BOKMÅL

UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet – prøveeksamen

Prøveeksamen i :	INF1000 — Grunnkurs i objektorientert programmering
Prøveeksamensdag :	Torsdag 22. november 2007
Tid for prøveeksamen :	14 – 17
Oppgavesettet er på :	14 sider
Vedlegg :	Ingen
Tillatte hjelpemidler :	Alle trykte og skrevne

- ***Den virkelige eksamen vil ha en forside som dette.***

- Les ***nøye*** gjennom hver oppgave før du løser den. For hver oppgave er angitt det maksimale antall poeng du kan få hvis du svarer helt riktig. Summen av poengene er 252, slik at f.eks 4 poeng tilsvarer omlag 3 minutter og 10 poeng omlag 7 min. (hvis du regner med å komme igjennom alt). Pass på at du bruker tiden din riktig.

- Kontroller også at oppgavesettet er komplett før du begynner å besvare det. Dersom du savner opplysninger i oppgaven, kan du selv legge dine egne forutsetninger til grunn og gjøre rimelige antagelser, så lenge de ikke bryter med oppgavens "ånd". Gjør i så fall rede for forutsetningene og antagelsene du gjør.

- Dine svar ***skal*** skrives på disse oppgavearkene, og ikke på separate ark. Dette gjelder både spørsmål med avkrysningssvar og spørsmål hvor du bes om å skrive programkode. I de oppgavene hvor det skal skrives programkode, anbefales det at du først skriver en kladd på eget ark før du fører svaret inn i disse oppgavearkene på avsatt plass.

- Noen av spørsmålene er flervalgsoppgaver. På disse oppgavene får du poeng etter hvor mange korrekte svar du gir. Du får ikke poeng hvis du lar være å besvare et spørsmål, eller dersom du krysser av begge svaralternativer.

- Hvis du har satt et kryss i en avkrysningsboks og etterpå finner ut at du ikke ønsket å krysse av der, kan du skrive "FEIL" like til venstre for den aktuelle avkrysningsboksen.

- Husk å skrive såpass hardt at besvarelsen blir mulig å lese på alle gjennomslagsarkene, men ikke legg andre deler av eksamensoppgaven under når du skriver.

Oppgave 1 (2 poeng)

a) Hvor mange int-verdier er det plass til i arrayen ”bingo”?

```
int[][] bingo = new int[2][];
bingo[0] = new int[10];
bingo[1] = new int[20];
```

Svar: .....

Oppgave 2 (20 poeng)

Er disse programsetningene lovlige i Java?

JA	NEI	
☐	☐	`int x = 4 / 2.0;`
☐	☐	`boolean t = (3.14 == 3);`
☐	☐	`float[] f = new int[10];`
☐	☐	`int i, j, k, l, m=0;`
☐	☐	`long heltall = 10/2;`
☐	☐	`int[3] tre = {1, 2, 3};`
☐	☐	`String false = "" + true;`
☐	☐	`double pi = (double) ("3" + ".14");`
☐	☐	`boolean a, b, c==(1==3/2);`
☐	☐	`int[] heletall = {(int) 1.25, (int)2.5, (int)3.75};`

Oppgave 3 (16 poeng)

a) Hvor mange ganger blir "INF1000" skrevet ut av følgende løkke:

```
for (int i=2; i<6; i+=2) {
  System.out.println("INF1000");
}
```

Svar: ..... ganger

b) Hvor mange ganger blir "INF1000" skrevet ut av følgende løkke:

```
for (int j=1; j < 10 ; j = j+1 ) {
  for (int i = j-2; ++i < j++; j = i+1)
    System.out.println("INF1000");
}
```

Svar: ..... ganger

c) Hvor mange ganger blir "INF1000" skrevet ut av følgende løkker:

```
for (int i=0; i < 2; i++){
  for (int j=1; j < 3; j++){
    for(int k=0; k < 4; k++){
      System.out.println("INF1000");
    }}}
```

Svar: ..... ganger

d) Hvor mange ganger blir "INF1000" skrevet ut av følgende løkke:

```
int k = 10;
  while (k-- > 10 - k)
    System.out.println("INF1000 ");
```

Svar: ..... ganger

Oppgave 4 (4 poeng)

Anta at følgende kodelinjer utføres:

```
int tre = 3, to = 2;
double trehalve = tre/to;
int hvaerjeg = ++to + to++;
if(trehalve-1>0) hvaerjeg+=7;
hvaerjeg += hvaerjeg++ + 3;
```

Hva er verdien til variabelen **`hvaerjeg`** rett etterpå?

