Ekstra mange tips til Oblig 3!

12. mars 2007
Are Magnus Bruaset og Arild Waaler
Inst. for informatikk, UiO

Når du skal løse en slik oppgave:

- Les oppgaveteksten nøye! Ikke gjør mer enn det er spørsmål etter!
- Identifiser objektene og bestem datastruktur for hver klasse som du trenger
- Tegn gjerne en figur over objektene og datastrukturen. Da er det lettere å programmere
- Skaff deg oversikt over spesielle teknikker du trenger å bruke, i dette tilfelle lesing fra og skriving til fil
- Skriv koden gradvis og test ut metoder etter hvert som du skriver dem
- Skriv gjerne først de metodene som er nødvendig for å få en enkel prototype av programmet ditt opp og kjøre! Utsett for eksempel filbehandlingen til slutt!

4 klasser er angitt i oppgaveteksten

- class Oblig3
  - her legger vi main-metoden som sparker det hele i gang.
  - fra denne klassen oppretter vi et Hybelhus-objekt og kaller en metode i dette objektet som styrer interaksjonen med brukeren
- class Hybelhus
  - her ligger den sentrale datastrukturen og metoder for alle funksjonene i menyen
  - vi initialiserer datastrukturen for hybelhuset i konstruktøren
- class Hybel
  - info knyttet til en hybel: leietager og utestående
- class Student
  - info knyttet til student: navn og saldo

class Oblig3 – forslag til kode

```
public class Oblig3 {

   public static void main(String[] args) {
      Oblig3 oblig3kjøring = new Oblig3(args);
   }

   Oblig3(String[] inputfiler){
      String datafil = "HaiHus.data";
      String strømfil = "Lysregning.data";
      if( inputfiler.length > 0 )
         datafil = inputfiler[0];
      if( inputfiler.length > 1 )
         strømfil = inputfiler[1];
      Hybelhus utsyn = new Hybelhus(datafil,strømfil);
      utsyn.kommandoløkke();
   }
}
```

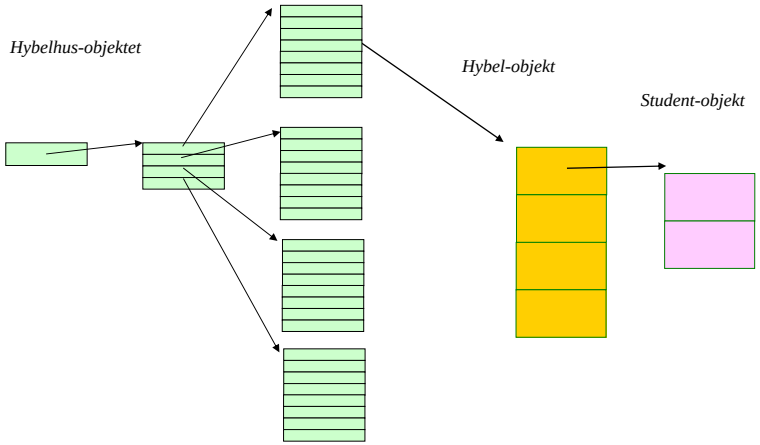
Det eneste main-metoden gjør, er å opprette et objekt av klassen Oblig3

Default filnavn som i oppgaveteksten, men med mulighet for endringer fra kommandolinjen. Bra for debugging – du kan lett bruke andre testfiler!

Filnavnene overføres til konstruktøren for Hybelhus-klassen

Så overfører vi kontrollen til metoden i utsyn som styrer brukerinteraksjon

Enkel skisse av datastrukturen



class Hybelhus: forslag til datastruktur

```
public class Hybelhus {
    private String datafil;
    private String strømfil;

    final int ANT_KOR = 4;
    final int ANT_ROM = 8;
    final String TOM_HYBEL = "TOM HYBEL";

    private int totalFortjeneste;
    private int antallHybelmånederMedTommeHybler;
    private int totaltAntallMåneder;

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

    private Hybel[][] hyblene = new Hybel[ANT_KOR][ANT_ROM];
```

'private' skjuler variable for andre klasser. Dette er god programmeringsskikk!

'final' markerer at verdiene ikke kan endres. Det er vanlig at slike konstanter har store bokstaver.

Variable for informasjonen som leses fra datafilen.

