INF1010 - Innleveringsoppgave 6

Frist: Onsdag 16. mars, 10:00

Maks 6 poeng

Om obligatorisk oppgave 4, 6 og 7 i INF1010, våren 2016: "Leger og resepter"

Du skal jobbe med en problemstilling omkring leger og resepter i 6 uker fra 19. februar til 30. mars og du kan til sammen få maksimum 16 poeng. Den første delen av oppgaven (oblig 4), som du skal jobbe med de første to ukene (maks 4 poeng), skal leveres senest 24. februar kl 10:00. Den 24. februar frigis den andre del av oppgaven (oblig 6). Denne skal leveres senest 16. mars kl 10:00 og gir maks 6 poeng. Den siste delen (oblig 7), er det meningen at du skal begynne å jobbe med 2. mars. Den skal leveres senest 30. mars kl 10:00, og har også maks 6 poeng.

Siden disse tre oppgavene bygger på hverandre er det bare mulig å hoppe av underveis. Det er ikke mulig å hoppe på, dvs. at du må ha godkjent 4 for å få godkjent 6, og du må ha godkjent 6 for å få godkjent 7. Du har godkjent en oppgave om du får ett eller flere poeng.

I hovedsak består oblig 4 av å lage et klassehierarki, oblig 6 av å lage en del beholdere, og oblig 7 av å sette det hele sammen til slutt til et ordrestyrt program. I oblig 6 og oblig 7 må du kanskje gå tilbake til klassehierarkiet i oblig 4 og gjøre små forandringer.

Beholdere til Leger og resepter

I denne oppgaven skal dere vise at dere kan bruke grensesnitt, abstrakte klasser, generiske klasser, subklasser, lenkelister og tabeller. Dere skal programmere en del beholdere som skal brukes i oblig 7. Mange av disse beholderene skal kunne brukes til andre formål, men for å ikke gjøre oblig 6 for abstrakt er det kanskje lurt å ha bruken i oblig 7 klart for øyet hele tiden. På gruppene vil dere få hjelp til å lage iteratorer. Når dere implementerer iteratorer behøver dere ikke implementere `remove()`.

De beholderene dere skal programmere i oblig 6 skal i oblig 7 ta vare på alle legemidler, resepter, leger og pasienter. I tillegg skal en lege ha en beholder over alle reseptene vedkommende har skrevet ut, og en pasient skal ha en beholder som inneholder alle pasientenes resepter. Legemidlene skal lagres i et objekt av klassen `Tabell`, reseptene skal lagres i et objekt av klassen `EnkelReseptListe`, legene skal lagres i et objekt av `SortertEnkelListe` og pasientene skal lagres i et objekt av klassen `Tabell`. Beholderen som inneholder en leges resepter skal være av klassen `EldsteForstReseptListe`, mens beholderen som inneholder en pasients resepter skal være av klassen `YngsteForstReseptListe`.

Les gjennom hele oppgaven før du starter å svare på noen spørsmål. Du må ha klar oversikt over resten av oppgave 6 før du besvarer oppgavene 6.1.

Oppgave 6.1

6.1 A Tegn opp klassehierarket for beholderne. Ikke ta med navn på metoder og variabler.

6.1 B Tegn opp datastrukturen.

Tegn alle beholderne, noen legemiddel-objekter, noen lege-objekter, noen pasient-objekter og noen resept-objekter. La det komme klart frem at en resept er med i mange beholdere. Tegn inn *public* metodene (grensesnittet) i alle beholderene. Ikke ta med andre metoder. Når det er flere like objekter behøver du bare tegne inn all variabler i ett objekt.

I oppgave 6.2 og 6.3.a frigjør vi oss fra leger, resepter mm. og definere generelle grensesnitt og klasser for beholdere

Oppgave 6.2: Grensesnitt (interface) for beholdere

Skriv programmet for det generiske grensesnittet `AbstraktTabell`. Det skal ikke være noen restriksjoner på hva slags elementer den abstrakte tabellen skal kunne inneholde. AbstraktTabell beskriver en beholder og du skal kunne:

- Sette et objekt inn i tabellen på en oppgitt plass (indeks). Metoden returnerer sann eller usann avhengig om operasjonen gikk bra eller ikke
- Finne et objekt basert på en indeks
- Iterere over listen

Skriv programmet for det generiske grensesnittet `AbstraktSortertEnkelListe`. En slik liste skal bare kunne inneholde elementer som implementerer grensesnittene `Comparable` (med seg selv) og `Lik`. En slik liste skal kunne:

- Sette inn et nytt element (i sortert rekkefølge, minste først)

- Finne et element basert på en nøkkel av typen `String`
- Itereres over, slik at innholdet kan bli listet opp i sortert rekkefølge, minste først

Oppgave 6.3: Klasser for beholdere

a) Generiske klasser

Skriv den generiske klassen `Tabell` som implementerer `AbstraktTabell`. Klassen skal lagre alle elementene i en array, og arrayens lengde skal oppgis som parameter til konstruktøren.

