UiO : Universitetet i Oslo

Oblig 3 (INF1000 - Høst 2010)

Gulbrand Grås Husleiesystem

> **Mål:** *Formålet med denne oppgaven er å gi trening i å løse et større programmeringsproblem ved hjelp av klasser og objekter, kombinert med de andre elementene vi har studert til nå som filbehandling, arrayer, metoder, og brukerinteraksjon via terminal.*

Leveringsfrist

Fredag 22. oktober kl. 16.00. Leveres i dette Innleveringssystemet. Husk å velge innleveringsnummer 3.

Leveringskrav

Før du leverer besvarelsen din kontrollér at den oppfyller følgende krav:

1. Besvarelsen må **benytte objekter av de 3 klassene Utsyn, Hybel, og Student** for å kunne bli godkjent. Dette betyr at informasjon om en hybel skal lagres i objekter av klassen *Hybel*, og informasjon om studenter lagres i *Student*-objekter.
2. Før du leverer skal du skrive en **kommentar øverst i programmet** der du sier noe om besvarelsen din generelt, f.eks. at du har løst alle deloppgaver og ekstraoppgaver, eller evt. hvilken deloppgave du ikke ble helt fornøyd med eller mangler litt arbeid.
3. Hele programmet skal legges i kun én .java-fil, som bør hete **Oblig3.java**, og skal inneholde alle klassene i besvarelsen din. Når du skal levere bør du først **komprimere den til en .tgz- eller en .zip-fil** (Se hint h). Ikke bruk andre komprimerte filformater. Hvis du ikke får til å lage .tgz- eller .zip-fil kan du levere den ukomprimerte .java-filen.
4. Java-nøkkelordet "`package`" skal ikke brukes. Dette er for at retter skal kunne kjøre programmet uten å bygge opp en spesiell mappe-struktur for hver innlevering.

Programmet skal bruke en datafil kalt **hybeldata.txt** som inneholder informasjon om husleiesystemet. Denne filen skal du som sagt ikke levere, men du kan anta at den som retter obligen din har følgende eksempel-datafilen klar i samme mappe som programmet ditt når hun kjører det: hybeldata.txt.

Du kan *diskutere* med andre studenter hvordan dere skal løse oppgaven, men det er ikke lov å *kopiere* noe Java-kode fra dem, selv om du endrer på koden etterpå; og det er heller ikke lov å hente programbiter fra andre besvarelser, for eksempel fra Internet. Hver student skal skrive sitt eget program. Dette er nærmere forklart i følgende krav til innleverte oppgaver ved Ifi, som du plikter å ha lest og forstått før du leverer din besvarelse:

```
http://www.ifi.uio.no/studier/studentinfo.html#krav.
```

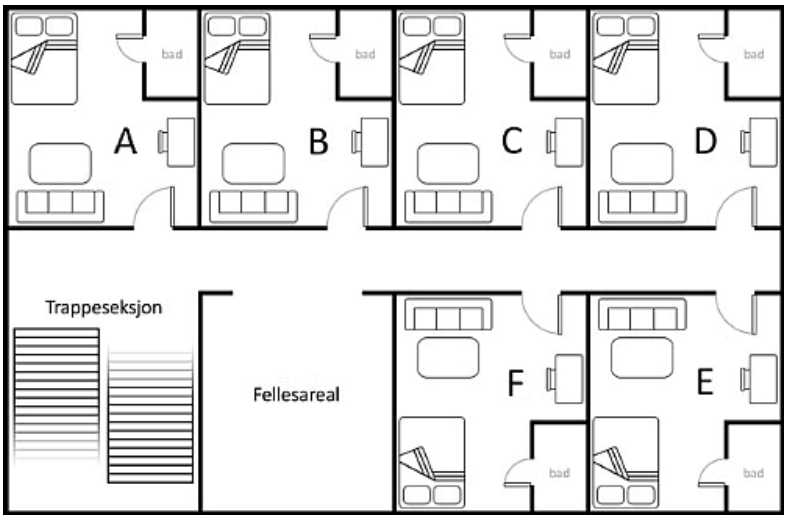
Alle innleveringer vil bli kontrollert med **Joly**. Hvis du samarbeidet mye med andre eller hentet viktig hjelp fra andre kilder enn lærebøker, websidene til kurset, eller gruppelærere og orakler, bør du nevne de andre kildene i besvarelsen din og hvilke deler av programmet det gjelder, som forklart i lenken ovenfor.

