

IN1010 V19, Obligatorisk oppgave 2

Innleveringsfrist: **Tirsdag 26.02 kl 23.59**

Introduksjon

I de obligatoriske oppgavene fremover skal du lage et system som holder styr på leger, pasienter, resepter og legemidler. I denne første oppgaven skal du opprette objekter og skrive klasser for legemidler, resepter og leger og gjøre relevante tester av disse.

Det forventes at du skriver ryddig og lesbar kode, og det anbefales at du skriver fornuftige kommentarer underveis slik at du selv har oversikt over hvordan de forskjellige delene av programmet fungerer.

Notat om ID

Noen av objektene i denne obligen skal kunne identifiseres ved en ID. Med ID menes her et unikt, ikke-negativt heltall.

Det første objektet som opprettes av en bestemt klasse skal ha id = 0, det andre id = 1, det tredje id = 2, osv. Merk at ID-ene ikke er unike på tvers av alle klassene, men hvert objekt av en bestemt type skal ha en id som ingen andre objekter av den samme klassen har. Det vil si at ingen legemidler har samme ID, og ingen resepter har samme ID, men en resept og et legemiddel kan ha samme ID.

Del A: Legemidler og Preparater

Legemidler deles inn i tre kategorier, som kalles Preparat A, Preparat B, eller Preparat C. Et preparat har et navn, en ID og en pris. I tillegg må vi for alle preparater kunne vite hvor mye virkestoff (mg) det inneholder totalt. Prisen og virkestoffet skal lagres som flyttall.

Et legemiddel **skal** være en av subtypene Preparat A, narkotisk, Preparat B, vanedannende, eller Preparat C, vanlig legemiddel. Det er stor forskjell på legemidler av disse tre typene, men i denne oppgaven skal vi bare ta hensyn til følgende krav for de forskjellige typene legemidler:

- *Preparat A* har et heltall som sier hvor sterkt narkotisk det er.
- *Preparat B* har et heltall som sier hvor vanedannende det er.
- *Preparat C* har ingen tilleggsegenskaper (annet enn klassens navn).

A1: Tegn klassehierarkiet beskrevet ovenfor. Du trenger bare å ha med navn på klasser og sammenhengen mellom disse. Lever tegningen som en bildefil eller som PDF.

A2: Skriv klassene *Legemiddel*, *PreparatA*, *PreparatB* og *PreparatC*. De tre sistnevnte klassene arver den førstnevnte. Konstruktøren til *Legemiddel* (og dermed også *PreparatC*)

skal ta inn String navn, double pris og double virkestoff (i den rekkefølgen). Konstruktørene til *PreparatA* og *PreparatB* skal i tillegg ta inn *int styrke*.

Legemiddel skal ha metodene *hentId*, *hentNavn*, *hentPris* og *hentVirkestoff* som returnerer de relevante verdiene. I tillegg skal klassen ha metoden *settNyPris*.

PreparatA har i tillegg metoden *hentNarkotiskStyrke*, mens *PreparatB* har metoden *hentVanedannendeStyrke*.

A3: Skriv et testprogram *TestPreparat*. I denne klassen skal du opprette ett objekt av hver av subklassene du har skrevet. Deretter skal du gjøre enkle enhetstester der du tester alle egenskapene til en instans før du går videre og gjør det samme for neste instans.

A4 (frivillig, men sterkt anbefalt): Overskriv *toString()* metoden i *Preparatene* slik at du lett kan skrive ut all tilgjengelig informasjon om objektene.

Tips 1: For hver test bør du beskrive en forventning til resultatet av testen, slik at du gir utskrift til brukeren avhengig av om du fikk forventet resultat eller ikke.

Tips 2: Det kan være lurt å skille ut testene i statiske metoder inne i *TestPreparat*-klassen.

Relevante Trix-oppgaver: [3.02](#), [3.03](#), [3.04](#) & [4.10](#)

Del B: Resepter

En Resept har en ID. I tillegg skal en resept ha en referanse til et legemiddel, en referanse til den legen som har skrevet ut resepten, og ID-en til den pasienten som eier resepten. En resept har et antall ganger som er igjen på resepten (kalles reit). Hvis antall ganger igjen er 0, er resepten ugyldig.

I denne oppgaven skal vi forholde oss til ulike typer resepter. De to hovedkategoriene er **hvite** og **blå** resepter.

Hvite resepter

Hvite resepter har i seg selv ingen nye egenskaper (utover et annet klassenavn). Derimot finnes det to subtyper av hvit resept: Militærresepter og P-resepter.

- **Militærresepter** utgis til vernepliktige i tjeneste. Som en forenkling sier vi at militærresepter **alltid gir en 100% rabatt** på prisen til et legemiddel.
- **P-resepter** gir unge en rabatt på prevensjonsmidler. Denne rabatten er statisk og gjør at brukeren betaler 108 kroner mindre for legemiddelet. *Merk* at brukeren aldri kan betale mindre enn 0 kroner. I tillegg til rabatten har P-resepter den egenskapen at de alltid utskrives med 3 reit.

Blå resepter

Det er stor forskjell på vanlige (hvite) og blå resepter (blant annet er utstedelsen av en blå resept forbundet med en del kontroller), men igjen skal vi gjøre en forenkling og si at bare prisen som betales er forskjellig: Blå resepter er alltid sterkt subsidiert, og for enkelhets skyld sier vi her at de har 75% rabatt slik at **pasienten må betale 25% av prisen på legemidlet.**

B1: Tegn klassehierarkiet beskrevet ovenfor. Du trenger bare å ha med navn på klasser og sammenhengen mellom disse. Lever tegningen som en bildefil eller som PDF.

