INF1000 – Forelesning 9

15. mars 2010

Tips til oblig 3, separatorer i easyIO,
Eliza (bruk av HashMap), mer om metoder og klasser

Christian M. Hansen
Institutt for informatikk, UiO

Generelt om oblig 3

- 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!

2

Oppgaveteksten – kort sammendrag

- Guldbrand Grå driver hybelhuset Utsyn
- Tre etasjer (1-3) med seks hybler (A-F) + fellesrom i hver etasje
- Husleie 5.000,- kr/mnd i etasje 1 og 2, 6.000,- kr/mnd i tredje etasje
- Utgifter: 1.200,- kr/hybel/mnd uavhengig av om bebodd. Fellesrom: 1.700,- kr/rom/mnd
- Beboerne betaler husleie, Guldbrand dekker alle andre utgifter
- Overskuddet hver mnd er samlet husleieinntekt minus samlede utgifter
- Depositum på 10.000,- ved innflytting, utflyttingsgebyr kr. 650,-
- Hver leietaker har en saldo, som ved månedens slutt må inneholde nok til å betale leie (varierer med etasje!)
- Hvis saldo går under minus én husleie, blir leietaker kastet ut

3

Deloppgaver/menyvalg

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

4

Fire klasser i oppgaveteksten

- class Oblig3
  - her legger vi main-metoden som sparker det hele i gang.
  - fra denne klassen oppretter vi et Utsyn-objekt og kaller en metode i dette objektet som styrer interaksjonen med brukeren
- class Utsyn
  - 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 husleie
- class Student
  - info knyttet til student: navn og saldo

5

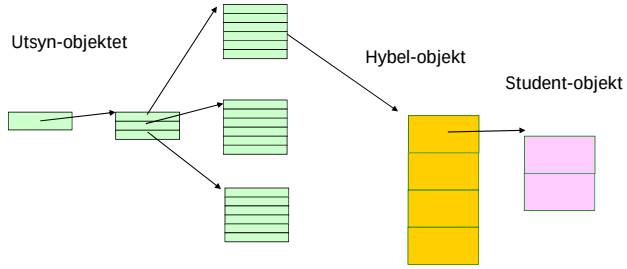
class Oblig3 – forslag til kode

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

Vi overfører kontrollen til metoden i utsyn som styrer brukerinteraksjon

6

Enkel skisse av datastrukturen



7

class Utsyn

```
public class Utsyn {
   Hybel[] hybler = new Hybel[3][6];

   In tast = new In();
   Out skjerm = new Out();
   final String DATAFIL = "hybeldata.txt";
   final String TOM_HYBEL = "TOM HYBEL";

   int måned;
   int år;
   int totalFortjeneste;
   int totaltAntallMåneder;

   Utsyn() { … }
   void ordreLøkke() { … }
   void skrivOversikt() { … }
   void registrerNyLeietaker() { … }
   …
}
```

`final` angir at variabelen ikke kan endres.

Variable for informasjonen som leses fra datafilen.

8

Konstruktøren til Utsyn-klassen

- Konstruktøren utføres når et objekt opprettes (og aldri siden).
- I konstruktøren er det vanlig å gi startverdier til datastrukturen. (Konstruktøren kan motta argumenter med informasjon som den trenger for å gi startverdier.)
- Siden poenget med denne oppgaven er å trene på å lage en objekt-orientert modell, skal viser vi i detalj hvordan man kan foreta innlesning fra datafilen i konstruktøren
- 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!

9

class Hybelhus: forslag til konstruktør

```
Utsyn() {
   skjerm.outln("Leser fra " + DATAFIL + ":");
   In hybelfil = new In(DATAFIL);
   måned = hybelfil.inInt(";");
   år = hybelfil.inInt(";");
   totalFortjeneste = hybelfil.inInt(";");
   totaltAntallMåneder = hybelfil.inInt(";");
   for (int i=0; i < hybler.length * hybler[0].length; i++) {
      int etg = hybelfil.inInt(";");  skjerm.out(etg);
      char bokstav = hybelfil.inChar(";");  skjerm.out(bokstav);
      int saldo = hybelfil.inInt(";"); skjerm.outln(" " + saldo);
      String studentnavn = hybelfil.inWord(";"); skjerm.out(" " + studentnavn);
      hybelfil.readLine(); //for å bli kvitt resten av linja
      Hybel hybel;
      if (etg == 3) {
         hybel = new Hybel(6000);
      } else {
         hybel = new Hybel(5000);
      }
      hybler[etg - 1][(int)(bokstav – 'A')] = hybel;
      if (!studentnavn.equals(TOM_HYBEL)) {
          hybel.settLeietaker(new Student(studentnavn.trim(), saldo));
      }
   }
   hybelfil.close();
}
```

