BOKMÅL

UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i :	INF1000 — Grunnkurs i objektorientert programmering
Eksamensdag :	Fredag 2. desember 2005
Tid for eksamen :	14.30 – 17.30
Oppgavesettet er på :	13 sider
Vedlegg :	Ingen
Tillatte hjelpemidler :	Alle trykte og skrevne

- 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 304, slik at f.eks 5 poeng tilsvarer omlag 3 minutter og 8 poeng omlag 5 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 (8 poeng)

a) Hvor mange intverdier er det plass til i arrayen tabell?

```
int[][] tabell = new int[11][3];
```

Svar: .....

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

```
for (int j=2; j < 10; j=j+2)
  System.out.println("INF1000");
```

Svar: .....

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

```
for (int j= 33; j > 1; j -=8)
  System.out.println("INF1000");
```

Svar: .....

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

```
for (int j=0; j <10 ; j++){
    for (int i=1; i< 10 ; i++)
      System.out.println("INF1000");
}
```

Svar: .....

Oppgave 2 (16 poeng)

Er disse programsetningene lovlige i Java?

JA	NEI	
☐	☐	`int i=1, j =i;`
☐	☐	`int[] x = new int[];`
☐	☐	`int[] x = new int[0];`
☐	☐	`int i = new int [7];`
☐	☐	`double int = 14;`
☐	☐	`double[] x = new double x[5];`
☐	☐	`long [] x = new int[33];`
☐	☐	`long [] a, b = new long[4];`

Oppgave 3 (5 poeng)

Anta at følgende kodelinjer utføres:

```
int  a = 25;
long c = 31;
a =  25 + (int)c;
c = (long) a + 31 ;
if(c == a) System.out.prinln("Like");
else       System.out.prinln("Forskjellige");
```

Hva skrives ut?

Svar: .....

Oppgave 4 (15 poeng)

Skriv ferdig metoden under, som med utgangspunkt i om en matvare er servert eller ikke (angitt med den boolske variabelen 'servert'), regner ut matprisen med merverdiavgift (Moms). Moms-satsen for mat som serveres er 25 %, men bare 11% for mat som ikke serveres

Svar:

```
double matPrisMedMoms(double matPrisUtenMoms, boolean servert) {









}
```

Oppgave 5 (25 poeng)

Anta at følgende array og variabel er deklarert i en metode i et program:

```
int[] c = new int[57];
double snitt s = 0.0;
```

Anta videre at arrayen **c** er fylt opp med ulike ikke-negative verdier, Vil alternativene under alltid gi som resultat at variabelen **snitt** inneholder gjennomsnittet av verdiene i c (regnet som et flyttall)?

JA **NEI**

☐ ☐
```
int k = 0;
while (k < c.length ) {
   snitt = c[k]/(double)c.length;
}
```

☐ ☐
```
int k = 0;
 snitt = c[0];
while (k < c.length ) {
   c[k++] += snitt;
}
snitt = snitt/c.length;
```

☐ ☐
```
int k = 0;
while (k++ < c.length ) {
   snitt = snitt + c[k-1];
}
 snitt = snitt/(k-1);
```

☐ ☐
```
int k = 0;
while (k < c.length ) {
   snitt += c[k++]/c.length;
}
```

☐ ☐
```
int k = 0;
while (k++ < c.length ) {
   snitt += c[k-1]/(double)c.length;
}
```

Oppgave 6 (15 poeng)

Anta at følgende setninger utføres:

```
int x = 1024, lnx= 0;
while (x > 1){
   x = x/2;
   lnx = lnx+1;
}
```

Hva er verdien til variabelen **lnx** like etter at setningene over er utført?

Svar: .....

Oppgave 7 (15 poeng)

Her skal du både skrive en metode som finner det største av tre tall: a, b og c og returnerer verdien av det. Du kan ikke anta at tallene har ulike verdier..

Svar:

```
int størst(int a, int b, int c) {









}
```

Oppgave 8 (25 poeng)

To personer skal gifte seg/inngå partnerskap, og vil i den anledning lag nye etternavn som er en sammensetning av de to etternavnene med bindestrek mellom, men selvsagt vil de beholde sine gamle fornavn. Du skal skrive en metode **`lagNyeNavn`**, som mottar de to opprinnelige navnene i en String-array som parameter, og returnerer en peker til en array hvor de nye navnene er satt inn, For enkelhets skyld skal du anta at hver av navnene i utgangspunktet består av bare ett fornavn og ett etternavn adskilt med en blank. Det er navnet i posisjon 0 i arrayen som skal komme sist i det nye felles etternavnet. Eks.: Hvis metoden blir kalt med navnene ”Sara Bakken” og ”Per Olsen”, skal arrayen som returneres innholde: ”Sara Olsen-Bakken” og ”Per Olsen-Bakken”).

(Hint: Du kan f.eks. løse denne oppgaven med. å bruke String-metoden split)

Svar:

```
String [] lagNyeNavn(String [] navn) {




}
```

Oppgave 9 (30 poeng)

På en fil **"INF1000-eksamen.txt"** ligger eksamenskarakterene for INF1000 med en linje for hver kandidat. En slik linje inneholder først et tall for kandidatnummeret og så, adskilt med en blank, karakteren for vedkommende som en bokstav (A-F).

