Obligatorisk oppgave 3 (INF1000 – Høst 2008)
– alternativ 1: OO-system

Gulbrand Grås husleiesystem

Leveringsfrist

Oppgaven må leveres senest ***fredag 24. oktober kl 16.00***. Viktig: Les slutten av oppgaven for detaljerte leveringskrav. **N.B.** 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.

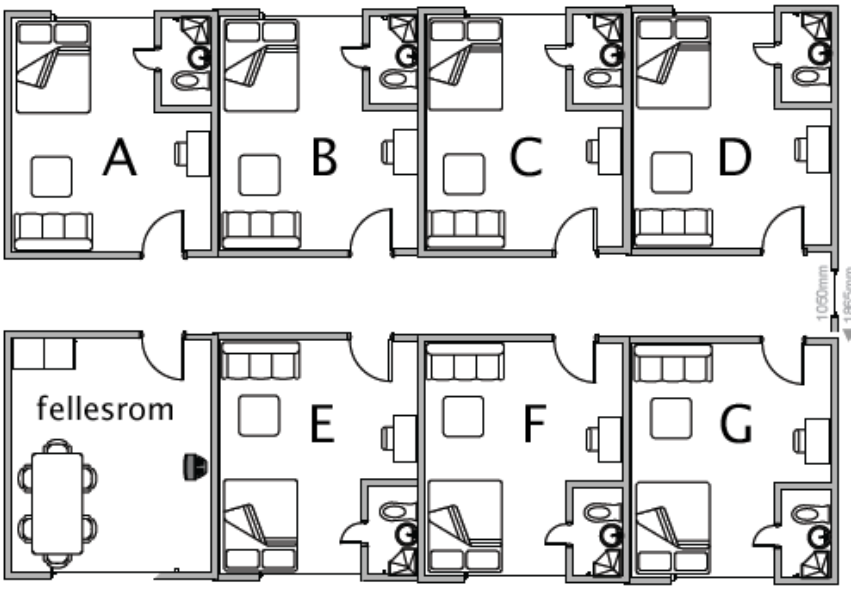
Formål

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.)

Oppgave

I likhet med de fleste andre universitetsbyer har Utopias hovedstad Amaurot dessverre stor mangel på studenthybler. Gulbrand Grå og hans firma HaiHus A/S lever av denne mangelen. Hybelhuset hans er så populær at du kan regne med at alle hyblene er opptatte nesten hele tiden og at han derfor også må holde orden på en venteliste av studenter som vil leie så fort en hybel blir ledig. Du skal hjelpe "Ærlige Gulbrand" å lage et system for å administrere utleie, utdeling av hyblene, og effektiv innkreving av husleie.

I denne oppgaven skal vi se på Gulbrands tre-etasjes studenthus Utsyn. Hver etasje (1-3) har 7 hybler og et fellesrom. Hver hybel har et unikt hybelnavn som består av etasjenr + en bokstav (A-G). Da er 3E hybel E i 3. etasje, mens 2C er hybel C i 2. etasje. Nedenfor vises de 7 smakfulle hyblene i 3. etasje og fellesrommet. De 2 andre etasjene har tilsvarende hybler, bortsett fra at hyblene i 1. etasje leies umøblert og har derfor lavere husleie:



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

Vi skal ha følgende menyvalg i systemet:

1. Avslutt
2. Skriv oversiktstabell
3. Registrer ny leietager
4. Registrer utflytting av leietager
5. Registrer innflytting fra venteliste
6. Registrer betaling fra leietager
7. Måneds-kjøring av husleie
8. Sjekk om noen leietagere skal kastes ut

Datafilen hybeldata.txt:
Programmet skal ved oppstart lese inn status for alle hyblene fra en fil: "hybeldata.txt", altså hvilke som er ledige og opptatte, samt navn på studenten og husleiestatus (hvor mye de er i pluss eller minus med husleia). På denne filen lagres også nåværende månedsnummer, år, antall måneder som systemet har vært i drift, og total fortjeneste som Gulbrand har tjent så langt; og et tilleggstall til annen evt. informasjon som du vil skal "overleve" mellom to kjøringer av programmet. Når brukeren velger kommandoen 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 mellom to kjøringer av programmet. Formatet på denne fila, "hybeldata.txt", kan du bestemme selv, men her er et forslag du godt kan benytte:

