Innleveringsoppgave 2

Legemidler og resepter

I denne og de to neste frivillige oppgavene 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. Du skal også gjøre relevante tester av disse.

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

Legemidler og resepter

Noen av objektene i denne oppgaven 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 klasse 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.

Oppgave 1 – Legemidler

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

Et legemiddel skal være en av subklassene `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`
  Et *narkotisk* legemiddel har et heltall som sier hvor sterkt narkotisk det er.
- `Vanedannende`
  Et *legemiddel* har et heltall som sier hvor vanedannende det er.

- `Vanlig`
  Et *vanlig* legemiddel har ingen tilleggsegenskaper (annet enn klassens navn).

(a) 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.

(b) Skriv klassene `Legemiddel`, `Narkotisk`, `Vanedannende` og `Vanlig`. De tre sistnevnte klassene arver fra den førstnevnte. Konstruktøren til `Legemiddel` (og dermed også `Vanlig`) skal ta inn `String navn`, `int pris` og `double mengdeVirkestoff` (i den rekkefølgen). Konstruktørene til `Narkotisk` og `Vanedannende` skal i tillegg ta inn `int styrke`. Alle disse instansvariablene, unntatt `pris`, skal være `public` og `final`. Også `id`-en skal være `public` og `final`. På denne måten kan alle deler av programmet ha tilgang til dem, men de kan ikke forandres.

`Legemiddel` skal ha metodene:

- `public int hentPris()`
- `public void settNyPris(int pris)`

Merk at det *ikke* skal være mulig å opprette en instans av klassen `Legemiddel`.

(c) 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 egenskapene til en instans av `Narkotisk`, før du går videre og gjør det samme for en instans av `Vanedannende` og til slutt for en instans av `Vanlig`.

Du kan teste så mange egenskaper du vil i denne oppgaven, men et minimum er at du tester at ID-ene blir riktig.

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

Det kan være lurt å skille ut testene i metoder inne i `TestLegemiddel`-klassen. En enkel test kan se slik ut:

```
private static boolean testLegemiddelId(Legemiddel
 ↪  legemiddel, int forventetLegemiddelId) {
    return legemiddel.id == forventetLegemiddelId;
}
```

(d) Overskriv `toString()`-metoden i klassen `Legemiddel` og dens subklasser slik at du lett kan skrive ut all tilgjengelig informasjon om objektene.

Relevante Trix-oppgaver:
03.02, *03.03* & *03.05*

Oppgave 2 – 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, uttales re-it). 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 hvite resepter. Det finnes også to subklasser av hvite resepter:

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

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 kun 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 bare må betale 25% av prisen på legemidlet. Alle priser rundes av til nærmeste hele krone.

Hint: Til avrunding kan du bruke `(int) Math.round(...)`.

(a) 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.

(b) Skriv klassen `Resept` og dens subklasser. Konstruktøren i `Resept` skal ta inn argumentene:

- `Legemiddel legemiddel`
- `Lege utskrivendeLege`
- `int pasientId`
- `int reit`

I den rekkefølgen. Merk at vi ikke skal kunne opprette en instans av selve klassen `Resept`, kun av subklassene. Klassen `MilitærResept` skal *ikke* ta inn `int reit` i konstruktøren.

Klassen `Resept` skal ha følgende metoder som henter relevant data:

- `public int hentId()`
- `public Legemiddel hentLegemiddel()`
- `public Lege hentLege()`
- `public int hentPasientId()`
- `public int 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`. Instansvariabelen `reit` må oppdateres.
- `abstract public String farge()`
  Returnerer reseptens farge, enten `hvit` eller `blå`.
- `abstract public int prisÅBetale()`
  Returnerer prisen pasienten må betale.

(c) 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 Oppgave 1 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 superklasse.

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 versjon av `Lege` foreløpig (for eksempel en tom klasse).

(d) Overskriv `toString()`-metoden i `Resept`-objektene slik at du lett kan skrive ut all tilgjengelig informasjon om objektene.

Relevante Trix-oppgaver:
04.01, *04.02* & *04.03*

Oppgave 3 – Leger

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*. Alle som har godkjenningsfritak, har en kontrollkode, 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 uforandret som en fil i innleveringen din):

```
interface Godkjenningsfritak {
    String hentKontrollkode();
}
```

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

(b) Skriv klassene `Lege` og `Spesialist` som beskrevet over. Konstruktøren i `Spesialist` skal i tillegg til navn også ta imot en `String kontrollkode`.

(c) Overskriv `toString()`-metoden i `Lege` og `Spesialist` slik at du lett kan skrive ut all tilgjengelig informasjon om objektene.

Relevante Trix-oppgaver:
05.03 & *05.04*

Oppgave 4 – 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 én 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.)

Oppgave 5 – Datastruktur

Lag en datastrukturtegning (ikke ta med metoder) som illustrerer objektene og deres tilstand om du hadde laget et veldig enkelt hovedprogram som bare opprettet ett `Legemiddel`-objekt, ett `Lege`-objekt og ett `Resept`-objekt. Legemidlet skal være narkotisk, resepten skal være en militærresept skrevet ut på dette legemiddelet og legen er en spesialist (som har skrevet ute denne resepten). Pass på å også få med klassedatastrukturene på tegningen din.

Relevante Trix-oppgaver:
03.07, *03.08*, *03.09* & *03.10*