Godkjentsystemet oppdateres manuelt av gruppelæreren din litt på etterskudd. Det vil derfor stå "ikke levert" inntil hun krysser av obligen din som godkjent/ikke-godkjent. Bekreftelsen på at leveringen gikk i orden får du i stedet via mail. Hvis du fikk en slik mail-bekreftelse og valgte riktig innleveringsnummer (nr. 3 for denne obligen) så er leveringen i orden.

Oppgave

Gulbrand Grå har et lite hybelhus kalt Utsyn, med 18 studenthybler som han leier ut til studenter i Utopias hovedstad Uqbar. Du skal lage et system for å administrere utleie av hyblene i hybelhuset. Utsyn har 3 etasjer, nummerert fra 1 til 3. I hver etasje er det 6 hybler, kalt rom A til F, og et fellesrom. Hver hybel har et entydig «hybelnavn» som består av etasjenummer og rom-bokstav, f.eks. heter hybelen i rom C i andre etasje "2C".

Nedenfor ser du en illustrasjon av en etasje. Alle etasjene har samme planløsning.



Økonomi

Gulbrand Grå leier ut hyblene i de to første etasjene for 5000 kroner i måneden, mens husleien for en hybel i toppetasjen er 6000 kroner på grunn av utsikten.

Gulbrand har begrenset hans egne utgifter som eier av hyblene til 1200 kroner per hybel (A-F) per måned, og 1700 kr for fellesarealene i hver etasje, ved å hyre inn et firma som tar seg av alt vedlikehold og andre løpende utgifter, inkludert reparasjoner, kommunale avgifter, Internet- og kabel-tv-abonnement i hyblene, og utstyr og strøm til fellesarealer. Alt dette tar firmaet Vedlikehold A/S seg av, mot at Gulbrand betaler hver måned: 1200 kr per hybel (for hyblene A-F i hver etasje, dette betaler han uansett om en hybel har beboer eller ikke), pluss 1700 kr per etasje for fellesarealer. Beboerne betaler bare husleiene sine for hyblene A–F til Gulbrand, og han tar fra disse inntektene og betaler Vedlikehold A/S for jobben de gjør. Strømregninger for hyblene A-F inngår ikke i systemet siden disse betales direkte og adskilt av beboerne til strømselskapene de velger. Gulbrand har installert egen strømmåler i hver hybel for å slippe å tenke på dette.

Datafilen "hybeldata.txt"

Når programmet starter (og før hovedmenyen skrives ut på skjermen), skal programmet lese datafilen hybeldata.txt. I denne filen er det lagret blant annet informasjon om hvor lenge systemet har vært i drift og navn på de nåværende leietagerne. Første linje i filen inneholder fire heltalls-verdier adskilt med semikolon:

```
int måned; int år; int totalfortjeneste; int totaltAntallMåneder
```

Her er "int måned" og "int år" nåværende måneds- og årstall (eller mer presist, månedsnummeret og årstallet da månedskjøring sist ble utført), hvor måned er et tall i området 1 til 12, og årstallet er f.eks. 2010. Det tredje tallet er Gulbrands totale fortjeneste siden systemet ble satt i drift; og det siste tallet angir antall måneder systemet har vært i drift.

Deretter er det 18 linjer, en for hver hybel, med følgende format:

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

Eksempel på fire mulige linjer som kan være i begynnelsen av datafilen:

```
9;2010;1400500;24
1;A;6000;Ole Johan
1;B;12000;Erik Smith
1;C;0;TOM HYBEL
...osv...
```

For tomme hybler skal studentnavnet lagres i datafilen som `TOM HYBEL`, med saldo `0`. Du kan anta at alle studenter har unike navn. (Se hint b under for tips om lesing av datafilen.) Internt i programmet kan du velge om du representerer tomme hybler med verdien `null`, eller med det spesielle studentnavnet `TOM HYBEL`, eller på en annen måte.

Alle disse dataene skal holdes oppdatert internt i programmet mens det kjører, og skal skrives tilbake til datafilen når brukeren utfører ordre 'o' i hovedmenyen ("Avslutt"). Slik kan Gulbrand starte og avslutte programmet uten å miste data.