Nedenfor ser du omrisset av et program som du skal fylle ut med koden til konstruktøren til **Karakter** og metoden skriv **prosentKarakterfordeling**. Konstruktøren skal lese inn data fra filen den får som parameter og notere hvor mange det er av hver karakter i arrayen **karakterAntall**. Metoden **prosentKarakterfordeling** skal først skrive ut hvor mange som det var sensur for på fila, så hvor mange som strøk (karakteren F) og deretter hvor mange % som fikk hver av de andre karakterene A-E. Husk at disse %-ene bare skal regnes av det antallet som stod til eksamen.

Svar:

```
import easyIO.*;

class KarakterStatistikk {
    public static void main(String [] args) {
         Karakter ks = new Karakter("INF1000-eksamen.txt");
        ks.prosentKarakterfordeling();
    }
}

class Karakter {
      int []karakterAntall = new int[6];

      Karakter (String fil) {









       }


      void prosentKarakterfordeling () {
```

```
  }
}
```

Oppgave 10 (30 poeng) - vanskelig

Du skal skrive en sorteringsmetode for en heltallsarray **a**, som skal bruke følgene sorteingsprinsipp: Du finner først det minste element i **a** og plasserer det deretter i **a[0]** ved å la det elementet som opprinnelig stod på plass 0 og det minste elementet bytte plass. Så skal du finne det nest minste elementet i **a** ved å starte letingen fra indeks 1 i **a** – dvs. lete i **a[1], a[2],..,a[a.length-1].** Du bytter så om dette nest minste og det som opprinnelig stod i plass **a[1]**. Slik fortsetter du med å starte stadig høyere opp i a for å lete etter det minste elementet som nå er igjen og la det bytte plass med der du starter å lete. Slik fortsetter du til du har plassert det nest største elementet i plass **a[a.length -2]** , og du er da ferdig med å sortere arrayen.

(*Hint*: Du får to for-løkker inne i hverandre, og den ytterste angir hvor du starter å lete i **a** etter det minste elementet som er igjen i resten av arrayen)

N.B. Du får ingen poeng for å skrive av sorteringsmetoden i læreboka eller fra prøveeksamen..

```
void sorter (int[] a) {









  }
```

Oppgave 11 (25 poeng)

I programmet nedenfor skal du lage en konstruktør til klassen Pyramide som har pyramidens bredde, lengde og høyde som tre double-parametreparameter. Du skal også lage en objektmetode i klassen Pyramide som regner ut volumet og returnerer denne verdien (du skal bruke formelen: volum = 0.333*høyde*bredde*lengde). Du skal også skrive programkode i main som oppretter to pyramider, en med langde =30.1,bredde=30.1 og høyde = 22.9 og en med samme grunnflate, men med dobbelt så stor høyde som den første pyramiden. Fra main skal du så kalle på volumberegningsmetoden i hvert av de to objektene og skrive ut en linje for hver pyramide med høyde, lengde og bredde samt volumet.

```
class PyramideTest {

    public static void main ( String [] args) {
         // skriv kode her som lager to Pyramide-objekter og
         // skriver ut deres høyde, lengde og volum









   } // end main
 } // end class PyramideTest

 class Pyramide {
     double høyde, lengde, bredde;

    // skriv konstruktør her




   // skriv objektmetode her som beregner og returnerer volumet




} // end class Pyramide
```

Oppgave 12 (20 poeng)

I et supermarked er det mange gjenstander (kalt varer), og hver enkelt vare kan bare kjøpes av en kunde. En kunde kan kjøpe flere varer, og hver enkelt vare er selvsagt bare til salgs i ett supermarked. Tegn et UML-klassediagram med de 3 (Java-)klassene som kan brukes til å representere dette problemet. Gi navn på disse klassene og plasser antall på forholdet mellom disse klassene.

Svar:

Oppgave 13 (25 poeng) – ekstra vanskelig

Anta at følgende program utføres:

```
class Familie {
   Familie sønn, datter;
  int avkom;

   Familie (int avkom) {
      this.avkom = avkom;
     if (avkom > 0) datter = new Familie(avkom-1);
     if (avkom > 1) sønn  = new Familie(datter.avkom -1);
      System.out.println("Antall:" + avkom);
   }

   public static void main(String[] args) {
      Familie stammor = new Familie(3);
   }
}
```

Tegn først opp for deg selv og finn ut hvor mange Familie-objekter vi får laget av setningen i main. Svar så på spørsmålet: hvilke linjer skriver programmet ut på skjermen?

Svar:

Oppgave 14 (50 poeng)

Anta at du fra de enkelte kassaapparatene i et supermarked har samlet på en fil opplysninger om alle salg for i dag. Denne fila: **`Salg.txt`** har en linje per salg på følgende form (du kan regne med at alle varenavnene er ulike):

```
<varenavn>  <antall  solgt>  <pris i kr. per stykk>
```

Tre eksempler på slike linjer kan være

```
Lano  2  12.50
Helkornbrød  4  14.00
Lano  1  12.50
```

Du kan anta at varenavnene er unike (dvs. at det ikke er to ulike produkter som begge heter Lano) og at pris og varenavn samsvarer på alle linjer (dvs. Lano koster 12.50 for alle varesalg).
Skriv et komplett program som leser inn fila **`Salg.txt`**, oppretter et objekt for hver vare og lagrer det i en passene datastruktur. Deretter skal programmet beregne og skrive ut hvor stor dagens salg totalt er i kroner, samt varenavnene til de to varene som har solgt best i antall og i kroner. Til sist skal detskrives ut en annen fil: **`Antallsolgt.txt,`** som for hver vare har en linje med varenavn og antall som er solgt av denne varen i dag totalt.
Svar:

(fortsetter på neste side)