Svar: .....

Oppgave 5 (20 poeng)

Skriv ferdig metoden under, som beregner og returnerer volumet *V* av en kasse i ***kubikktommer*** ut fra kassens lengde, bredde og høyde i ***centimeter***. Formelen du skal bruke er: V=l*b*h, og hvor l er lengden, b er bredden og h er høyden til kassa. Dersom en av de tre variablene (l, b, h) er mindre eller lik 0 skal metoden returnere verdien 0.0. I tillegg trenger du å vite at 1 tomme er 2.54 centimeter.

Svar:

```
double kassevolumIKubikktommer(int lengde, int bredde, int hoyde){




}
```

Oppgave 6 (20 poeng)

Skriv ferdig metoden under, som med utgangspunkt i en array med heltall (heltall) og en øvre grense (grense), finner alle de verdiene i arrayen heltall som er ***mindre eller lik*** grensen og returnere en ny array med alle de verdiene (men ikke de som er over grensen). Dersom metoden blir kalt, for eksempel slik

```
plukkUtUnderGrensen(new int[]{9, 1, 7, 8, 2, 3}, 7);
```

så skal resultatet være en ny array som inneholder tallene: `{1, 7, 2, 3}`
Svar:

```
int[] plukkUtUnderGrensen ( int [] heltall, int grense ){




}
```

Oppgave 7 (10 poeng)

Anta at følgende program kjøres:

```
class Tall{
    public static void skriv(String tall){
        int i=0;
        while(tall.charAt(i++) == '0'){
            tall = tall.substring(i, tall.length());
        }
        System.out.print (tall);
        System.out.print (", ");
    }
    public static void main(String[] args){
        String stor  = "0110110";
        String liten = "0011001";
        skriv(stor);
        skriv(liten);
    }
}
```

Hva skriver programmet ut?

Svar: .....

Oppgave 8 (15 poeng)

Anta at vi har denne metoden:

```
public int beregn(boolean[] tall){
   int verdi=0;
   for(int i=0; i<tall.length; i++){
      if(tall[i]){
         int x = 1;
         for(int j=i;j<tall.length-1; j++){
            x *= 2;
         }
         verdi+=x;
      }
   }
   return verdi;
}
```

Anta videre at vi har arrayen:

```
boolean[] tall = new boolean[]{true, false, true, true };
```

