Oblig3-forklaringer (særlig π -oppgaven)

Arne Maus
Inst for informatikk, 20.okt. 2006

Dagens oversikt

I) Om konstruktører og this

- Feil dere gjør nå:
  - Dere må lage metoder i klassene **og** kalle dem

II) Generelt om hvordan lage OO-programmer

III) En god del tips til π -oppgaven, grovskisse av en løsning

IV) Noen tips til Husleie-oppgaven (rette uklarhet om fortjeneste til Gulbrand mm.)

I) Konstruktører – startmetoder i klasser

- Når vi lager et objekt av en klasse med **new**, kaller vi egentlig en metode som heter det samme som klassen (derfor parentesen bak klassenavnet).
- Vi får automatisk med en slik konstruktør-metode fra oversetteren dersom vi ikke skriver en slik konstruktør selv. Den vi får automatisk er uten parametere og gjør ingen ting.
- Konstruktører nyttes i all hovedsak til å gi fornuftige startverdier for variable i objektet som dannes.
- De konstruktørene vi skriver kan ha parametere.
- Konstruktørene skal ikke ha noen type foran seg, heller ikke void.
- Vi kan ha flere konstruktører i en klasse, men da må parameterne være ulike i antall eller typen av parametrene

Eksempel Student med en konstruktør

```
class Student {
   String navn;
   Kurs [] mineKurs = new Kurs[3];

   Student(String navn, Kurs [] k){
     this.navn = navn;
     for (int i = 0; i<k.length; i++ ){
         mineKurs[i] = k[i];
         mineKurs[i].antStudenter++;
     }
   }
```

Eksempel Student med 2 konstruktører

```
class Student {
   String navn;
   Kurs [] mineKurs = new Kurs[3];

   Student() {
     mineKurs = new Kurs[0];
   }

   Student(String navn, Kurs [] k){
     this.navn = navn;
     for (int i = 0; i<k.length; i++ ){
         mineKurs[i] = k[i];
         mineKurs[i].antStudenter++;
     }
   }
```

this

- Av og til trenger vi en peker til det objektet metoden vi utfører er inne i. Java-ordet this gir oss alltid det.
- Brukes i to situasjoner:
  - Vi har en konstruktor, og parametrene til denne heter det samme som objekt-variable i objektet. Eks:

```
class A {
   int antall;
   A (int antall ){
       this.antall = antall;
   }
} // end A
.. A apek = new A(12);
```

  - Vi skal kalle en metode i et annet objekt (gjerne av en annen klasse). Da kan vi bruke this for å overføre en parameter til denne metoden om hvilket objekt kallet kom fra.

II) Slik lager du OO-prgrammer

- Hvilke typer av objekter har jeg i problemet ?
  - Lag en **klasse** for hver slik type av gjenstander
  - Hvilke data er lagret i et objekt av hver slik klasse
    - Deklarér disse dataene i toppen av klassene
- Hva skal de ulike objektene kunne gjøre ?
  - Lag en **metode** for hver av disse operasjonene i den tilhørende klassen

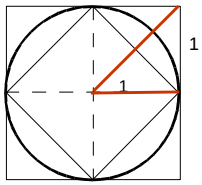
Når du kjører programmet:

- I main:
  - Les evt. parametere i String-arrayen ‘args’
  - Lag de objektene du har oversikt over – si **new** på den/de klassene dette gjelder
  - Kall (minst) en metode i ett av objektene du har laget (enten konstruktøren eller en annen i objektet)
- I enhver metode som kalles:
  - Lag evt. nye objekter (med new) som du trenger der
  - Gjør evt. egne beregninger
  - Kall evt. andre metoder i objekter du har pekere til for å gjøre deler av det denne metoden skal gjøre

III) π-oppgaven

- Gi trening i bruk av klasser og objekter, arrayer og enkel utskrift til fil samt presis programmering av (kjente) matematiske metoder.
- π = Forholdet mellom diameter og omkrets av en sirkel
- Gitt to formler
  - Machins formel (løs denne)
  - Størmers formel (vis hvordan løses med denne etter å ha løst Machin)

tan(45°) = tan(π/4) = 1



$\pi/4 = \arctan(1)$

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$

$$\pi = 16\arctan\frac{1}{5} - 4\arctan\frac{1}{239} = 16\left[\frac{1}{5} - \frac{1}{3*5^3} + \frac{1}{5*5^5} - \frac{1}{7*5^7} + \ldots\right] - 4\left[\frac{1}{239} - \frac{1}{3*239^3} + \frac{1}{5*239^5} - \frac{1}{7*239^7} + \ldots\right]$$

