Obligatorisk oppgave nr. 3 (av 4) i INF1000, V-2007

Leveringsfrist

Oppgaven må leveres senest **fredag 23. mars kl 16.00** via det elektroniske innleveringssystemet.

Viktig: les slutten av oppgaven for detaljerte leveringskrav. **NB**. Det presiseres at du minimum skal ha de 4 klassene beskrevet i hintene for å få godkjent løsningen din. En løsning uten minst disse klassene godkjennes ikke.

Formål

Trening i å løse et litt større programmeringsproblem ved å kombinere de ulike elementene vi har sett på til nå (arrayer, metoder, klasser/objekter, brukerinteraksjon, filbehandling, m.m.).

Generell informasjon

Denne obligatoriske oppgaven skal løses individuelt. Besvarelsen må godkjennes som bestått for at du skal kunne gå opp til eksamen, men bidrar ikke til selve eksamenskarakteren. Studenter som har godkjent obligatorisk oppgave 3 fra et tidligere semester trenger ikke levere på nytt, men bør sjekke at de er oppført i listen over tidligere godkjente innleveringer. Gjennomføring og innlevering av oppgaven skal skje i henhold til gjeldende retningslinjer, se

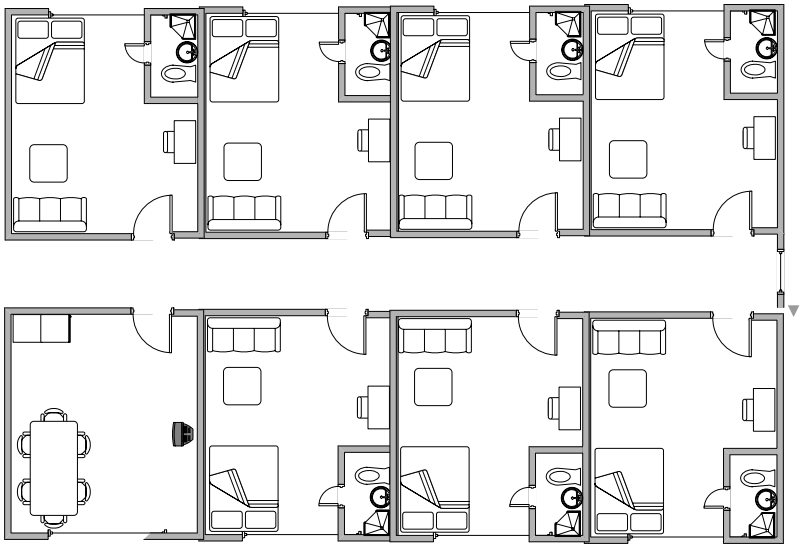
http://www.ifi.uio.no/studinf/skjemaer/erklaring.pdf (norsk)
http://www.ifi.uio.no/studinf/skjemaer/declaration.doc (engelsk)

Enhver innlevering av besvarelse på denne oppgaven tas som en bekreftelse på at retningslinjene er lest og forstått.

Oppgave

I likhet med de fleste andre universitetsbyer har Ruritanias hovedstad Uqbar dessverre fortsatt mangel på studenthybler. Gulbrand Grå og hans firma HaiHus A/S lever av denne mangelen. Du skal hjelpe Gulbrand å lage et system for å administrere utleie og effektiv innkreving av husleie. Besvarelsen skal anta at systemet settes i drift fra og med mars 2007.

I denne oppgaven skal vi se på hans enetasjes studenthus Utsyn. Utsyn består av 4 separate ganger (1-4), hver med 7 hybler + fellesarealer. Hver hybel er unikt hybelnavn som består av gangnr + en bokstav (B-H). Fellesarealet i hver gang har bokstaven A. Da er 2A er fellesarealet i gang 2, mens 3G er hybel G på gang 3. Nedenfor vises de 7 smakfulle hyblene på gang 1 og fellesarealet (de andre 4 gangene er identiske):



Programmet skal være menystyrt, dvs. det skal skrive ut på skjerm en meny over mulig kommandoer og be brukeren om å gi en kommando. Programmet skal gi en feilmelding hvis brukeren taster inn feil. Menyen skal gå i en løkke helt til brukeren taster inn tallet 0 for å avslutte.