Menyvalgene

Programmet skal være menystyrt. Det skal skrive ut på skjermen en meny over mulige ordre og be brukeren om å taste en av disse. Hvis brukeren taster inn en ulovlig ordre, skal det gis feilmelding. Programmet skal gå i løkke og fortsette å lese og utføre ordre helt til brukeren taster ordre 'o' for å avslutte. Bruker skal kunne gi 7 ordre:

1. Skriv oversikt
2. Registrer ny leietager
3. Registrer betaling fra leietager
4. Registrer frivillig utflytting
5. Månedskjøring av husleie
6. Kast ut leietagere
0. Avslutt

Detaljert beskrivelse av ordrene:

1. **Skriv oversikt**
   Når bruker gir denne ordren skal programmet skrive ut en oversikt over alle hyblene, som viser for hver hybel: hybelnavn, leietager-navn, og saldo. Dersom hybelen er ledig, skal teksten LEDIG skrives ut i stedet for leietager-navn i oversikten, og saldoen skal vises som 0. Til slutt skal nåværende måned, år, antall måneder systemet har vært i drift, og totalfortjeneste skrives ut på skjermen. Eksempel på hvordan oversikten kan se ut:

   ```
   Hybel Leietager               Saldo
   ----- --------------------- -------
   1A    Ole Johan                6000
   1B    Erik Smith              12000
   1C    ( LEDIG )                   0
              ...osv...
   Måned/år, og driftstid:  9/2010,  24 mnd. i drift
   Totalfortjeneste:        1400500 kr
   ```

2. **Registrer ny leietager**
   Denne ordren brukes når en student ønsker å flytte inn og leie en av hyblene. Først skal systemet sjekke om det finnes ledige hybler. Hvis det

ikke er noen ledige hybler skal programmet bare skrive ut en melding om det og returnere til hovedmenyen.

Hvis det finnes ledige hybler skal hybelnavnene til disse (f.eks 1C, 2B) skrives ut på skjermen, og så skal programmet spørre hvilken av disse studenten ønsker å leie (se hint e under for hvordan slike hybelnavn kan leses fra tastatur). Bruker skal kunne taste inn ønsket **hybelnavn** som 1C, 2B, osv. Er valgt hybel ledig, skal programmet spørre om studentens navn, og registrere innflyttingen.

Studenten som flytter inn betaler samtidig et depositum på 10 000 kroner. Fra dette trekkes det med én gang månedsleien for den første måneden (husk at månedsleien er avhengig av etasje). Det som er til øvers blir studentens **saldo** (her kan du lese en definisjon av saldo, også på engelsk), men det kan forklares som beholdningen i en slags forenklet bankkonto som leietagerne har hos Gulbrand: Når de betaler inn for husleie legges beløpet til i saldoen, og ved "månedskjøring" blir husleien trukket fra saldo. Saldoen vil være negativ hvis studenten skylder Gulbrand penger. Han liker å gjøre ting enkelt for seg selv, så han trekker alltid én hel månedshusleie ved innflytting, som nevnt ovenfor, uansett hvilken dag i måneden studenten flyttet inn.

Programmet skal til slutt skrive ut en beskjed på skjermen om at innflyttingen ble gjennomført. Beskjeden skal inneholde hybelnavnet (etasje+bokstav), studentens navn, og gjenværende saldo.

3. **Registrer betaling fra leietager**
   Programmet skal spørre om et hybelnavn (f.eks. 2B), og beløpet som beboeren i den oppgitte hybelen skal betale. Hvis hybelen som ble oppgitt ikke hadde beboer skal det skrives en feilmelding. Hvis det derimot var en beboer der skal beløpet adderes til studentens saldo og en passende melding skrives på skjermen. Saldoen fungerer som en slags konto som studenten har hos Gulbrand, og hver student må passe på å ha nok penger i denne saldoen til å dekke husleiene hver måned.

4. **Registrer frivillig utflytting**
   Programmet spør om *navnet på studenten* som ønsker å flytte ut. Deretter går programmet gjennom alle hyblene og ser om det finnes en student med det navnet. Hvis studenten ikke ble funnet, skal det gis en feilmelding.