Leddene i de to Arctan-rekkene

$$\pi = 16\arctan\frac{1}{5} - 4\arctan\frac{1}{239} = 16\left[\frac{1}{5} - \frac{1}{3*5^3} + \frac{1}{5*5^5} - \frac{1}{7*5^7} + \ldots\right] - 4\left[\frac{1}{239} - \frac{1}{3*239^3} + \frac{1}{5*239^5} - \frac{1}{7*239^7} + \ldots\right]$$

$$L5_0 = \frac{1}{5}, \quad og \quad L5_i = -L5_{i-1}\frac{2i-1}{(2i+1)*5^2} \quad for \quad i > 0$$

$$L239_0 = \frac{1}{239}, \quad og \quad L239_i = -L239_{i-1}\frac{2i-1}{(2i+1)*239^2} \quad for \quad i > 0$$

Problem

- Vi skal lage to arctan-rekker
  - løser vi den ene, så løser vi lett den andre (bare ett tall forkjellig (5 mot 239)
- Skal (må) bruke heltalls-regning
- Vi skal ha 10 000 siffer
  - i heltallsarrayer
  - 4 sifre i hvert heltall ⇒ ca. 2500 tall i en slik array

Regning med flere sifre i en gruppe mellom to 'mange-sifrete' tall

```
1) Addisjon to-og-to (a = a + b) :

mente ->     1   1   1
            32  87  77  94    a
          + 01  44  22  61    b
          --------------------
          =  34  32  00  55    ny verdi av a

2) Multiplikasjon to-og-to (a = a * b)


      87  27  94  *   74  44
----------------------
        38  40  29  36
    64  58  67  56
----------------------
=   64  97  07  85  36
```

Nøyaktig forklaring på den første linjen i multiplikasjonen: 94*44 = 4136, 36 noteres og 41 blir mente. Neste: 27* 44 = 1188, Vi legger til menten, 41 fra forrige multiplikasjon, og får 1188+41=1229; 29 noteres som svaret og 12 blir mente til neste siffergruppe. Så 87*44 = 3828 + mente 12 = 3840. 40 noters for denne siffergruppen og 38 blir menten. Siden menten til neste er mindre enn 100, går denne enkelt som mente til neste gruppe igjen.

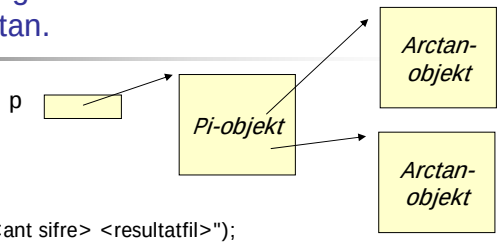
Multiplikasjon i oppgaven

- På forestående side er multiplikasjonen mellom to tall, ***hver*** med mange sifre
- I oppgaven blir multiplikasjon og divisjon mellom et mangesifret tall **og** ett (enkelt) heltall (int). Det blir enklere enn det forklaringen foran tyder på, men vanskelig nok for multiplikasjon.

π -oppgaven – skisse av ett mulig program som løser oppgaven

- Løsningen ***skal*** bestå av 3 klasser:
  - Oblig3Pi:
    - Inneholder main, lager ett objekt av Pi-klassen og kaller en metode i dette Pi-objektet som regner ut svaret
    - skriver ut svaret (som nå ligger i Pi-objektet)
  - Pi:
    - Inneholder en int-array med plass til de første 10 000 sifrene i π
    - Lager to objekter av klassen Arctan, og kaller metoder i hver av disse som regner ut verdien av hver sin Arctan-rekke
    - Adderer/subtraherer disse to rekkene sammen til et svar på verdien av π som lagres i dette Pi-objektet
  - Arctan
    - Har to int-arrayer: en for verdien av nåværende ledd og en for summen (så langt) av hele rekka (begge med plass til 10 000 sifre)
    - Går i løkke:
      - **Regner ut verdien av neste ledd (i arrayen med 10 000 sifre)**
      - **Adderer/subtraherer denne til verdien av hele rekka, siffergruppe for siffergruppe**

Skisse av Arnes programkode til: Oblig3Pi, Pi og Arctan.