Konstruktøren til Hybelhus-klassen

- Konstruktøren utføres når et objekt opprettes og aldri siden.
- I konstruktøren er det vanlig å legge initialisering av datastrukturen. Konstruktøren mottar gjerne argumenter med informasjon som den trenger for initialiseringen.
- Siden poenget med denne oppgaven er å trene på å lage en objekt-orientert modell, skal jeg vise i detalj hvordan man kan foreta innlesning fra datafilen i konstruktøren ☺
- Tips om lesing fra strømfilen står i oppgaveteksten. Skriving til filer må dere selv finne ut av (se kap. 3 i boka!).
- Vær nøye med å teste ut koden steg for steg! Når dere kommer til skriving av fil, vær nøyaktig med å teste at filen ser akkurat slik ut som den skal etter skriving!
- Tips ved innlesning: Skriv ut til skjerm samtidig som dere leser inn. Lurt for feilsøking!

class Hybelhus: forslag til konstruktør

```
Hybelhus( String datafil, String strømfil ) {
        this.datafil = datafil;
        this.strømfil = strømfil;
        skjerm.outln("Leser fra " + datafil + ":");
        In hybelfil = new In(datafil);
        for(int i = 0; i < ANT_KOR * (ANT_ROM - 1); i++) {
                int gang = hybelfil.inInt(";");  skjerm.out(gang);
                char bokstav = hybelfil.inChar(";");  skjerm.out(bokstav);
                String studentnavn = hybelfil.inWord(";"); skjerm.out(" " + studentnavn);
                int saldo = hybelfil.inInt(";"); skjerm.outln(" " + saldo);
                hybelfil.readLine(); //for å bli kvitt resten av linja.
                hyblene[gang - 1][(int)(bokstav - 'A')] = new Hybel(studentnavn.trim(), saldo);
        }
        totaltAntallMåneder = hybelfil.inInt(";");
        antallHybelmånederMedTommeHybler = hybelfil.inInt(";");
        totalFortjeneste = hybelfil.inInt(";");
        hybelfil.close();
}
```

Nyttig metode i class Hybelhus

```
void listLeietagere(){
    for(int i = 0; i < ANT_KOR; i++){
     for(int j = 1; j < ANT_ROM; j++){  //Tar ikke med fellesarealene her
        skjerm.out(i+1);
        skjerm.out((char)(j+'A') + " ");
        Student student = hyblene[i][j].getLeietaker();
        if(student != null)
            skjerm.outln(student.getNavn() + " " + student.getSaldo());
        else
           skjerm.outln(TOM_HYBEL  + " " + hyblene[i][j].getUtestående());
     }
    }
}
```

Her har jeg brukt en vanlig Java-konvensjon når metodene for å hente dataverdi begynner med get. Metodene for å oppgi verdier skal begynne med set ifølge konvensjonen. NB! Bruk alltid slike metoder for å aksessere variable i en annen klasse og la variablene være deklarert private. Alt annet er dårlig programmeringsskikk og gir i lengden opphav til stygge bugs!

Videre jobbing med class Hybelhus:

- Begynn gjerne med å lage metoden for å styre menyen – jeg har kalt denne ”kommandoløkke()”. Identifiser alle metodene du skal kalle herfra.
- Tenk igjennom hva hver metode skal gjøre og hvordan den skal gjøre det FØR du begynner å skrive kode. Bruk gjerne små skisser og stikkord.
- Spesielt må du tenke over hvilke metoder du trenger fra de andre klassene og hvordan dataverdier skal endres.
- Utfordringen er å bryte oppgaven du skal løse ned i mange småproblemer – som hver for seg er enkle å løse – og så sette sammen løsningene av småproblemene slik at de sammen kan løse hele oppgaven.

... og de andre klassene

- For klassene Hybel og Student gjelder det samme: list opp alle metodene du trenger og tenk over hvilke data disse trenger. Hvordan skal de få tak i informasjonen de trenger?
- Husk at du aldri bør kode overalt samtidig. Når du har fått en viss oversikt, skriv kode for de enkle metodene først og test dem grundig før du går videre.
- Ikke lås deg til en bestemt løsning og datastruktur fra starten. Ofte mangler vi oversikt når vi begynner, og blir nødt til å endre konstruksjonen av programmet underveis fordi det dukker opp ting vi ikke hadde tenkt på da vi startet. Dette er heller regelen enn unntaket!

... og til slutt vil jeg bare ha sagt at:

- Å jobbe med obligene er det du lærer mest av i kurset.
- Bruk boka aktivt som oppslagsverk i denne prosessen. Studer relevante programeksempler i boka! Det er utrolig mye lettere å lære seg noe når du trenger det for å løse en oppgave – enn å lære seg det bare for å lese til eksamen!
- Prøv å bli litt kjent med Java API samtidig. Det er gøy! Java-verdenen er stor og interessant. Prøv deg frem på egen hånd!
- ... og lykke til, da!