Vi skal ha følgende menyvalg i systemet:

0. Avslutt
1. Skriv liste over ledige hybler
2. Registrer ny leietager
3. Registrer frivillig flytting av leietager
4. Månedskjøring av husleie med strøm
5. Registrer betaling fra leietager
6. Sjekk om noen leietagere skal kastes ut
7. Skriv økonomirapport

Programmet skal ved oppstart lese inn status fra en fil :"HaiHus.data", for alle hyblene altså hvilke som er ledige og opptatte, samt studentens navn og husleiestatus (hvor mye de er i pluss eller minus med husleia). På denne filen lagres også summer av strømregningene, studentenes ordinære husleieinnbetalinger og deres husleieforskudd hittil siden systemet ble tatt i drift i tillegg til all annen informasjon som må "overleve" mellom to kjøringer av programmet. Når brukeren velger Avslutt, skal programmet skrive alle disse opplysningene tilbake til samme fil slik de da har blitt endret. Dette gjør at Gulbrand ikke vil tape data hvis strømmen går eller maskinen slås av. Formatet på denne fila: "HaiHus.data", kan du bestemme selv, men her er et forslag du godt kan benytte:

Filen har først 28 linjer, en for hver hybel, der hver linje er på følgende form:

```
int gang; char bokstav; String studentnavn; int saldo;
```

Et eksempel på en linje kan være:

```
2; C; Albert Aalesund; 2339;
```

For tomme hybler settes studentnavn lik "TOM HYBEL" og saldo er mindre eller lik 0 (hvordan den kan bli negativ er beskrevet under). Etter dette skal det stå en linje med format :

```
int totaltAntallMåneder; int antallHybelmånederMedTommeHybler; int totalFortjeneste
```

Det nest siste tallet teller totalt antall hybelmåneder med ledige hybler – har programmet gått i 2 måneder med 22 ledige hybler i den første og 13 i den neste, viser dette tallet 35. Det siste tallet skal akkumulere totalfortjeneste på all utleievirksomhet. Inn i dette beløpet skal det først regnes 3000 kr fortjeneste pr. utleid hybel pr. måned, mellomlegget mellom hva studentene betaler for strømmen og hva han selv betaler videre til strømleverandør, og evt. torpedogebyr og sluttgebyr (se under). Siden Gulbrand regner med å alltid få inn det noen skylder ham, oppdaterer vi totalfortjenesten hver gang vi oppdaterer skyldig beløp og ikke når vi registrerer betalt beløp.

Gulbrand har beregnet at avskrivninger og faste kostnader, kommer på 1000 uqbarske kroner per hybel per måned. Han tar kr. 4000 per hybel per måned, ***pluss*** strøm for forrige måned.

Gulbrand får en månedlig spesifisert strømregning fra elverket Lysmegopp A/S. Regningen kommer på fil med følgende format:
`husnr` (dette huset er nr 42)`: gangnr:bokstav for areal`(A-H)`:antall kwt (kilowatt-timer) forbruk`

Et eksempel på en linje i fila: **`42:1:A:365`**

(her har altså fellesarealet i gang nr 1 et strømforbruk på 365 kw). Det er altså 32 linjer på denne fila, hvorav 28 angår hyblene. Lysmegopp krever 80 øre per kilowatttime(kwt), mens Gulbrand krever 2 kroner av studentene per kwt (noe må han jo ha igjen for strevet og fordi Gulbrand selv også dekker strømmen til

fellesarealene). Strømregningen brukt på hver hybel legges direkte til i husleien. Siden dette skal gjøres automatisk (dvs. via et menyvalg), trenger ikke programmet å ta vare på tidligere måneders strømregninger.

Når en hybel blir ledig og ikke blir leid ut den påfølgende måned, legges strømregningen likevel til saldoen (**`utestående`**) for hybelen. På denne måten kan saldoen bli negativ for en ledig hybel. Systemet er altså at man i første måned betaler utestående strøm fra forrige leieboer, mens man til gjengjeld slipper å betale siste måned i sin egen leieperiode.

For de ulike menyvalgene gjelder:

- **Skriv liste over ledige hybler**
  En enkel liste med på skjermen med hybelnavn (gangnummer+bokstav) på de ledige hyblene.