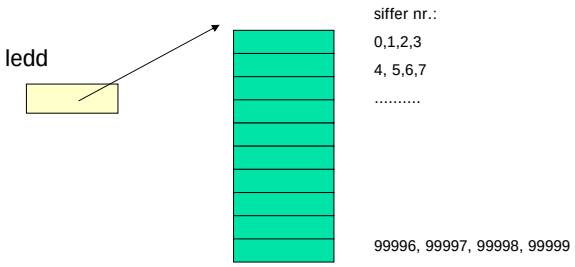

```
class Oblig3Pi{
 public static void main(String[] args) {
  if (args.length != 2 ) {
   System.out.println("bruk: >java Pi <ant sifre> <resultatfil>");
  }else{
      Pi p = new Pi(new Integer(args[0]), args[1]);
      p.print();
}}}

class Pi{
   <data, bla. array for pi >
   Pi(int numDigits, String file) {....}
    void print() {}
    void add (Arctan a) { }
    void sub (Arctan a) {}
   <noen flere metoder?>
} // end class Pi
```

```
class Arctan{
   <data, bla. array for neste ledd
              og hele arctan-rekka>
    Arctan(.............) {
     }
    void beregnSerie() { }
    void lagNesteLedd(int i) { }
    void div (long div) { }
    void mult (int mult) {
    void addToSeries(){}
    void subFromSeries() {}
}
```

Hvordan representere:

- Pi (svaret)
- Verdien av en arctan-rekke
- Verdien av *ett ledd* i en slik arctan-rekke
- svar på alle:
  - Hver som en int-array med litt over 10000/4 = ca. 2500 elementer
  - Eks ett Ledd i en arctan rekke



Hva er svaret

- **Fasit:** De første sifrene er: 3.1415 9265 3589 7932 3846 2643 3832 7950 2884 1971 og på plass 761 etter komma finner vi: 4999 9998 3729 (merk seks 9-tall på rad) og de siste sifrene er 6999 8922 5695 9688 1592 0560 0101 6552 5637 5678

  (og fra plass 10001): 5667 2279 6619 ; og som nevnt i tips 6, kan de aller siste sifrene være gale.

Mindre og mindre ledd

1. Du vet at rekkene du skal regne ut konvergerer og at hvert ledd er mindre enn det foregående og at svaret bare har ett siffer foran komma (slik at det blir en del tester du slipper å gjøre i koden).

hvordan regne ut negative ledd

- Det lønner seg å regne ut hele tiden å regne den positive verdien av hvert ledd i arctan-rekkene og så hhv. trekke fra og legge denne til arrayen som holder summen av rekka.

Litt flere sifre enn 10 000

- Du bør av regne med litt flere sifre, f.eks. 4 ekstra, enn det kravet oppgaven stiller (10 000) fordi de siste sifrene lett blir unøyaktige (du vet ikke om alle de små bidragene fra de leddene utenfor det siste leddet du regner ut, ville gi ett tillegg i det siste, og evt. også det nest siste sifferet,...)

Ikke multipliser opp en unøyaktighet

- Det kan lønne seg, når du regner ut neste ledd, å foreta multiplikasjonen før divisjonen (gir mindre feil i de siste sifrene).

start ikke helt til toppen av arrayen

- Du bør i tillegg gjøre arrayene dine 1-2 plasser lengre i hver ende enn det som skal til for å representere de ca.10004 sifrene du regner på. Dvs. at du plasserer 3 (heltallsdelen av pi) i plass nr. 1 og ikke nummer 0 i arrayene dine.
  Dette fordi når du regner på en array 'a', så vil du ofte referere til a[i-1] og a[i+1] uten at du hele tiden ønsker å sjekke om du har kommet utenfor arrayen.

Skriv med regnereglene **nøyaktig**

1. Det lønner seg å grundig å skrive ned reglene for de fire regneartene (regnes det fra venstre mot høyre eller høyre mot venstre, hvordan og når overføres mente og må resultatet i en siffergruppe 'normaliseres' etter utregningen,...). Her vil du få bruk for % og / operatorene (rest og heltallsdivisjon).

