# Avdeling for allmenne fag

**EKSAMEN**

**852 OBJEKTORIENTERT PROGRAMMERING**

**10.12.2002**

| | |
|---|---|
| Tid: | 9.00 – 13.00 (14.00)<br>4 timar for dei som har tatt midtsemesterprøve.<br>5 timar for dei som ikkje har tatt midtsemesterprøve. |
| Målform: | Nynorsk/bokmål |
| Sidetal: | 3+3 + framside |
| Hjelpemiddel: | Alle trykte og skrevne |
| Merknader: | Berre dei som ikkje har tatt midtsemesterprøve skal gjere oppgåve 5. Oppgåve 1-4 utgjer 80% av oppgåvesettet. |
| Vedlegg: | Ingen |

**Eksamensresultata vert offentleggjort på Arena høgskole. Passordet til Arena har alle studentar fått i eige brev ved semesterstart. Du går inn på Arena høgskole gjennom HiTs nettsider.**

## Eksamen 852 Objektorientert Programmering 10.12.2002 - Nynorsk

**Råd og retningsliner.** Les oppgåveteksten godt fleire gonger før du går i gang med å løyse oppgåva. Deloppgåvene er uavhengige av kvarandre i den meining at om du ikkje får til éi oppgåve, kan du likevel gjere neste, som om den fyrste var løyst. Fordél tida godt på alle oppgåvene. Om du meiner ei oppgåve er upresis, så skriv din eigen presisering.

## Oppgåve 1

Det skal utviklast eit animasjonsprogram basert på fylgjande kode

```
class Ball extends Thread {
  static Color farge = Color.blue;    // Fargen til alle ballar
  protected int x=0;                   / posisjon til ballens senter
  protected int y=0;
  protected int dx=0;                 // fart i x- og y-retning
  protected int dy=0;
  protected int diameter;             // ballens diameter
  protected Teikneflate teikneflate;  / t.d. noko canvas-aktig

  public Ball() {}
  public Ball(int nyX, int nyY, int nyDx, int nyDy,
    int nyDiameter, Teikneflate nyTeikneflate) {
    setX(nyX);
    setY(nyY);
    setDx(nyDx);
    setDy(nyDy);
    setDiameter(nyDiameter);
    setTeikneflate(nyTeikneflate);
  }

  public void setX(int nyX) {...}
  public void setY(int nyY) {...}
  public void setDx(int nyDx) {...}
  public void setDy(int nyDy) {...}
  public void setDiameter(int nyDiameter) {...}
  public void setTeikneflate(Teikneflate nyTeikneflate)
  public int getX() {...}
  public int getY() {...}
  protected boolean fortsett() {...}
  protected void viskut() {...}
  protected void flytt() {...}
  protected void kollisjon() {...}
  protected void teikn() {...}
  protected void vent() {...}
  public void run() {
    while (fortsett()) {
      viskut();
      flytt();
      kollisjon();
      teikn();
      vent();



interface Teikneflate {
  public Graphics getGrafikk();
  public Dimension getDimensjon
  public Color getBakgrunn();
}
```

## 1a (10%)

Lag ferdig alle set... og get... metodane i class Ball. Dersom ein metode får inn ein heilt opplagt gal parameterverdi skal du kaste eit unntak.

## 1b (15%)

Lag fylgjande seks metodar:

1) fortsett-metoden skal lagast slik at run-metoden går evig.
2) viskut-metoden skal teikne ballen med teikneflata si bakgrunnsfarge.
3) flytt-metoden skal flytte ballen i samsvar med dx og dy verdiane.
4) kollisjons-metoden skal sjå til at ballen sprett når den treff veggane i teikneflata.
5) teikn-metoden skal teikne ballen med ballanes farge.
6) vent-metoden skal lagast slik at run-metoden teiknar ca 25 bilete i sekundet.

## Oppgåve 2

Ball-objekt går i rette liner, men her skal vi sjå på effekten av gravitasjon.

