# IN1010 V20, Obligatorisk oppgave 2

***Innleveringsfrist:*** *Tirsdag 18.02 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

Legemidler deles inn i tre kategorier, narkotisk, vanedannende og vanlige legemidler. Et legemiddel har et navn, en ID og en pris. I tillegg må vi for alle legemidler kunne vite hvor mye virkestoff (mg) det inneholder totalt. Prisen og virkestoffet skal lagres som flyttall.

Et legemiddel **skal** være en av subtypene Narkotisk, Vanedannende, eller 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:

- *Narkotisk* legemiddel har et heltall som sier hvor sterkt narkotisk det er.
- *Vanedannende* legemiddel har et heltall som sier hvor vanedannende det er.
- *Vanlig* legemiddel 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*, *Narkotisk*, *Vanedannende* og *Vanlig*. De tre sistnevnte

klassene arver den førstnevnte. Konstruktøren til *Legemiddel* (og dermed også *Vanlig*) skal ta inn String navn, double pris og double virkestoff (i den rekkefølgen). Konstruktørene til *Narkotisk* og *Vanedannende* 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*.

*Narkotisk* har i tillegg metoden *hentNarkotiskStyrke*, mens *Vanedannende* har metoden *hentVanedannendeStyrke*.

***Merk:* Det skal ikke være mulig å opprette en instans av klassen *Legemiddel, kun subtypene***

**A3:** Skriv et testprogram *TestLegemiddel*. I denne klassen skal du opprette et 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 Legemiddlene 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 *TestLegemiddel*-klassen.

*Tips 3:* Kompiler ofte! Jo mindre kode du må sjekke for feil hver gang du kompilerer, jo lettere blir det. Det er et krav at du skal kompilere før du leverer!

**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), men det skal være mulig å opprette instanser av dem. 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:*** 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 utskrivendeLege, 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 PResept skal **ikke** ta inn *int reit* i konstruktøren.

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.

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.

*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.

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

**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 12. februar.*

Konstruktøren i Lege tar kun inn en String med legens navn. Lege skal ha en metode for å hente ut navnet til legen.

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. Når du kopierer tekst fra pdf vil det komme noen spesialtegn med, så vi anbefaler å skrive av selv):

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

**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).

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.