   Blir studenten funnet, skal du registrere i systemet at hybelen ikke lenger har beboer (du kan f.eks. angi dette vha. `null`-peker eller bruke det spesielle navnet `"TOM HYBEL"`, men det du velger må selvsagt fungere sammen med resten av programmet ditt).

   Studenten blir trukket et gebyr på 650 kroner for utflyttingen, som du legger til Gulbrands totale fortjeneste, og resten av saldoen får studenten tilbake (hvis den var på mer enn 650 kr). Skriv ut på skjerm evt. tilgodebeløp etter å ha trukket utflyttingsgebyret fra saldoen. Hvis studenten hadde mindre enn 650 kr. igjen i saldoen antar vi for enkelhets skyld at hun betaler restbeløpet nå samtidig med utflyttingen. Skriv i så fall ut det totale restbeløpet hun betalte; dette beløpet legges til i Gulbrands totalfortjeneste, inkludert utflyttingsgebyret.

   [***Ekstraoppgave 1*: Alfabetisk sortering:** Hvis du ønsker det kan du utvide denne ordren til å vise en liste eller meny med student-navnene sortert alfabetisk, slik at det blir lettere for Gulbrand å velge den som skal flytte ut.]

5. **Månedskjøring av husleie**
   Meningen er at Gulbrand skal utføre denne ordren første dag i hver måned. Når ordren velges skal programmet først spørre om brukeren virkelig ønsker å utføre månedskjøringen, for måneden som kommer etter måneds- og årstallet da månedskjøring sist ble utført (f.eks. hvis siste månedskjøring ble utført for måned 9 i år 2010, kan spørsmålet være "Ønsker du å utføre månedskjøring for måned 10/2010 (j/n)?"). Svarer brukeren `n`, returnerer programmet til hovedmenyen.

   Svarer brukeren `j`, skal månedsnummeret (og ved årsskiftet årstallet også) oppdateres, og det utføres det månedlige regnskapet. Programmet går gjennom alle hyblene: For hver hybel som har beboer trekkes månedsleien for hybelen fra studentens saldo og legges til Gulbrands månedsfortjeneste hvis det var nok dekning i saldoen. Gulbrand belaster altså husleien forskuddsvis, for måneden som nettopp har begynt. Hvis noen leietagere ikke hadde nok i saldoen så trekkes likevel hele husleien fra saldoen, som går da i minus, men Gulbrands fortjeneste økes bare med det som var på saldo (han har uansett en god løsning på slike tilfeller, og får inn resten av fortjenesten som forklart i menyvalg 6). Videre trekkes (Gulbrands) utgifter fra (Gulbrands) månedsfortjeneste. Gulbrands utgifter er kort og godt det han betaler til Vedlikehold A/S for vedlikehold og lignende. Husk at det er forskjellige utgifter for hybler og fellesarealer, som forklart i avsnittet Økonomi ovenfor.

   Til slutt skal følgende skrives til skjerm:

   a. **Måned/år** som månedskjøringen gjelder for; **og driftstid** i antall måneder systemet har vært i drift, inkludert den nye måneden.
   b. **Månedens fortjeneste:** er Gulbrands inntekter minus utgifter i denne månedskjøringen. Hvis du ønsker å vise andre inntekter fra utflyttinger, innflyttinger, eller lignende siste måned, så skriver du dette ut som en egen post, den vanlige "månedens fortjeneste" skal kun vise regnskapet for månedskjøringen forklart ovenfor.
   c. **Totalfortjeneste:** er Gulbrands nye totalfortjeneste, oppdatert med med denne månedens fortjeneste.
   d. **Gjennomsnittlig månedsfortjeneste** regnes ut som (totalfortjeneste / totaltAntallMåneder).

6. **Kast ut leietagere**
   Hybelhuset til Gulbrand er veldig populært, med mange studenter som ønsker å leie. For å åpne plass for nye leietagere (og sikre inntektene sine) så har han innført en regel om at leietagere som har kommet så langt i minus på saldoen at de skylder mer enn én husleie, blir kastet ut ved hjelp av torpedo når denne ordren utføres. Kravet til hver enkelt student som kastes ut er det de skylder i husleie pluss et utkastingsgebyr på 3000 kroner. Torpedoen og Gulbrand deler dette gebyret likt. Gulbrands halvdel av gebyret pluss det studenten var i minus på saldoen legges til totalfortjenesten med én gang når dette menyvalget kjøres (siden hjelperen hans alltid greier å ordne disse sakene raskt), og hybelen registreres som ledig.