B2: Skriv klassen `Resept` og dens subklasser. Konstruktøren i `Resept` skal ta inn et `Legemiddel` legemiddel, en `Lege` utskrivende `Lege`, en int `pasientId` og int `reit` (i den rekkefølgen). **Merk: Vi skal ikke kunne opprette en instans av selve klassen `Resept`**, kun av subtypene.

Klassen `Resept` skal ha følgende metoder som henter relevant data: `hentId`, `hentLegemiddel` (henter tilhørende `Legemiddel`), `hentLege` (henter utskrivende `Lege`), `hentPasientId` og `hentReit`.

I tillegg skal klassen ha følgende metoder:

- `public boolean bruk`: Forsøker å bruke resepten én gang. Returner `false` om resepten alt er oppbrukt, ellers returnerer den `true`.
- `abstract public String farge`: Returnerer reseptens farge. Enten "hvit" eller "blaa".
- `abstract public double prisAaBetale`: Returnerer prisen pasienten må betale.

B3: Skriv et program `TestResepter` der du oppretter instanser av de forskjellige klassene. Du vil også trenge å opprette noen objekter av klassen `Legemiddel` for å gjøre dette.

B4 (frivillig, men sterkt anbefalt): Overskriv `toString()` metoden i resept objektene slik at du lett kan skrive ut all tilgjengelig informasjon om objektene.

Tips: Du vil også ha behov for en instans av klassen `Lege`. Selv om du ikke har skrevet denne klassen enda kan du opprette en forenklet (tom) versjon av `Lege` foreløpig.

På samme måte som i Del A skal du nå gjøre enhetstester av instanser av de forskjellige klassene. For hver instans holder det at du tester egenskapene som er implementert forskjellig fra reseptens foreldreklasse.

Relevante Trix-oppgaver: [4.01](#), [4.02](#), [4.03](#) & [4.11](#).

Del C: Leger

Merk: Relevante temaer for å løse del C undervises først onsdag 13. februar.

Konstruktøren i Lege tar kun inn en String med legens navn. Lege skal ha en metode for å hente ut et navn og skrive ut en resept.

Noen leger er Spesialister. Spesialister har fått godkjenningsfritak til å skrive ut resept på narkotiske legemidler. Å ha godkjenningsfritak kan gjelde for andre enn leger, så dette skal implementeres som et grensesnitt (interface). Alle som har godkjenningsfritak, har en kontroll ID, som kan hentes ut for å sjekke at godkjenningsfritaket ikke blir misbrukt.

Spesialist skal arve fra Lege og skal i tillegg implementere følgende grensesnitt (dette grensesnittet må du legge ved som en fil i innleveringen din):

```
public interface Godkjenningsfritak {  
    public int hentKontrollID();  
}
```

Lege skal ha metoden *Resept skrivResept(Legemiddel legemiddel, int pasientID, int reit)* throws *UlovligUtskrift*, som lager et reseptobjekt med legemiddel, pasientID, og en referanse til samme legen, og returnerer det. Du kan velge selv hva slags resept som skrives ut.

Om en vanlig lege prøver å skrive ut et narkotisk legemiddel (altså av typen *PreparatA*), 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)

C1: Tegn klassehierarkiet beskrevet ovenfor (inkludert grensesnittet). Du trenger ikke å ta med metoder eller annet innhold i klassene/grensesnittet. Lever tegningen som en bildefil eller som PDF.

C2: Skriv klassene Lege og Spesialist som beskrevet over. Konstruktøren i Spesialist skal i tillegg til navn også ta imot en int kontrollID.

C3 (frivillig, men sterkt anbefalt): Overskriv *toString()* metoden i Lege og Spesialist slik at du lett kan skrive ut all tilgjengelig informasjon om objektene.

Relevante Trix-oppgaver: [5.03](#) & [5.04](#).

Del D: Integrasjonstest

I denne delen skal du lage et hovedprogram som gjør det vi vil kalle en minimal integrasjonstest - vi skal altså teste hvordan de forskjellige delene av systemet fungerer sammen. Hovedprogrammet skal gjøre følgende:

- Opprette minimum en instans av hver eneste klasse og la disse inneholde nødvendige referanser til andre objekter.
- Skrive ut relevant informasjon om hvert enkelt objekt. (Her vil det lønne seg å ha overskrevet toString() metoden i alle klassene du har skrevet).
- **(frivillig, men anbefalt):** La integrasjonstesten lese inn data fra en fil. Bruk filformatet som er gitt i vedlegget.

I tillegg skal du lage en datastrukturtegning som illustrerer objektene og deres tilstand når du har kjørt dette testprogrammet. Ved behov, se [notatet om datastrukturer](#) eller repeter [Trix-oppgaver fra oblig 1](#).

Vedlegg

Filformat for innlesing av fil:

```
# 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 ikke )
Dr. Cox, 0
Dr. Wilson, 0
Dr. House, 12345
Dr. Hillestad Lovold, 0
# Resepter (type, legemiddelNummer, legeNavn, persID, [reit])
blaa, 1, Dr. Cox, 2, 3
hvit, 2, Dr. Hillestad Lovold, 3, 10000
prevensjon, 0, Dr. House, 1
militaer, 3, Dr. Hillestad Lovold, 3, 2
```