Nyttig metode i class Utsyn

```
void listLeietagere(){
   for (int i=0; i < hybler.length; i++) {
      for (int j=0; j < hybler[0].length; j++) {
         skjerm.out(i+1);
         skjerm.out((char)('A' + j) + " ");
         Student student = hyblene[i][j].getLeietaker();
         if (student != null)
            skjerm.outln(student.getNavn() + " " +
                    student.getSaldo());
         else
            skjerm.outln(TOM_HYBEL);
      }
   }
}
```

Java-konvensjon:
- Metodene for å hente dataverdi begynner med **get**
- Metodene for å angi verdier skal begynne med **set**
- Bruk alltid slike metoder for å aksessere variable i objekter
- La variablene være deklarert **private**
- Alt annet er dårlig programmeringsskikk og gir i lengden opphav til stygge bugs!

11

Videre jobbing med class Utsyn...

- Begynn gjerne med å lage metoden for å styre. 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.

12

... 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!

13

... og til slutt:

- 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!

14

1. Innlesning i easyIO, bruk av skilletegn

- Alle filer betraktes som en strøm av tegn (inludert de vi ikke alltid ser CR = vognretur, LF = linjeskift, og en serie ('ukjente') blanke tegn.)
- Lesing styres av en 'lese-pil'
  (som etter åpning av fila, peker på tegn nr. 1)
- Ved lesing, beveger lese-pilen seg høyre-over.
- Lese-pilen går aldri bakover.

15

1. Separatorer (skilletegn) i easyIO

1. Tegnene som leses av easyIO deles I to:
   A) De tegnene som skal skal leses og tolkes som data.
      - = de tegnene som ikke er separatortegn.

   B) Separator-tegnene som ligger foran og etter data (avgrenser det som leses som data).
      - Separatortegn er **alltid**: CR, LF og skjulte (rare) blanke tegn.
      - Når systemet starter er også vanlig blank (mellomrom) og tabulator separatortegn.
      - Hva som er separatotegn kan brukerne selv velge med unntak av de tegnene som alltid bukes.
2. Når en metode I easyIO kalles - eks. inInt(sep); så:
   1. Først leses det forbi evt. gamle (forrige) separator-tegn.
   2. Deretter leses det forbi evt. nye separator-tegn
   3. Nå står lese-pilen på første ikke-separator.
   4. Alle tegn fram til første nye separator-tegn leses som data.
   5. Lesepilen står etter lesing av data på første-ikke-separator funnet i pkt.4.

16

Redusert ”hybeldata.txt” (legg merke til alle mellomrom (’blanke’) i fila) :

```
2;C;2400;Ole Brun
3;A;3000;TOM HYBEL
```

For å si det enklere:

Først hoppes det over først alle gamle og så alle nye skilletegn i 'sep'.

Deretter leses alt frem til neste nye skilletegn ***som de data vi egentlig er interessert i.***

Eks. `int i =fil.inInt(”;”);`

vil først hoppe over evt, gamle skilletegn og så ';' , så lese de neste tegnene fram til første 'blanke' eller ; og prøve så å tolke det som står mellom disse to skilletegnene som et heltall.

17

2.Lage et psykolog-program

- Tar inndata – et spørsmål/en setning fra pasienten og svare som en dårlig psykolog
- Ha en fil med enkeltord, og hvis pasienten bruker et av ordene, så svare med tilhørende setning
- Bruke en HashMap for å holde par av ord, setning
- Eks:
  - far, Fortell meg om din far
- Må ha noe å si hvis vi ikke får treff:
  - Så interessant, fortell meg mer om dette.

18

```
> hei
 Hei, jeg heter Sigmund, hva heter du
> Jeg heter Arne
 Så det er altså navnet
> Ja
 Fint, skal vi snakke om din barndom?
> Ja og særlig om mor
 Fint, skal vi snakke om din barndom?
> Mor sier jeg jo
 Lever fortsatt din mor
> nei
 Så du mener ikke det
> hun slo meg ofte
 Var det ofte at noen slo?
> ganske ofte
Interessant. Fortell mer.
>
```

19

Programskisse

```
import easyIO.*;
import java.util.*;
class Eliza {
  public static void main(String [] args) {

  }
}
class Samtale {
  HashMap hash = new HashMap();
  In tast = new In();
  void lesFraFil() {

  }
  void snakk() {

  }
}
```