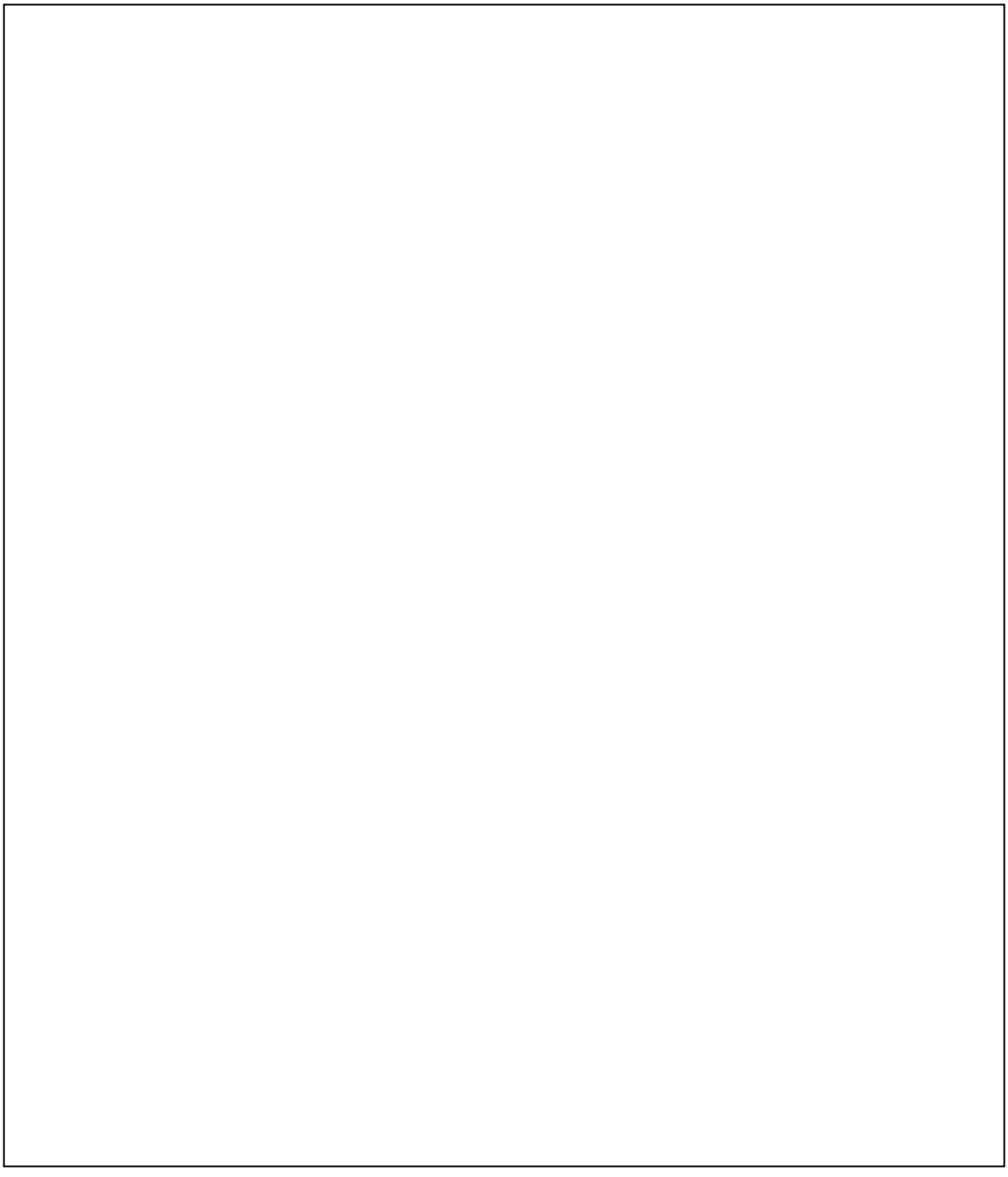
Hva returneres fra metodekallet `beregn(tall)`?

Svar: .....

Oppgave 9 (20 poeng)

Under ’Formel 1’-løp lagrer lagene enormt mange parametere fra bilene og det som hender rundt bilene. Det måles og lagres temperaturer, omdreininger, osv for bilen, lufttrykk for dekkene, puls, for føreren, hvor lang tid laget bruker på å skifte hjul, og mye annet. Du skal hjelpe et ’Formel 1’-lag med å lage et dataprogram som skal analysere disse dataene og dere skal starte med å tegne et UML-klassediagram for lagets datasystem. Systemet er organisert rundt en bil. En bil har fire hjul og naturligvis bare én fører. Motoren behøver ikke ha noen sensorer, men kan ha mange. Når bilen kommer inn til vedlikehold (pit stop) er det fra 4 til 10 personer som overhaler bilen. Tegn et UML-klassediagram med de 7 (Java-)klassene som kan brukes til å representere dette problemet. Gi navn både på disse klassene og på relasjonene mellom dem og plasser antall på forholdet mellom disse klassene.

Svar:



Oppgave 10 (60 poeng)

Gunnar Grei driver DVDsjappa AS hvor han leier ut DVDer. Nå vil Gunnar ha et enkelt datasystem for utleie, og du skal hjelpe han å lage deler av dette. Han har to filer med samme format over filmene, ”Ledig.txt” og ”Utleid.txt” over henholdsvis de ledige og de utleide filmene.