- **Registrer ny leietager**
  Her spør systemet om navn til studenten og hybelnavn samt at det registreres at studenten har betalt kr. 8000 i forskudd på husleie. Hvis det skrives inn navn på en opptatt hybel skal det gis feilmelding. Fortjenesten oppdateres videre med 3000 kr fordi studenten betaler fulle 4000 kr for den måneden hun flytter inn + oppsummerte strømregninger fra den/de måneder hybelen sto tom.

- **Registrer frivillig flytting av leietager**
  Her spørres det om studentens navn og hybelnavnet (eks. 3F, 1B,…) og det regnes ut hva studentene har tilgode på husleie fratrukket et ekspedisjonsgebyr på kr. 800.- som legges til fortjenesten. Beløpet studenten har til gode skrives til skjerm. Deretter registreres hybelen som ledig og utestående på hybelen settes lik 0.

- **Månedskjøring av husleie med strøm**
  I begynnelsen av måneden kjøres dette menyvalget (straks Gulbrand har fått strømregningen). Her belastes først alle leietagernes objekter med 4000 kr hver. Så leses en fil (”Lysregning.data”) fra Lysmegopp med forrige månedens strømforbruk og denne legges til husleien for de enkelte hyblene som har leietagere, med 2 kr per kilowatt-time. Her skal det til sist skrives ut på skjermen hvor mye som Gulbrand hadde i fortjeneste denne måneden.(= sum leieinntekter – 1000 kr for hver av hyblene, også de som ikke er leid ut. Regningen fra Lysoppmeg A/S må selvsagt også trekkes fra). Her skal programmet ikke akseptere at dette menyvalget kjøres mer enn en gang per måned.

- **Registrer betaling fra leietager**
  Denne metoden spør om hybelens navn (gangnummer og bokstav) samt hvilket beløp som betales. Metoden registrerer selvsagt i leietageren objekt hvor mye som er betalt.

- **Sjekk om noen leietagere skal kastes ut**
  På slutten av måneden kjører Gulbrand denne rapporten. Den sjekker om noen av leietagerne har kommet i minus med husleien (de har da brukt opp forskuddet og ikke betalt nok i husleie). Disse kastes uten bønn. For hver av disse kalles en metode **`tilkallTorpedo (String hybelNavn, String student, int krav)`**, som skriver på en fil (”Torpedo.txt”) alle studentene som skal kastes ut denne måneden. Denne filen sender Gulbrand til en av sine gode venner som aldri har problemer med slike oppdrag. Kravet til studenter som kastes ut er skyldig husleie + et torpedogebyr på kr. 1000. til Gulbrand. Eventuell betaling av torpedoene er en sak mellom studentene og torpedoene og behandles følgelig ikke i programmet. Slike hybler kan derfor med en gang markeres som ledige i systemet. Husk at det er bare en fil du skal lage for alle studentene som skal kastes ut den måneden, mens selve metoden skal kalles en gang for hver av disse studentene. Filen ”Torpedo.txt” skal ha følgende format:

    ```
    hybelNavn: StudentNavn: krav;
    ```

- **Skriv økonomirapport**
  Her står du fritt til å skrive ut tall på skjermen som kan interessere Gulbrand, men som et minimum skal du skrive ut summen av all fortjeneste Gulbrand har hatt siden systemet ble tatt i

bruk, hvor mange måneder systemet har vært i virksomhet og hvor mange månedsleier han i gjennomsnitt hver måned har gått glipp av fordi hybler har stått tomme.

Du kan i dette systemet anta at alle studenter har entydige navn. Du skal også gå ut fra at hvis en student flytter inn en måned, så må hun betale Gulbrand husleie for hele måneden (det blir enklest slik for Gulbrand).

Når programmet avsluttes, skal det skrive til fil status for alle hybler og studenter, samt regnskapet (som består av tre tall: sum strømregninger, sum innkrevet husleie vis ordinær husleie innbetaling og sum fortjeneste). Dette skrives som beskrevet ovenfor til en egen fil "HaiHus.data".