Hvis du har lyst og tid: Når du setter noe inn i tabellen og det ikke er plass, skal du lage en ny array som er lang nok (innenfor rimelighetens grenser), og så kopiere alle elementene over til den nye arrayen.

Skriv den generiske klassen `SortertEnkelListe` som implementerer `AbstraktSortertEnkelListe` som en enveisliste. Du skal programmere denne listen selv, ikke bruke klasser fra Java-biblioteket.

b) Ikke-generiske klasser

Skriv klassen `EnkelReseptListe`. Klassen `EnkelReseptListe` skal inneholde en envisliste med en peker til første og en peker til siste element i listen. Klassen skal kunne ta vare på resepter, og en resept må kunne være med i flere objekter av denne klassen. Metodene i klassen skal kunne sette inn en resept og finne en resept basert på reseptnummeret. Hvis resepten som det letes etter ikke finnes i listen, skal det kastes et unntak. Skriv også en iterator over listen. Igjen skal du programmere denne listen selv, ikke bruke klasser fra Java-biblioteket.

Skriv subklassene `EldsteForstReseptListe` og `YngsteForstReseptListe`. Når du itererer over den første klassen skal du starte med den eldste resepten (den som ble satt inn først) og gå mot yngre (de som ble satt inn sist). Når du itererer i den andre klassen, skal du starte i den yngste enden.

Hint: Forskjellen på de to subklassene til klassen `EnkelReseptListe` kan være bare metoden som setter inn en resept.

Utfordring: I både `SortertEnkelListe` og `EnkelReseptListe` skal du skrive en iterator. Kan du klare å bruke den samme iteratoren i begge klassene? Da må de to listene kanskje ha de samme nodene?

Oppgave 6.4: Enkle enhetstester for alle beholderene

Skriv et lite testprogram for hver av klassene `Tabell`, `SortertEnkelListe`, `EldsteForstReseptListe` og `YngsteForstReseptListe`. Prøv å lage programmene slik at både vanlige tilfeller og noen spesialtilfeller blir testet.

Innleveringen på denne oppgaven er tegningene fra oppgave 6.1, alle grensesnittene og klassene, og de fire kjørbare programmene fra oppgave 6.4

Krav til innleveringen

1. Klassenavnet og filnavnet skal være identisk.
2. Klassenavn skal skrives med stor forbokstav.
3. Variabelnavn skal ha liten forbokstav.
4. Oppgaven må kunne kompilere og kjøre på IFI sine maskiner.
5. .class filer skal ikke leveres.
6. Ikke bruk æ, ø eller å i .java-filene(heller ikke som kommentarer eller utskrift).
7. Filene skal inneholde gode kommentarer som forklarer hva programmet gjør.
8. Programmet skal inneholde gode utskriftssetninger som gjør det enkelt for bruker å forstå.
9. Metodenavn skal skrives med liten forbokstav.
10. Koden skal være riktig indendert. Er du usikker, se Appendix J i Big Java.
11. Hver klasse skal ligge i sin egen .java-fil.

Fremgangsmåte for innleveringer i INF1010

1. Lag en fil som heter README.txt. Følgende spørsmål skal være besvart i filen:
   1) Hvordan synes du innleveringen var? Hva var enkelt og hva var vanskelig?
   2) Hvor lang tid (ca) brukte du på innleveringen?
   3) Samarbeidet du med noen under innleveringen? Hvis ja, skriv brukernavn på den/de du samarbeidet med.
   4) Var det noen oppgaver du ikke fikk til? Hvis ja:
      a) Hvilke(n) oppgave er det som ikke fungerer i innleveringen?
      b) Hvorfor tror du at oppgaven ikke fungerer?
      c) Hva ville du gjort for å få oppgaven til å fungere hvis du hadde mer tid?
2. Logg inn på Devilry.
3. Lever .java-filene, bilde av klassehierarkiet og README.txt i *samme innlevering*.
4. Husk å trykke lever og sjekk deretter at innleveringen din er komplett.