20

```
import easyIO.*;
import java.util.*;

class Eliza {

  public static void main (String [] args) {
     if (args.length !=1) {
       System.out.println(" bruk: >java Eliza <fil-med-ord> ");
     } else {
       Samtale sam = new Samtale();
       sam.lesFraFil(args[0]);
       sam.snakk();
       }
  } // end main
}
```

21

```
class Samtale {
    HashMap <String,String>hash =
              new HashMap<String,String>();
    In tast = new In();

    void lesFraFil (String filnavn) {
      In fil = new In(filnavn);
        while (!fil.lastItem()) {
          String søkeord = fil.inWord();
          String svar = fil.inLine();
          hash.put(søkeord, svar);
        }
        fil.close();
        System.out.println
           ("Antall ord lest: " + hash.size());
    }
```

22

```
void snakk() {
        while (true) {
          System.out.print("> ");
          boolean funnetMatch = false;
          do {
             String ord = tast.inWord().toLowerCase();
             if (hash.containsKey(ord)) {
                 String svar = hash.get(ord);
                 System.out.println(svar);
                 funnetMatch = true;
             }
          } while (tast.hasNextChar() && !funnetMatch);

          if (!funnetMatch) {
             System.out.println("Interessant. Fortell mer.");
          }
          if (tast.hasNextChar()) {
             tast.readLine();  // Tømmer inputbufferet
          }
        }
    } // end snakk
}
```

23

Ordfil.txt

```
far   Fortell meg mer om din far
faren Hadde du et vanskelig forhold til din far?
slo   Var det ofte at noen slo?
lei    Du sier du er lei deg, hvorfor det
mor    Lever fortsatt din mor
penger Er du bekymret for om du har nok penger
sint   Hvorfor bruker du 'sint' - er du selv sint
glad   Så bra
ikke   Forklar dette nærmere
vær    Blir du deprimer av dårlig vær
nei    Så du mener ikke det
ja     Fint, skal vi snakke om din barndom?
barn   Har du barn eller barnebarn?
barnebarn Hva heter de
datter  Hvor gammel er hun?
hei    Hei jeg heter Sigmund, hva heter du
jeg    Hvordan føler du deg
heter  Så det er altså navnet
gjenta Vil du heller snakke om din mor?
```

24

ALICE: En kunstig intelligens-basert prate-robot

- ALICE = Artificial Linguistic Internet Computer Entity
- http://alice.pandorabots.com

3. Hva er en metode

- En metode er en valgfritt antall programsetninger vi gir et navn
- All kode i programmet er inne i en metode (som igjen er inne i en eller annen klasse)
- Skille mellom
  - å *deklarere* en metode (= skrive Javakode for og kompilere)
  - *Utføre* en metode (det som skjer når vi kaller den)
  - Når vi deklarerer en metode, skjer det 'ingen ting '
- En metode blir utført hver gang den kalles fra koden i en annen metode:
  - da hopper utførelsen av programmet til starten av den kalte metoden
  - har den kalt metoden parametere, kopieres verdiene brukt i kallet til metodens parameter-variable (de er som lokale variable i den kalte metoden)

26

Hva skjer når vi kaller en metode

- Når vi kaller en metode, blir det opprettet et **metodeobjekt** og vi kopierer over verdiene brukt i kallet til parameterne
- Dette metodeobjektet
  - inneholder alle lokale variabler og parameterne til metoden
  - når setningene i metoden utføres, brukes disse variablene og parametrene av metoden
  - metodeobjektet fjernes automatisk når metoden er ferdig utført og returnerer
- Merk forskjellen på å deklarere en metode, og at den utføres.

27

```
class C {

  int  skrivAntall(int i){
       System.out.println(" Du har kalt meg med:" + i);
       return i+10;
}}

class D
{
  static int dobbel( int k) {
       return 2*k;
  }

  void gjørMye(C cc, int v) {
       System.out.println(" gjørMye kalt");
       int j = cc.skrivAntall(v);
       System.out.println(" 1.verdien av j:" + j);
       j = dobbel(j);
       System.out.println(" 2.verdien av j:" + j);
       System.out.println(" 3.verdien av skrivAntall(j):"
                                  + cc.skrivAntall(j) );
  }

  public static void  main ( String[] args)  {
      C c = new C();
      D megSelv  = new D();
      megSelv.gjørMye(c,2);
}}
```

```
>java D
gj°rMye kalt
Du har kalt meg med:2
1.verdien av j:12
2.verdien av j:24
Du har kalt meg med:24
3.verdien av
  skrivAntall(j):34
```

28

Hvorfor bruke metoder

- Vi deler opp programmet i metoder fordi:
  - Noen program setninger brukes *flere* steder, eller:
  - Vi vil dele opp programmet i mindre deler
    - Ingen metode bør være lenger enn 30 linjer (og helst mindre)
  - Hver del gjør noe veldefinert som fremgår av navnet:
    - regner ut en bestemt formel
    - skriver ut en meny
    - leser noen data fra terminal eller fil
    - tegner ut opplysninger på skjermen
    - .....

