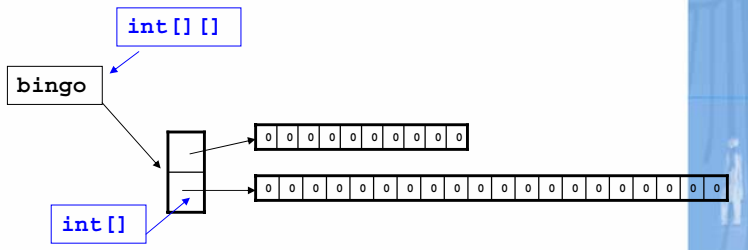
INF1000 – Løsningsforslag prøveeksamen høst 2007

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];
```



float og long i oppgave 2:

- De kom med ved et uhell.
  - Beklager!
- På eksamen vil vi bare bruke double og int

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: 2 ganger 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: 5 ganger ganger.

Oppgave 3 forts …

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

```
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: 16 ganger ganger.

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

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

Svar: 5 ganger 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;
```

tre	to	trehalve	hvaerjeg
3	2	-	-
3	2	1.0	-
3	4	1.0	6
3	4	1.0	6
3	4	1.0	15

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

Svar: 15

Oppgave 5 (5 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.

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

    if(lengde<=0 || bredde <=0 || hoyde<=0)
        return 0.0;



    return lengde*bredde*hoyde/2.54/2.54/2.54;


}
```


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}`


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

  int[] res = null;
  int ant = 0;

  for(int i=0;i<heltall.length;i++){
     if(heltall[i]<=grense)ant++;
  }

  res = new int[ant];
  ant = 0;

  for(int i=0;i<heltall.length;i++){
     if(heltall[i]<=grense)
        res[ant++] = heltall[i];
  }

  return res;
}
```


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);
    }
}
```

```
                    i      tall
1. kall     -   0110110
            0    110110
2. kall     -   0011001
            0    011001
```

Hva skriver programmet ut?

110110, 011001

Svar: .....

Oppgave 8 (15 poeng)