   Programmet går gjennom alle hyblene og finner studentene med saldo enda lavere enn én månedsleie i minus (husk de forskjellige leieprisene!). For hver av disse studentene skal du kalle følgende hjelpemetode: (som du også skal programmere)

```
void tilkallTorpedo(int etasje, int rom, int krav) { // ...
```

   Denne metoden skriver hybelnavn (etasje+rom), studentens navn, og pengekravet, både til skjerm og til en fil kalt `torpedo.txt`. Når metoden

skriver til filen, skal den ikke overskrive det som ligger der, men bare legge til en ny linje på slutten (se hint f nedenfor).

o. **Avslutt**
Ved utførelse av denne ordren skal nødvendige data skrives til `hybeldata.txt`: måned, år, totalfortjeneste, antall måneder i drift, samt leietager og saldo for alle hyblene. Deretter skal programmet avslutte.

[***Ekstraoppgaver 2 - 5*: Hjelpemetoder**]
Her er fire eksempler på mulige hjelpemetoder som kan være nyttige i programmet ditt. Disse har både inn- og ut-parametre. Programmér disse eller andre hjelpemetoder du får bruk for og benytt dem i løsningen din på de andre deloppgavene.

```
Hybel spørOmHybel(String ledetekst) {
    // x.2) Skriver ut "ledetekst" på skjermen, leser et hybelnavn fra
    //      tastatur, og returnerer den tilhørende Hybel-pekeren.
    return null;
}
String finnHybelnavn(int etg, char rom) {
    // x.3) Konverterer etg. og rom til et hybelnavn.
    return null;
}
Student finnBeboer(String hybelnavn) {
    // x.4) Finner og returnerer peker til leietageren som leier hybelen
    //      angitt i inn-parameteren "hybelnavn".
    return null;
}

String stortTallTilString(int tall) {
    // x.5) Lager en tekst med en mer lettlest form av inn-parameteren "tall"
    //      med mellomrom hvert 3. siffer, f.eks. 1400500 skal gi "1 400 500".
    return null;
}
```

Hint

Disse hint er ment for de som ønsker litt ekstra-hjelp. Du trenger ikke lese dette avsnittet for å løse obligen. Det vil også bli lagt ut flere hint i kurs-bloggen, blant annet om hvordan du kan løse obligen uten bruk av EasyIO. Følg ellers med på "Siste beskjeder" på kurshjemmesiden, der vil det blant annet komme informasjon om orakelhjelp til denne obligen.

a. **Programstruktur:** Nedenfor ser du et programskjelett du kan bruke som utgangspunkt hvis du ønsker det. Du kan utvide det med flere metoder, klasser, osv. Hvis du synes det er vanskelig *å komme i gang* med denne obligen anbefales det å lese litt teori om *klasser og objekter*, f.eks. kapittel 8 i læreboka eller Marit Nybakkens notat "objekter.pdf"; eller løse ukeoppgaver 6 og 7.
Det står i leveringskrav ovenfor at du må benytte objekter av minst **3 klasser**: Utsyn, Hybel og Student. Det er nok med ett objekt av klassen *Utsyn* (det representerer hele hybelhuset). Programmet bør benytte 18 objekter av klassen *Hybel*. Det kan argumenteres for at klassen *Student* er overflødig siden vi bare vet to ting om studentene: saldo og navn. Det er likevel fornuftig å ha med denne klassen for å samle disse to verdiene i ett type objekter, og også i tilfelle Gulbrand senere ønsker å utvide systemet med mer informasjon om hver student, som f.eks. mobilnummer, innflyttingsmåned o.l.

```
// Skriv en kommentar om besvarelsen din her: ...
// ...
import easyIO.*;

class Oblig3 {
    public static void main(String [] args) {
        Utsyn s = new Utsyn();
        s.ordrelokke();
    }
}

class Student {
    String navn; // studentens navn
    int saldo;   // studentens saldo

    // Evt. metoder for å behandle en Student...
}

class Hybel {
    Student leietager;  // peker på et Student-objekt
    int husleie; // 6000 hvis hybelen er i 3. etasje, ellers 5000.

    // Evt. metoder for å behandle en Hybel...
}
```

```
class Utsyn {
    In tast = new In();
    Out skjerm = new Out();

    Hybel[][] hybler = new Hybel[3][6];
    // Variabler for økonomidata kan legges inn her...

    // Konstanter:
    final String FILNAVN = "hybeldata.txt";
    final String TOM_HYBEL = "TOM HYBEL";
    // Konstruktør for klassen Utsyn
    Utsyn() {
        // Her kan du lese datafilen "hybeldata.txt" og lagre hele innholdet
        // i datastrukturene dine.  Husk å opprette Hybel- og Student-objekter.
        // Eksempel på innlesing av en saldo, for en gitt "etg" og "rom":
        //    hybler[etg][rom] = new Hybel();
        //    hybler[etg][rom].leietager = new Student();
        //    hybler[etg][rom].leietager.saldo = innfil.inInt(" ;");
    }

    void ordrelokke() {
        int ordre = -1;
        while (ordre != 0) {
            // Skriv ut meny:
            skjerm.outln("Meny:");
            skjerm.outln("1. ...");

            // Les ordre fra bruker
            skjerm.out("Ordre: ");
            ordre = tast.inInt();

            switch(ordre) {
                case 1: skrivOversikt(); break;
                case 2: registrerNyLeietager(); break;
                case 3: registrerBetaling(); break;
                case 4: // ...
                // ... Fyll ut resten...
                default: // Gi feilmelding.
            }
        }
    }

    // Metoder for de forskjellige ordrene i ordrelokke()

    void skrivOversikt() { /* ... */ }
    void registrerNyLeietager() { /* ... */ }
    void registrerBetaling() { /* ... */ }

    // ... Fyll ut med minst 5 metoder til
}
```

b. **Konstruktør:** I klassen Utsyn er det en metode som ikke har det reserverte ordet `void` eller noe annet foran metodenavnet, og heter det samme som klassen: «`Utsyn()`». Det er en spesiell type metode kalt en *konstruktør*, som blir utført automatisk når man oppretter et objekt av klassen (med `new Utsyn()`). Du kan lese mer om konstruktører i avsnitt 8.7 og 8.11 i læreboken, eller i forelesningsnotatene fra uke 7, eller i Marit Nybakkens notat «konstruktører.pdf».
Det anbefales å starte arbeidet med obligen ved å programmere konstruktøren `Utsyn()` slik at den leser datafilen. I begynnelsen bør du også skrive ut på skjermen det som du leser inn fra fil, slik at du kan kontrollere at datafilen blir lest riktig, f.eks. i stedet for bare å si «`int x = fil.inInt(" ;");`» kan du bruke:

```
int x = fil.inInt(" ;");  System.out.println("x=" + x);
```

c. **null-pekere:** Det ble nevnt i menyvalg 4 ovenfor at du kunne velge om du angir ledige hybler ved hjelp av peker-verdien *null* eller ved å bruke det spesielle Student-navnet `"TOM HYBEL"`. Det første alternativet er vanskeligere å programmere enn det andre, men gir en mer "objektorientert" løsning. Hvis du velger første alternativet så kan du teste om peker-variabelen *leietager* faktisk peker på et eksisterende Student-objekt ved et gitt tidspunkt slik:

```
if (leietager != null) {
```

«**`null`**» er et reservert ord i Java som angir peker-verdien «intet objekt». `if`-testen ovenfor sjekker derfor om *leietager* peker på noe som

helst objekt annet enn «intet objekt». Omvendt kan du også sjekke om en peker ikke peker på noe objekt, ved hjelp av `== null`. For å sette en peker til å peke på `null`, f.eks. når en student flytter ut, kan du skrive:

```
leietager = null;
```

d. **NullPointerException:** Når man bruker «prikk-notasjon» på en peker (f.eks. *`peker.saldo`*) må man passe på at den ikke inneholder `null`-verdien, ellers kræsjer programmet og Java-kjøresystemet gir feilmeldingen *NullPointerException*. F.eks, hvis du ønsker å få tak i *leietager*`.navn` så må du passe på at *leietager* peker på noe annet enn `null`. Dette kan ordnes ved å utføre første if-testen vist i hint c ovenfor, og bare bruke prikk-notasjon hvis if-testen ga *true*; eller, en enklere måte er å kombinere if-testen med en annen betingelse i en eksisterende if-test ved hjelp av && (*og*) operatoren. Pass på at betingelsen `!= null` står først i den sammensatte if-testen, f.eks. slik:

```
Student s = hybler[i][j].leietager; // s (og leietager) kan være null
if (s != null && s.navn.equals("Ola")) {
   // Hit kommer vi bare hvis s ikke er null og s.navn er Ola
}
```

Flere tips til NullPointerException og andre feilmeldinger kommer i *Ukeoppgaver 7*.

e. **Hybelnavn:** Ofte skal vi lese eller skrive ut *hybelnavn* (som 2B, 3E). De består av et tall og en bokstav (uten noe skilletegn imellom), men datastrukturen vi bruker for hybler er en array, som må indekseres med heltall; derfor må vi konvertere bokstaven **fra char til int**. Det kan gjøres ved hjelp av «`int rom = (int) (bokstav - 'A');`». I tillegg må vi finne en måte å trekke ut etasjen (2) fra hybelnavnet (2B), når det ikke er skilletegn mellom etasje-nummer og rom-bokstav. Det kan gjøres ved å lese begge deler som tegn (char) først, og konvertere begge til int, for eksempel slik: (Dette viser hvordan vi kan lese et helt *hybelnavn* fra tastatur og deretter trekke ut etasje og rom som heltall)

```
skjerm.out("Oppgi hybelnavn: ");
int etg = (int) (tast.inChar(" \n\r") - '1'; // '1' gir [0], '2' gir [1], osv.
char bokstav = tast.inChar(" \n\r"); // Les rombokstaven
int rom = (int) (bokstav - 'A'); // 'A' gir [0], 'B' gir [1], osv.
```

Det som står i parenteser etter `inChar(`*...her...*`)` angir tegn som innlesingen skal hoppe over, dvs. skal bare betraktes som skilletegn mellom de ønskede input-data hvis de blir lest inn. I koden ovenfor er det angitt at mellomrom (`" "`), linjeskift (`\n`), og vognretur (`\r`) skal betraktes som skilletegn av `inChar`. Etter disse setningene vil variabelen `rom` inneholde verdien 0 hvis brukeren tastet inn en A-hybel (f.eks. 3A), 1 hvis det var en B-hybel, osv.; og etasjenummer vil havne i variabelen `etg` (fratrukket 1).

Omvendt kan man konvertere **fra int til char**, slik:

```
char bokstav = (char) ('A' + intVariabel);
```

f. ***Append* modus** Hver gang metoden *tilkallTorpedo(..)* blir kalt skal du legge til en ny linje på slutten av filen torpedo.txt (uten å overskrive det som allerede var i filen). Da bør man åpne filen i *append*-modus, som sørger for nettopp det (utskrift som sendes til filen blir da bare lagt til på slutten av filen uten å endre linjene den hadde fra før). Hvis man bruker EasyIO kan dette gjøres ved å angi en ekstra-parameter «`,true`» når man åpner filen:

```
Out fil = new Out("torpedo.txt", true);
```

Husk å lukke filen når du er ferdig med å skrive til den! (`fil.close()`).

g. **Eksisterer filen?** Når programmet starter skal du lese inn datafilen `hybeldata.txt`. Hvis filen ikke finnes i samme mappe der du kjører programmet, kan programmet ditt kræsje. Dette kan man unngå ved å lage datafilen manuelt, eller laste ned *eksempel-datafilen*, eller du kan skrive programmet slik at det kan **teste om filen finnes**, og ta høyde for det hvis ikke. Følgende kode kan brukes til denne testen, og krever at man har setningen `import java.io.*;` øverst i programmet:

```
if (new File("hybeldata.txt").exists()) {
   // Les inn filen
} else {
   // Filen finnes ikke.  Gi feilmelding eller opprett tomme hybler.
}
```

h. **.tgz og .zip:** Du kan lage en .tgz-fil som bare inneholder filen Oblig3.java slik på Linux:

```
tar -cvzf Oblig3.tgz Oblig3.java
```

Og tilsvarende for å lage en .zip-fil med bare Oblig3.java på Linux:

```
zip -r Oblig3.zip Oblig3.java
```

Hvis du har spørsmål, kommmentarer, eller finner feil i oppgaveteksten kan du skrive disse i kurs-bloggen.

Lykke til!