29

Problemløsning med metoder

- Når vi har laget en metode, og vi har forsikret oss om at den er 'riktig', så har vi laget en *ny operasjon*
- Vi kan nå i resten av koden tenke at vi nå har en slik operasjon tilgjengelig og nytte denne som om den var innebygd i Java
  - eks: skrive ut en meny, regne ut en bestemt formel,..
- Vi trenger da ikke tenke på alle detaljene om *hvordan* denne operasjonen blir utført, bare *at* den blir gjort.
- Vi har da laget et (lite) verktøy som kan gjenbrukes og lettere løse vårt større problem (hele systemet)
- Denne måte å programmere på heter *bottom-up* programmering og nyttes mye.
  - Eks: Java-biblioteket kan best forstås som en diger verktøykasse med nyttige operasjoner og datastrukturer vi kan (og ofte bør) bruke for å lage vårt program

30

Hva er en klasse

- En klasse er en beskrivelse av hvordan *ett* objekt av en bestemt type i vårt problem er.
  - Inneholder variable som beskriver egenskaper for ett slikt objekt – eks:
    - Navn, adresse, studiepoeng, kurs... for klassen Student
    - Registreringsnummer, eier, type, årsmodel for klassen Bil
  - Inneholder metoder som er fornuftig handlinger for ett slikt objekt – eks:
    - skrivUtVitnemål(), meldPåEmne(),.. i klassen Student
    - beregnÅrsavgift(), skiftEier(),.. i klassen Bil

31

Skille mellom deklarasjon og bruk av en klasse

- Når vi deklarerer en klasse (= skriver Javakode for) skjer det 'ingen ting' i programmet
- Når vi oversetter og starter opp programmet vårt med javac og java, skjer 'lite':
  - De variable og metodene det står static foran er tilgjengelig
  - Ingen kode (med unntak av main) utføres
- Først når vi sier **new** på en klasse, får vi laget et objekt av klassen
  - Objektet inneholder alle variable og metoder som ikke har static foran deklarasjonen (objekt-variable og – metoder)
  - Når vi sier new, kaller vi en konstruktør-metode i klassen, og først når den er ferdig, returnerer new det med det nye objektet

32

```
class Konto1 {
  String eier;
  int kontoNum, saldo = 0;

  Konto1(String e) {
       eier = e;
  }

  void  settInn(int beløp) {
      saldo = saldo + beløp;
  }

  boolean taUt(int beløp) {
        // moderne bank med muligheter for overtrekk
        saldo = saldo - beløp;
        return saldo > 0;
  }
}

class Bank1
{
  Konto1 [] kontiene = new Konto1[100000];

  public static void  main( String[] args)  {
     Bank1 b = new Bank1();

     for (int i = 0; i < b.kontiene.length; i++) {
        b.kontiene[i] = new Konto1("kunde nr." + i);
        b.kontiene[i].settInn(100);
    }
 }
}
```

33

Forskjeller mellom klasser og metoder

- Begge lager objekter når de kalles, men:
- Et metode-objekt:
  - fjernes når metoden returnerer
  - inneholder 'bare' variable og parametere som alle er skjult for resten av programmet
- Et objekt laget med **new** fra en klasse:
  - er i hukommelsen etter at det er laget (så lenge det minst er en peker som peker på det)
  - kan inneholde både metoder og variable, som kan nyttes av resten av programmet (med en peker og .)

34

Ikke alt i et objekt bør være synlig fra resten av programsystemet - innkapsling

- Vi ønsker ofte at resten av systemet bare skal se deler av et objekt
  - eks: **`int saldo`** i Konto1-objektet bør være skjult, resten av programmet skal bare bruke **`settInn() og taUt()`** metodene.
- Vi kan regulere tilgangen til variable og metoder ved å sette enten :
  - **`private`**
  - **`public`**
  - **`protected`**
- foran en metode eller deklarasjonen av en variabel

35