Hint:

1. I denne oppgaven er det ikke gitt hvor mange klasser du skal lage, men du skal i alle fall ha fire klasser: **`Oblig3`**, **`HybelHus`**, **`Hybel`** og **`Student`**. Det kan argumenteres for at **`Student`** er overflødig siden vi bare vet to ting om hver student: navnet og hvor mye de har innbetalt til Gulbrand. Det er vel likevel fornuftig med en egen klasse for Studenter fordi hvis Gulbrand ønsker å utvide systemet, kan det tenkes han vil vite mer om hver student som innflyttingsmåned, mobiltelefonnummer, hjemmeadresse, osv. Vi kan da bruke følgende programskjelett som du kan utvide med mange flere metoder og evt. andre klasser (antall linjer i "mønsterbesvarelsen" er ca. 450):

```
class Student {
         int  saldo;
         String navn;
         // Metoder som behandler studentene og deres variable.
         …….
   } // end Student

   class Hybel {
         Student leietager;
         int utestående; // brukes for strøm hvis leiertager == null
         // Metoder som behandler hyblene og deres variable.
         ……
   } // end Hybel

   class HybelHus {
        <økonomi-data her>
         Hybel [][] hyblene = new Hybel[4][8];

         HybelHus(String filnavn) {
            <konstruktør for klassen HybelHus,
             les "HaiHus.data" og oppprett
             Hybel og Student-objektene>
          }
         ……..
         void kommandoLøkke() {
               ……
         }
   } // end HybelHus

   class Oblig3 {
     public static void main (String[] args) {
          HybelHus utsyn = new HybelHus("HaiHus.data");
          utsyn.kommandoLøkke();
      } // end main
   } // end class Oblig3
```

2. I eksemplet på strukturen i programmet ovenfor er klassen **HybelHus** utstyrt med en metode (uten **void, static** eller noe annet foran) som heter det samme som klassen. Det er en konstruktør (les avsnitt 8.7 og 8.11 i læreboka). En konstruktør er en metode som kalles automatisk når man sier **new,** og er en måte å overbringe parametere til objektene vi lager. (Konstruktører er meget nyttige, men ikke nødvendige. Som alternativ kunne vi først ha laget objektet ned **new** og så brukt '.'-operatoren til å sette verdier inn i objektet)

3. I skjelettet av løsningen ovenfor er det en **Student**-peker **leietager** i **Hybel**-klassen. For å teste om den peker på et objekt, kan man teste:

```
if (leietager == null)
```

null er et Java-ord som er pekerverdien 'intet objekt'. Dvs. **if**-testen slår til hvis **leietager** ikke peker på et objekt. Verdien **null** kan også brukes i tilordningssetninger for eksempel hvis **leietager** peker på et objekt, kan vi få fjernet det med:

```
leietager = null;
```

4. Vi har behov for å plassere alle **Hybel**-objektene i en array. I program-eksempelet er det nyttet en todimensjonal array, hvor første indeksen går på gangnummer og den andre på hybel innen en gang. Nå er problemet at hybelen har et bokstavnavn (A,B,…,H) og ikke et tall. Dette kan løses som følger:

```
System.out.print("Gi gang:");
int gang = tast.inInt("\n");
System.out.print("Gi hybelbokstav:");
tast.skipWhite();
bokstav = tast.inChar();
i = (int) (bokstav - 'A');
```

Da vil **hyblene[gang-1][i]** finne riktig peker i arrayen. Motsvarende, hvis vi vet nummeret "i" til en hybel i gangen , og vil ha bokstaven , gjør vi det slik:

```
char b = (char) ('A' + i);
```

Begge disse omgjøringene fra **char** til heltall og motsatt, hviler på at bokstavene A,B,… ligger etter hverandre i kodingen av tegnsettet.

5. Når du skal skrive på en fil flere ganger, som ved hver kall på **tilkallTorpedo**, og ønsker å legge noe til filen (ikke overskrive det som står der allerede), må du åpne fila med "**append**". Dvs. at når du sier **new**, må du ha med en parameter nr. 2 som er **true**. Det åpnes da en ny fil hvis filnavnet er nytt, men hvis ikke, åpnes den gamle fila på slutten av det som hittil er skrevet – for eksempel:

```
void tilkallTorpedo (String hybelNavn, String student, int krav){
   torpedo = new Out("Torpedo.txt",true);
   < .. skriv neste utkastelse av en student her …>
   torpedo.close();
}// end tilkallTorpedo
```