Formatet på filen er slik at det er én linje per film med flg. data for hver film:

<nr> <Utleid til/ledig> <språk> <type film> <Tittel mm>

Hvor:

```
<nr>           er en fortløpende nummerering av hver DVD-film Gunnar har
<utleid til>   inneholder enten ordet LEDIG eller e-postadressen
               til den som har leid denne DVDen,
<språk>        ”Engelsk”, ”Norsk”, ”Svensk”, ”Dansk” eller ”Urdu”
<type film>    ”Drama”, ”Action”, ”Komedie”, ”Erotisk” eller ”Barnefilm”
<tittel mm>    filmtittelen etterfulgt av regissør eller de viktigste
               skuespillerne.
```

Her er eksempel på to filmer, ”Løver for lam” (som ikke er utleid) og ”Du levande”, utleid til arnem@ifi.uio.no (den første er da på ”Ledig.txt” den andre er på : ”Utleid.txt”):

```
4101   LEDIG   Engelsk  Drama  Løver for lam, med  Meril Streep og Tom Cruise
4114   arnem@ifi.uio.no   Svensk  Drama   Du levande, regi: Roy Anderson
```

Oppgaven din skal være å lage utlånsdelen av systemet. Først leses filen ”Ledig.txt” inn og så spørres neste bruker om e-postadresse, om hvilke type film de ønsker og om de ønsker noe spesielt språk (eller om det er likegyldig). Deretter skal systemet gå gjennom de filmene fra ”Ledige.txt” ***som passer til brukerens ønske*** (type film + eventuelt språk) og for hver av disse ***som tilfredstiller brukerens krav*** spørre om brukeren ønsker å leie denne eller ikke. På det spørsmålet kan brukeren gi tre svar: ”ja”, ”nei” eller ”ferdig” (hvor ”ferdig” betyr at brukeren nå ikke ønsker få presentert flere valg, men vil leie de han/hun eventuelt hittil har valgt.)

Systemet skal så skrive ut tre filer: ”Bestilling.txt”, med e-postadressen til brukeren etterfulgt av numrene til de filmene vedkommende nå har leid. ”Ledige.txt” oppdateres med å skrives ut på nytt med de leide filmene fjernet (men med de øvrige filmene som ikke er utleid) og ”Utleid.txt” med de tidligere utleide filmene og i tillegg med de filmene som nå leies og med e-postadressen til brukeren i ”utleid til”-feltet. (Vi ser at dette systemet mangler et program for retur av DVDer, men det skal du ***ikke*** skrive nå).

Vi setter få krav til løsningen din foruten at den minst skal ha to klasser: **`DVDSystem og BrukerModul`** (men sannsynligvis flere). Konstruktøren i **`BrukerModul`** kalles med ”Ledige,txt” og ”Utleid.txt” som parametere.

Svar:

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

class DVDSystem {
    public static void main(String [] args) {
        BrukerModul b = new BrukerModul(”Ledige.txt”, ”Utleid.txt”);
    }
}

class BrukerModul {


   BrukerModul (String ledige, String utleid) {
```

Oppgave 11 (15 poeng) Litt vanskelig

Anta at følgende program utføres:

```
class Liste {
    Liste neste;
    String tekst;

    Liste(Liste neste, String tekst){
        this.neste = neste;
        this.tekst = tekst;
    }

    void skrivUt(){
        if(neste != null) neste.skrivUt();
        System.out.print(tekst + " ");
    }

    public static void main(String args[]){
       Liste l = new Liste(
         new Liste(new Liste(new Liste(null, "A"), "B"), "C"), "D");