Først står det en linje for hver hybel, 21 linjer i alt, med hver linje på følgende form:

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

Et eksempel på en linje kan være:

```
2; C; Ole Brun; 2400;
```

For tomme hybler settes studentnavn lik "TOM HYBEL" og 0 i siste tall. Etter disse 21 linjene skal det stå en siste linje med format:

```
int måned; int år; int totaltAntallMåneder; int totalFortjeneste; 0;
```

Det siste tallet er ikke helt nødvendig men kan f.eks. brukes som nevn i deloppgave 4. Det nest siste tallet teller totalt antall måneder som systemet har vært i drift, og det foregående tallet skal akkumulere totalfortjeneste på all utleievirksomhet. Inn i dette beløpet regnes det differansen mellom hva Gulbrand får innbetalt i husleie fratrukket omkostninger for vedlikehold, strøm, torpedogebyr, osv. (se under). Siden Ærlige Gulbrand regner med å alltid få inn det noen skylder ham, på et eller annet vis, så oppdaterer vi totalfortjenesten hver gang vi oppdaterer alle skyldig beløp og ikke når vi registrerer enkeltbetalinger fra leietagerne.

Gulbrand har beregnet at avskrivninger, faste kostnader, og strøm kommer på 1000 kroner per hybel per måned. Han tar kr. 5500 per hybel per måned, for hyblene i de to øverste etasjene (som er møblert), og bare 4500 kr for hyblene i første etasje (som er umøblert).

Datafilen venteliste.txt:

Hybelhuset Utsyn er veldig populær blant studentene i Utopia, og derfor er det nesten alltid beboere i alle hyblene. Likevel så ønsker ikke Gulbrand å gå glipp av potensielle kunder, så har etablert følgende enkel ordning for å bedre tilgangen til nye leietagere når noen flytter ut. Når nye studenter ønsker å leie men det ikke er noen ledige hybler, så spør han om de vil stå på venteliste. Hvis de svarer ja plasserer han dem på slutten av ventelisten. Når så en hybel blir ledig ringer han den studenten i ventelisten som har ventet lengst og lar han flytte inn og starte sin leieperiode. Vi kan regne med at studentene som har valgt å melde seg på venteliste alltid vil flytte inn så fort de får sjansen, og uansett hybeltype.

Ventelisten lagres i en fil "**`venteliste.txt`**", som brukes bare i menyvalgene 2 og 4. Du trenger derfor ikke lese denne datafilen når programmet starter, men kan gjøre det hvis du ønsker det. Formatet på filen er veldig enkel, det står bare navnet på én student i hver linje, den som har ventet lengst står på første linje, og den som meldte seg sist står i siste linje i filen. Dette formatet gjør det lett å legge til nye studenter på ventelista, ved å åpne filen i *append*-modus, se hint nr. 5.

Menyvalgene:

Etter at programmet ditt har lest inn datafilen **`"hybeldata.txt",`** og laget de objektene som der beskrives, skal det gå i en kommandoløkke hvor bruker kan gi følgende kommandoer:

1. **Skriv oversiktstabell:**
   Denne metoden skal skrive ut en oversikt som viser, for hver hybel: hybelnavn, studentnavn (eller "ledig" hvis hybelen er ledig), og saldo (eller skyldig beløp hvis det er i minus). Utskriften kan formateres i en tabell som vist under, eller på en annen måte.

```
Hybel Leietager            Saldo
----- -------------------- -----
1A    ( ledig )               0
1B    Albert Johnsen       4300
1C    Johanne Ullersen     9000
1D    ( ledig )               0
         .... osv ....

Måned: 10.2008.  Måneder i drift: 48.  Totalfortjeneste: 274000.
```

   Til slutt skrives det ut nåværende måned, antall måneder systemet har vært i drift, og total fortjeneste for Gulbrand.

2. **Registrer ny leietager:**
   Denne kommandoen kjøres når en ny leietager ønsker å leie hybel. Systemet spør da om navnet til studenten. Hvis det finnes en ledig hybel så registrerer systemet at studenten har flyttet inn (programmet kan velge fritt hvilken ledig hybel som tas, det er så få ledige at studentene ikke bryr seg om hvilken det er, bare om hybel-type). Studenten betaler da 9500 kr, som inkluderer forskudd og lagres i **saldo** for studenten etter fratrekk av første månedsleie (5500,- for møblert, 4500,- for umøblert hybel). Gulbrands fortjeneste økes

med hele husleien siden han avskriver vedlikeholdsutgiftene o.l. for alle hyblene under månedskjøring. Systemet gir da en beskjed til bruker om at innflytting gikk i orden.

Hvis det derimot ikke fantes noen ledig hybel, så spør programmet **`(j/n)`** om studenten vil melde seg på ventelisten. I så fall legges studentnavnet til på slutten av ventelisten (se mer om dette ovenfor under beskrivelsen av datafilen **`"venteliste.txt"`**); og beskjed skrives på skjermen om at studenten er lagt i ventelisten.

3. **Registrer utflytting av leietager:**
   Her spørres det om navnet på studenten som vil flytte ut. Hvis navnet ikke finnes blant beboerne gis det en feilmelding, og hvis den fantes registreres det at studenten flytter ut: Det regnes da ut hva studenten har tilgode på husleie fratrukket et ekspedisjonsgebyr på kr. 700,- som legges til i fortjenesten. Beløpet studenten har til gode skrives til skjerm. Videre registreres hybelen som ledig.

4. **Registrer innflytting fra venteliste:**
   Denne kommandoen skal flytte inn i hybelhuset en leietager fra ventelisten hvis det finnes en ledig hybel og ventelisten ikke er tom. Husk at noen studenter på ventelisten er desperate og bryr seg hverken om typen hybel (møblert/umøblert), etasje, eller rom, så her skal man bare finne en eller annen ledig hybel og flytte inn studenten som står først på ventelisten. Det er nok at denne metoden bare matcher én ventende student med en ledig hybel. Skriv ut til slutt en beskjed på skjermen om innflytting gikk i orden.

   Hvis innflytting gikk i orden skal regnskapet oppdateres på samme måte som i deloppgave 1, og metoden skal helt til slutt "fjerne" første person fra ventelisten. Dette skal gjøres ved å kall på en hjelpemetode **`void fjernFraVenteliste(String navn)`**, med en parameter som angir navnet som skal tas bort fra ventelisten. Metoden kan programmeres på flere måter, f.eks.: (1) den kan endre selve datafilen **`"venteliste.txt"`** og fjerne første linje i den (dette kan gjøres ved å lese hele innholdet fra filen inn i en array med mer enn nok plasser, f.eks. 200, så lukke filen og åpne den igjen for utskrift). (2) En annen måte er å la datafilen **`"venteliste.txt"`** beholde alle navnene og i stedet ha en variabel i programmet som sier hvilket linjenummer neste ventende student er i. Velger du det siste bør du passe på at programmet husker det i neste kjøring, f.eks. ved å skrive linjenummeret ut som siste tall i datafilen **`"hybeldata.txt".`**

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

6. **Månedskjøring av husleie:**
   Første dag i hver måned kjører Gulbrand dette menyvalget (husk å oppdatere månedsnummeret). Han trekker husleiene forskuddsvis, for måneden som nettopp har begynt. Alle leietagernes objekter belastes derfor nå med husleien (5500,- for møblert; 4500,- for umøblert hybel) for inneværende måned.

   Til sist skrives ut på skjermen: (1) hvor mye Gulbrand hadde i fortjeneste denne måneden, beregnet som sum leieinntekter minus 1000 kr for hver av hyblene (kr. 1000 er Gulbrands

utgifter) , også de som ikke er leid ut og fellesarom. (2) Skriv også ut oppdatert totalfortjeneste for hele perioden siden systemet ble satt i drift; og (3) gjennomsnittsfortjenesten per måned for hele driftstiden (til dette bruker du telleren for totalt antall måneder i drift).

Metoden for månedskjøring skal også ha en mekanisme for å unngå at Gulbrand utfører månedskjøring flere ganger samme måned ved feiltakelse. Du kan f.eks. skrive ut hvilken måned månedskjøring ble sist utført, og spørre bruker om nåværende måned.

Når denne kommandoen er kjørt, skal nåværende måned (og evt. også år) økes til neste måned.

7. **Sjekk om noen leietagere skal kastes ut:**
   Gulbrand kjører denne kommandoen når han har behov for raske penger. Den sjekker om noen av leietagerne har kommet så langt i minus med betalingene at de skylder en hel husleie eller mer. Disse kastes ut uten bønn. For hver av disse kalles en hjelpemetode **`tilkallTorpedo(int etg, int rom, int krav)`** som skriver på skjermen en liste over studentene som skal kastes ut denne måneden og kravet de må betale. Denne listen sender Gulbrand til en av sine gode venner som aldri hatt problemer med slike oppdrag. Kravet til studenter som kastes ut er skyldig husleie + et utkastingsgebyr på kr. 2000 til fordeling mellom Gulbrand og torpedoen hans. Halvparten av dette registreres som fortjeneste. Slike hybler kan med en gang markeres som ledige i systemet. Husk at utskriften skal tilsammen bli en liste, mens selve hjelpemetoden kalles en gang for hver av studentene i trøbbel.

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

Når programmet avsluttes, skal det skrives til fil status for alle hybler og studenter, samt regnskapet til Gulbrand, som beskrevet ovenfor under "Datafilen **`"hybeldata.txt".`**

Hint

1. I denne oppgaven er det ikke gitt hvor mange klasser du skal lage, men du skal i alle fall ha 4 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 Ærlige Gulbrand. Det er vel likevel fornuftig med en egen klasse for studenter fordi hvis Ærlige Gulbrand ønsker å utvide systemet, kan det tenkes han vil vite mer om hver student som innflyttingsmåned, mobiltelefonnummer, osv. Du kan bruke følgende programskjelett, som du kan utvikle med mange flere metoder og evt. andre klasser (antall linjer i 'mønsterbesvarelsen' er ca. 350) :

```
import easyIO.*;

class Student {
    String navn;
    int saldo;
```

```
    // Evt. metoder som behandler studentene og deres variable.
    // ...
}

class Hybel {
    Student leietager;
    char type; // 'm'=møblert (husleie 5500,-),
               // 'u'=umøblert (husleie 4500,-)
    // Lag også objektvariabel for husleie, og evt. hybelnavn.

    // Evt. metoder som behandler hybler og deres variabler.
    // ...
}

class HybelHus {
    Hybel[][] hyblene = new Hybel[3][7];

    // Her kan du deklarere objektvariabler for økonomi-data:
    // ...

    In tast = new In();
    Out skjerm = new Out();

    // Konstruktør for klassen HybelHus:
    HybelHus() {

        // 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 ett studentnavn:
        //    hyblene[gang][rom] = new Hybel();
        //    hyblene[gang][rom].leietager = new Student();
        //    hyblene[gang][rom].leietager.navn = innfil.inWord(";");
    }

    void kommandoLøkke() {
        int kommando = -1;

        while (kommando != 0) {

            // Legg her setninger som skriver ut menyen:
            skjerm.outln("0 - Avslutt");
           // ...

            skjerm.out("Kommando: "); // Ledetekst
            kommando = tast.inInt();

            switch (kommando) {
            case 0: avslutt(); break;
            case 1: oversiktstabell(); break;
            case 2: registrerNyLeietager(); break;
            case 3: registrerUtflyttingAvLeietager(); break;
            case 4: // ... fyll inn case-er for de 4 andre kommandoene her.
            …… …….
            default:// ...
             }
        }
    }
```

```
    // Skriv de andre metodene her, en for hver kommando:

    void oversiktstabell() {
        // ...

    }

    void registrerNyLeietager() {
        skjerm.outln("** Registrer ny leietager **");
        // ...
    }

    void registrerUtflyttingAvLeietager() {
        // ...
    }

    void registrerInnflyttingFraVenteliste() {
        // ...
    }

    // Delarer hjelpemetoden "fjernFraVenteliste" her.  Husk parameteren.
        // ...


    // Osv... (5 metoder til, inkludert hjelpemetoden i del 7)...
    // ...

}

class Oblig3 {
    public static void main (String[] args) {
        HybelHus hh = new HybelHus();
        hh.kommandoLøkke();
    }
}
```