Utskrift med 0 også først i siffergruppe

- Hvis du skriver ut med easyIO i siffergrupper med fire tall og med en blank mellom (f. eks 10 slike siffergrupper per linje), vil du oppleve at eventuelle null først i en slik gruppe ikke skrives ut. Hvis du først omgjør tallene til en String med følgene metode, får du også med 0-er først i siffergruppa som så kan skrives ut med metoden for å skrive ut en String i easyIO:

```
String lagString(int a) {
       String s = ""+a;
       while ( s.length() < 4)
          s= "0" + s;
       return s;
}
```

Regn bare på de deler av et ledd som er ≠ 0

1. Det vil halvere regnetiden hvis du passer på hvor langt ned i arrayen leddene i den arctan-rekken du nå regner ut alle har blitt null. Når du så begynner beregning av neste ledd, begynner du på det stedet fordi du vet at neste ledd er mindre. Husk at ved multiplikasjon vil du kunne 'heve' denne null-grensen (midlertidig).

2. I programmet heter denne grensen: 'isZero', og peker på det ord i arrayen hvor alle ord ovenfor dette = 0 (så langt vi hittil er kommet i utregning av 'neste ledd')

En av add-metodene (max = 10 000)

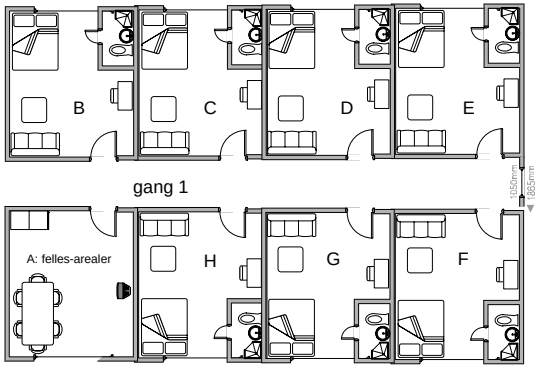
```
void addToSeries(){
    // adder serie  til serie
    int res;
    for (int i = num; i>= isZero; i--){
        res = serie[i] +ledd[i];
        serie[i]= res % max;
        int j=i;
        while(res >= max) {
            res= res/max + serie[j-1];
            serie[j-1] = res% max;
            j--;
        }
    }
} // end Arctan.add
```

Størmers formel er nå lett

$$\pi = 4\left(44\arctan\frac{1}{57} + 7\arctan\frac{1}{239} - 12\arctan\frac{1}{682} + 24\arctan\frac{1}{12943}\right)$$

II) Husleie-oppgaven – en rekke krav

- Grådig hybel-vert
- 4 x 7 hybler + 4 fellesrom



Feil/unøyaktigheter i oppgaven

- s2:"..har programmet gått i 2 måneder med 32 ledige hybler i den første og 28 i den neste, viser dette tallet 60."
  - Kommentar – siden det max er 27 ledige hybler, er dette lett uforståelig (kommer fra en versjon av med dobbelt så mange hybler)
- Hvem får "torpedogebyret" på kr. 1000.- ?
  - Gulbrand (det står i teksten), men da har han utgifter tillegg – og utregningen av han totale fortjeneste blit 'gal'
  - Torpedoen ?, men da er teksten feil (og torpedogebyret skal da ikke inkluderes i Gulbrands fortjeneste

Svar: Velg den løsningen dere foretrekker og skriv det inn som en kommentar i programmet

Bevisste 'uklarheter' i oppgaven

- Gulbrands 'fortjeneste' – kan være både:
  - for én student på én måned.
  - For én måned totalt hele hybelhuset
  - I sum for alle måneder

  Finn ut fra situasjonen hva det er snakk om

Meny

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

Tre filer

- HaiHus.data
  - Skrives og leses
  - Fil med opplysninger om hyblene
  - Er 'databasen' for systemet
- Torpedo.txt
  - Skrives – ordre til en håndfast venn
- Lysregning.data
  - Leses bare – strømforbruk (husk å hoppe over/spesialbehandle fellesrommene)

Format på filene

Hvodan sikre at Månedskjøring med husleie bare kjøre én gang?

- Du må vel førstspørre Gulbrand hvilken måned han registrerer for
- Så må du ha i programmet ha opplysning om hvilken måned som sist ble registrert
  - Skal denne ligge også på fil ?

"HaiHus.data" – eksempel på format

Opplysninger om hver hybel:

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

Tipsene

1. Programstrukturen – de 4 klassene: Oblig3, HybelHus, Hybel og Student - står i oppgaven
2. Konstruktører – se foran
3. Testing på null
4. Konvertering til/fra bokstav til tall
5. Åpning av en fil med 'appand' skriving sist på filen, ikke overskriving.