        l.skrivUt();
    }
}
```

Tegn først opp for deg selv og finn ut hvor mange objekter av klassen **`Liste`** vi får laget fra de ulike setningene i main og hvilke verdier klassevariablene til disse har. Svar så på spørsmålet; hva skriver programmet ut på skjermen?

Svar : .....

Oppgave 12 (30 poeng)

I denne oppgaven skal du bidra til å lage en versjon av matematikeren John Horton Conways ”Game of Life”. ”Game of Life” er et eksempel på en simulasjon og man kan for eksempel se på det som en simulering av hvordan en populasjon med dyr eller bakterier utvikler seg over flere generasjoner. ”Spillet” starter med et rutenett hvor noen ruter (som vi kaller celler) er i live (Vi må merke disse på en eller annen måte), og noen er døde. ”Spillet” har ingen spillere og det er mulig å la spillet gå i uendelig mange runder. For hver runde bestemmes hva som skal gjøres med cellene ut ifra 4 regler. Det kan bli flere eller færre levende celler fra en runde til en annen. Man kan starte med en utgangsposisjon hvor for eksempel annenhver celle lever, eller hvor man har en ”klynge” med levende celler i et hjørne og se hvordan det utvikler seg. Hver celle har inntil 8 naboceller, de rett ved siden av på 4 sider, samt de fire inntill ruten diagonalt. Cellene inntil kanten av brettet har naturligvis færre naboer. Reglene for hva som skjer med rutene i rutenettet er som følger:

- En levende celle med færre enn to levende naboer dør av *ensomhet*
- En levende celle med flere enn tre levende naboer dør av *trengsel*
- En levende celle med to eller tre levende naboer forblir uendret.
- En død celle med nøyaktig tre levende naboer kommer til live

Vi kan løse dette problemet med en todimensjonal matrise av typen boolean, hvor en død celle representeres med verdien false og en levende celle med verdien true. Programmet går i en hovedløkke hvor den først tegner (skriver ut) brettet på skjermen slik det ser ut nå, så lager

brettet for neste runde ut fra de fire reglene og til slutt i løkken venter programmet et par sekunder (så man får tid til å se brettet før det blir tegnet over). Du skal skrive de to metodene **`lagNesteRunde`** og **`antallNaboer`** som til sammen skal lage neste runde. Du skal skrive **`lagNesteRunde`** slik at den kaller **`antallNaboer`**. **`lagNesteRunde`** skal lage en ny array med brettet for neste runde basert på denne rundens brett, de fire reglene og antallet naboer for de enkelte cellene, som den får fra å kalle metoden **`antallNaboer`** for cellen. Husk at du ikke vet størrelsen på arrayen før funksjonen blir kalt. Du skal ***ikke*** skrive funksjonene som tegner ut brettet (**`skriv`**) eller som venter (**`vent`**).

To hint: Du trenger to arrayer i **`lagNesteRunde`**. Og, pass på kantene av rutenettet i **`antallNaboer`**.

Vi demonstrerer ”Game of Life” på gjennomgangen i kveld!
Svar:

```
class GameOfLife{
  boolean celler[][];
  GameOfLife(boolean[][] celler){
    this.celler = celler;
  }
  void spill(){
    while(true){
      skriv(celler);
      celler = lagNesteRunde(celler);
      vent(1000);
      }
  }
  boolean[][] lagNesteRunde(boolean[][] celler){
    // Denne skal skrives!









    }
```

```
int antallNaboer(int x, int y, boolean[][] celler){
  // Og denne skal skrives!









}
```

Oppgave 13 (20 poeng)

Gunnar Grei kjent fra oppgave 8 har nå en lys ide. Hver måned får han en liste over de nye DVDene til DVDsjappa AS, med angivelse av type film, språk, regissør og skuespillere. Hvis han nå samlet opp de bestillingsfilene han lager, kunne han finne ut hva kundene hans var interessert i, og sende en e-post til dem når det kom en ny film av en type de minst hadde leid før. Hadde f.eks Siri Snill leid nesten bare barnefilmer ville hun hver måned få oversikt om de nye barnefilmene, mens Karsten Steroid, som mest leide erotiske filmer, ville få e-post om månedens nye DVDer av det slaget.

Du skal nå **ikke** lage systemet, men vurdere Gunnars lyse ide etter "Lov om persondata" og begrunne dine kommentarer til om dette er tillatt ved både å vise til konkrete paragrafer og skrive din egen vurdering om hvorfor disse paragrafene eventuelt kommer til anvendelse.

Svar: