere vil bruke til å samarbeide. Det er viktig at alle er komfortable med verktøyene dere bruker. Vanlige verktøy for samarbeid er git/github, https://codeshare.io/, eller google docs. Man kan også finne en egen løsning som passer for gruppen.

A3: Velg hvilke løsninger dere skal basere dere på. Dere alle skal ha løsninger på de delene av oppgaven som bygger på tidligere obliger, men kun bruke en av dem i denne obligen. Hva er fordeler og ulemper med løsningen dere velger? Noter dette på stikkordsform, og nevn hvem sin løsning som er valgt for de forskjellige tidligere løsningene. Sett dere inn i den valgte løsningen.

Del B: Klassen Pasient

B1: 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.

B2: Endre klassene som tar inn en *int pasientid* til å ta inn en *Pasient pasient.*

Del C: 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) {
      //Gjoer noe med e ...
}
```

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

C1: Sørg for at grensesnittet *Liste<T>* utvider *Iterable<T>*.

C2: 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!

C3: Utvid klassen Lenkeliste<T> med metoden *Iterator iterator*, som returnerer et nytt LenkelisteIterator-objekt.

Relevante Trix-oppgaver: 7.04 & 7.06

Del D: Klassen Lege

Senere i oppgaven ønsker vi å kunne sortere leger.

D1: 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".

D2: Klassen Lege skal også kunne holde styr på hvilke resepter den har skrevet. Utvid klassen med en instans *Lenkeliste<Resept> utskrevedeResepter* og funksjonalitet for å hente ut denne listen av resepter.

D3: Lege skal ha metoder for å opprette instanser av de fire Reseptklassene 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 utskrevene resepter, før de returneres.

Metodesignaturene skal se slik ut:

```
public HvitResept skrivHvitResept(Legemiddel legemiddel, Pasient pasient, int reit) throws
UlovligUtskrift;
```

```
public MillitaerResept skrivMillitaerResept(Legemiddel legemiddel, Pasient pasient, int
reit) throws UlovligUtskrift;

public PResept skrivPResept(Legemiddel legemiddel, Pasient pasient) throws UlovligUtskrift;

public BlaaResept skrivBlaaResept(Legemiddel legemiddel, Pasient pasient, int reit) throws
UlovligUtskrift;
```

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

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

(denne klassen må også legges til i besvarelsen din)

Spesialister kan alltid skrive ut Narkotiske legemidler

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

Relevante Trix-oppgaver: 7.01 & 9.01.

Del E: 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 dere må tenke gjennom hva som skjer når nye objekter som er avhengige av andre objekter legges til.

Legesystemet skal benytte seg av listene dere 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 dere selv hva som er mest hensiktsmessig.

E1: Skriv en metode for å lese inn objekter fra fil. Følg filformatet i vedlegg 2. Bruk *skrivResept*-metodene i legeobjektet for å opprette Resept objekter. Dersom et objekt er ugyldig, skal det ikke legges inn i systemet.
Hint: Husk å behandle unntak som kan kastes.

Filformatet er gitt av **vedlegg 2**.

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

E2: 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 E3).
Opprette og legge til nye elementer i systemet (deloppgave E4).
Bruke en gitt resept fra listen til en pasient (deloppgave E5).
Skrive ut forskjellige former for statistikk (deloppgave E6).
Skrive alle data til fil (deloppgave E8).

E3: Implementer funksjonalitet for å skrive ut en ryddig oversikt om alle elementer i legesystemet. Leger **skal** skrives ut i *ordnet rekkefølge*.

E4: 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 dere 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!

E5: 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 dere nederst i oppgaven ***(vedlegg 1).***

E6: 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.

E7: Legg til funksjonalitet til å velge hvilken type resept som blir laget under kommandoløkken

E8: 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 dere velger å gjøre det kan dere *lese fra samme fil som dere skriver til*.

Relevante Trix-oppgaver: 9.01.

Del F: Erfaringsnotat

Om man gjør oppgaven alene erstattes F1-F3 med F5. Grupper trenger ikke besvare del F5.

I denne deloppgaven skal dere forklare deres erfaringer, og hva dere vil ta med dere til senere gruppeinnleveringer og teamarbeid. Del F leveres som et tekst- eller pdf-dokument. Denne delen må besvares for å få godkjent, men innholdet vil ikke påvirke bedømmelsen

Skriv MAKS en side til oppgave F1-F3.
F1: Hva gikk bra?
F2: Hva ville dere gitt dere gjort annerledes neste gang?
F3: Hva har dere fått ut av gruppearbeidet
F4 (Frivillig): Hvordan kan organiseringen av gruppearbeidet forbedres? Var det noe som burde informeres mer om, ble det lagt godt nok opp til?
F5 (Ikke for grupper): Hvordan var det å løse denne oppgaven individuelt? Hva er fordelene og ulempene med å løse oppgaver i team?

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). Det skal også leveres tekst- eller pdf-dokumenter for del A og del F.

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,narkotisk,450,75,8
Paralgin Forte,vanedannende,65,400,5
Placebo Pianissimo,vanlig,10,0
Ibux,vanlig,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,pasientID,type,[reit])
1,Dr. Cox,2,hvit,7
2,Dr. Hillestad Lovold,3,blaa,1000
0,Dr. House,1,millitaer,12
3,Dr. Hillestad Lovold,3,p,
```