6. Når man bruker en peker, må man jo først teste om den peker på et objekt før man evt. kan få adgang til noe i objektet. Her er det nyttig å bruke **&&** testen (betinget-og) som er slik at den bare tester det som står til høyre for **&&** hvis det som står til venstre er sant – eks.:

```
Student s = hyblene[i][j].leietager ;
if (s != null && s.navn.equals("Ola")) {
   // her kommer vi bare hvis s peker på et studentobjekt
   // og navnet i det studentobjektet er lik "Ola"
   ..........
}
```

Vi tester altså først om pekeren er forskjellig fra **null**, og hvis det er sant, så bruker vi en objekt-variabel i det objektet **s** peker på. Eksempelet forutsetter at alle elementene i **hyblene** -arrayen peker på et **Hybel**-objekt og at alle **navn** i **Student**-objekter peker på et **String**-objekt (hvis det er usikkert, må vi tilsvarende teste for det).

7. Registrering av hvor mange ledige hybler det er, kan registreres i metoden **`månedskjøringHusleie`** siden den kjøres bare en gang per måned. Samtidig beregner du her også månedens fortjeneste som legges sammen med Gulbrands totale fortjeneste.

8. For å lese inn lysregningen kan du benytte følgende kode for å dele opp linjene:

```
husnr = lysregning.inInt(":");
gangnr = lysregning.inInt(":");
bokstav = lysregning.inChar(":");
kwt = lysregning.inInt(": \n");
lysregning.skipWhite();
```

9. I denne oppgaven skal du ved oppstart lese fila: ”HaiHus.data” , som inneholder alle vesentlige opplysninger om leietagere, utleide hybler og samlet fortjeneste. Problemet er at første gangen systemet kjøres, så finnes ikke denne fila. Det er to måter å håndtere dette på. Du kan enten lage filen ”HaiHus.data” (slik den ser ut før noe er leid ut) med editoren din (emacs, TextPad, ...) som er denne fila er når intet er utleid; eller bedre: Du tester i koden om fila finnes. Importerer java.io-pakke (**`import java.io.*;`**) og der du leser inn fila, tester du slik:

```
if (new File("HaiHus.data").exists()) {
      // Fila finnes, og vi kan lese den
      In systemdata = new In("HaiHus.data");
      < les data>
      systemdata.close();
 } else {
      // Fila finnes ikke, og du skal sannsynligvis ikke gjøre
      // noe, eventuelt bare gjøre visse initialiseringer.
 }
```

10. For å sikre at den månedlige utsendingen av husleie med strøm bare skjer en gang per måned, må du nok både spørre Gulbrand om hvilken måned han skal ha husleiekrav for. Dessuten må du på fila "HaiHus.data" ha opplysninger om den siste måneden det allerede er skrevet ut husleie for.

Leveringskrav

Oppgaven skal utføres og leveres individuelt via Joly-systemet. Det ferdige programmet med filene”Oblig3.java”, :og filene ”HaiHus.data” og ett eksempel på fila ”Torpedo.txt” hvor minst en student skal kastes ut sendes inn til Joly innen leveringsfristen. I tillegg skal det legges ved ett eksempel på kjøring av systemet (ta klipp ut/lim inn fra kommandovinduet og legg det i TextPad eller Emacs og lagr det som ”Kjøring.txt”) Disse tre filene kan du for å få levert dem som en fil i Joly, enten legge dem i en zip-fil eller sette dem sammen (etter hverandre) med en editor til en fi:l”TilleggsLeveringOblig3.txt” (hvor du legger inn skillelinjer hvor en fil slutter og neste begynner).

NB. Husk at Joly-systemet bare virker på Blindern-maskinene (og som oftest hjemmefra hvis du har VPN) Sender du inn innleveringen din fra en e-postkonto fra yahoo eller hotmail, er det en stor sjanse at spamfilteret på Ifi tar besvarelsen og den vil aldri nå hjelpelærerer. Send derfor helst besvarelsen din fra den e-postkontoen du har fått her på Universitetet. Du er ansvarlig for at besvarelsen din blir levert!

(Virker ikke Joly, leverer du ved å sende inn som vedlegg til e-post til gruppelæreren din).

Lykke til!