UNIVERSITETET I OSLO

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;
}
```

i	j	x	verdi
0	-	1	0
0	0	2	0
0	1	4	0
0	2	8	0
0	-	8	8
1	-	1	8
2	-	1	8
2	2	2	8
2	-	2	10
3	-	1	10
3	-	1	11

Anta videre at vi har arrayen:

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

Hva returneres fra metodekallet `beregn(tall)`?

Svar: 11

Oppgave 9 (20 poeng)

UNIVERSITETET I OSLO

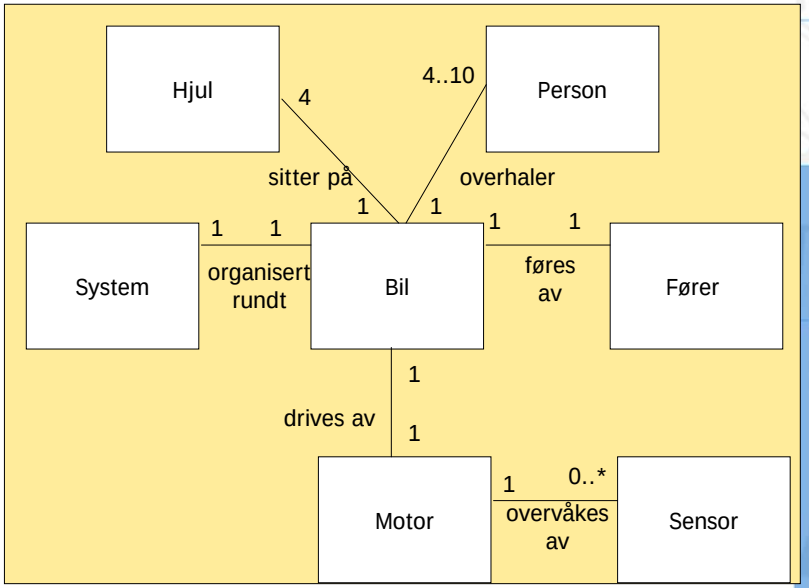
Under 'Formel 1'-***løp*** lagrer lagene enormt mange parametere fra bilene og det som hender rundt bilene. Det måles og lagres temperaturer, omdreininger, osv for bilen, lufttrykk for dekkene, puls, for føreren, hvor lang tid laget bruker på å skifte hjul, og mye annet.

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.

UNIVERSITETET I OSLO

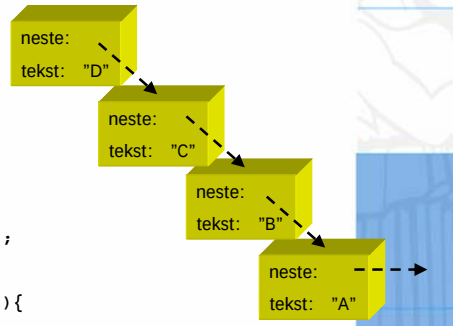


Oppgave 11 (15 poeng)

UNIVERSITETET I OSLO

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 : A B C D

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.

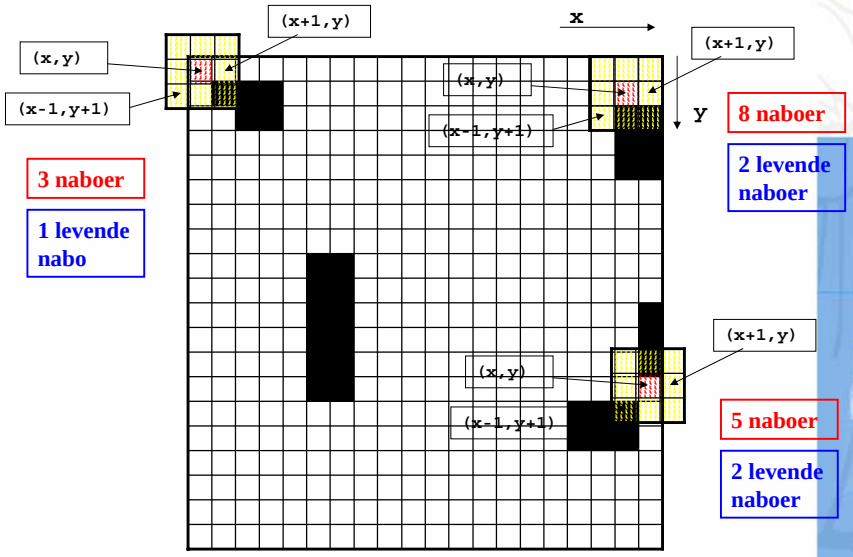
Man kan starte med en utgangsposisjon hvor for eksempel annenhver celle lever, eller hvor man har en "klynge" med levende celler i et hjørne og se hvordan det utvikler seg.

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:

Reglene

- 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***

Rutenettet



Skallet

```
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){
      // Skulle skrive denne ...

    }

    int antallNaboer(int x, int y, boolean[][] celler){
      // Og denne ...

    }
}
```

Lage runde

UNIVERSITETET I OSLO

```
boolean[][] lagNesteRunde(boolean[][] celler){
    boolean[][] nyeCeller =
        new boolean[celler.length][celler[0].length];

    for(int x=0;x<celler.length; x++){
        for(int y=0;y<celler[x].length; y++){

            int ant = antallNaboer(x, y, celler);

            if(celler[x][y]){
                if(ant<2) nyeCeller[x][y] = false;
                else if(ant>3) nyeCeller[x][y] = false;
                else nyeCeller[x][y] = true;
            } else {
                if(ant==3) nyeCeller[x][y] = true;
            }
        }
    }
    return nyeCeller;
}
```


Antall naboer

UNIVERSITETET I OSLO

```
int antallNaboer(int x, int y, boolean[][] celler){
   int ant = 0;
   if(x>=1 && y>=1)
       if(celler[x-1][y-1]) ant++;
   if(x>=1)
       if(celler[x-1][y]) ant++;
   if(x>=1 && y<celler.length-1)
       if(celler[x-1][y+1]) ant++;
   if(y>=1)
       if(celler[x][y-1]) ant++;
   if(y<celler.length-1)
       if(celler[x][y+1]) ant++;
   if(x<celler.length-1 && y>=1)
       if(celler[x+1][y-1]) ant++;
   if(x<celler.length-1)
       if(celler[x+1][y]) ant++;
   if(x<celler.length-1 && y<celler.length-1)
       if(celler[x+1][y+1]) ant++;
   return ant;
}
```

Institutt for Informatikk Fredrik Sørensen og Arne Maus 22. november 2007