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 konstruktor (les avsnitt 8.7 og 8.11 i læreboken). En konstruktør er en metode som kalles automatisk når man sier `new`, og er en måte å overbringe parametere til objektene vi lager

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. 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 etasjenummer og andre på hybel innen i en etasje. Nå er problemet at hybelen har et bokstavnavn (A,B,...,H)og ikke et tall. Dette kan løses på følgende måte:

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

Da vil: **`hyblene[etg-1][i]`** finne riktig peker i arrayen. Og motsvarende, hvis vi vet numeret 'i' til en hybel i etasjen , 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 utskrift til **`"venteliste.txt"`** i deloppgave 2, og ønsker å legge noe *til* på slutten av filen (ikke overskrive det som står der allerede), må du åpne fila med *append* – dvs. at når du utfører instruksjonen **`new`**, så har du med en parameter nr. 2 som er **`true`**. Det opprettes da en ny fil hvis filnavnet er nytt, men hvis ikke, åpnes den gamle fila på slutten av innholdet den hadde fra før. F.eks. **`= new Out("venteliste.txt",true).`**

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 .

7. I denne oppgaven skal du ved oppstart lese fila **"hybeldata.txt",** 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 lager du en passende datafil på forhånd i samme mappen der du har programmet, eller tester i koden om fila finnes. Det siste kan gjøres ved å Importerer java.io-pakken (`import java.io.*;`) og der du leser inn fila, tester du slik:

```
if (new File("hybeldata.txt").exists() ) {
      // Kode for å lese fila:
      ... = new In("hybeldata.txt");
      // Les data og opprett objektene som beskrevet på fila...
      ...close();
 } else {
```

```
        // Her kommer programmet hvis datafilen ikke finnes fra før.
        // Du må da opprette alle objektene som representerer
       //  et  hybelhus uten beboere
    }
```

8. For å sikre at den månedlige beregningen av husleie og fortjeneste bare skjer en gang per måned, må du nok både spørre Gulbrand om hvilken måned han skal ha husleiekrav for, samt på fila "hybeldata.txt" ha opplysninger om hvilke måneder det allerede er skrevet ut husleie for.

Leveringskrav

Hver student skal levere sin egenprogrammerte løsning, og plikter å ha lest og forstått følgende krav til innleverte oppgaver ved Institutt for informatikk:

http://www.ifi.uio.no/studinf/skjemaer/erklaring.pdf

Oppgaven skal utføres og leveres individuelt via Joly-systemet. Det ferdige programmet med filen "Oblig3.java" og filen "hybeldata.txt" leveres elektronisk i Joly-systemet innen leveringsfristen (.java-filen må legges inn først av de to filene):
http://obelix.ifi.uio.no:8080/wizard.html

N.B. Husk at Joly-systemet bare virker på Blindern-maskinene (og som oftest hjemmefra hvis du har VPN eller Remote Desktop i Windows) 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).

Tilbakemelding blir gitt innen to uker etter innleveringsfristen (og vanligvis litt raskere). Dersom besvarelsen vurderes til *ikke godkjent*, vil du få en begrunnelse, – og hvis retteren anser at besvarelsen har potensiale til å kunne forbedres tilstrekkelig til å bli godkjent, vil du normalt få anledning til dette, og vil i så fall få en oversikt over hva som må forbedres. Fristen for dette vil normalt være 5 arbeidsdager.

For mer informasjon se Reglement for obligatoriske oppgaver:

www.ifi.uio.no/studier/studentinfo.html#oblig

Lykke til!