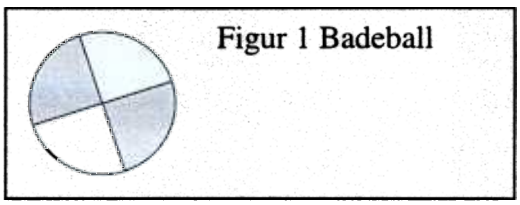
## 2a (10%)

Lag ein class GBall som subklasse av Ball slik at den er utsett for gravitasjon nedover. Legg inn ein ekstra variabel som beskriv kor kraftig gravitasjonen er, alle GBall'ar er utsett for ***like kraftig*** gravitasjon. Lag ei setGravitasjon-metode for å sette ny verdi på denne variablen. Lag parameterlaus konstruktørmetode, og konstruktørmetode med 6 parametre. Reimplementer flytt-metoden slik at dy endrast med verdien av gravitasjonen før ballen flyttast på vanleg vis. Hugs å nytte mest mogeleg av den koden som er i superklassa.

## 2b (10%)

Vi skal kunne stoppe ballen. Legg inn ein ekstra instansvariabel i class GBall, lag ein public parameterlaus metode "stopp", og reimplementer fortsett-metoden slik at ballen stoppar når stopp-metoden kallast.

## Oppgåve 3 (20%)


Figur 1 Badeball

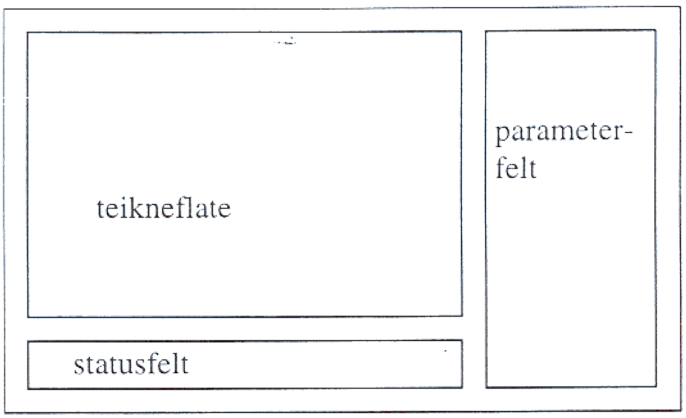
Det skal lagast ein class Badeball som subklasse av GBall. Ein badeball er laga av fire bitar og kan sjå ut som i figur 1. Denne er eigentleg gul og raud, men ser lys og mørk grå ut her.
Generelt kan kvar badeball ha sine to fargar.
Badeballar kan rotere medan dei svever gjennom lufta, kvar badeball roterar med sin rotasjonshastighet, målt i omdreiningar pr. sekund. Det vanlegaste er å ha mindre enn ein omdreining pr. sekund. class Badeball treng altså fire nye instansvariable: to fargar, rotasjonshastighet, og rotasjons"posisjon".

Lag class Badeball, med sine nye instansvaraible, set-metodar for fargane og rotasjonshastighet (treng ikkje get-metodar), parameterlaus konstruktørmetode, konstruktørmetode med 9 parametre (6 "gamle" + 2 fargar og rot.hast.), reimplementert flytt-metode og teikn-metode.

## Oppgåve 4 (15%)

Det skal utviklast ein enkel testapplikasjon for å sjå om dei tre ball-typane verker som dei skal. Grensesnittet skal sjå ut omtrent som vist i figur 2.

Figur 2 Brukargrensesnittet




All teikning skal skje i teikneflata. Statusfeltet skal vise kor mange Ballar, GBallar og Badeballar som er oppretta, og parameterfeltet skal gje brukar høve til å opprette nye Ballar, GBallar og Badeballar, bestemme parametre for desse (startposisjon, starthastighet, rotasjonshastighet, fargar). Dessutan høve til å justere gravitasjonen, og ein avslutt-knapp.

Teikn detaljane (med ramme rundt panel) i parameterfeltet slik du velgjer å designe det, og teikn komponenthierarkiet for heile brukargrensesnittet, der du set på komponenttype, layoutmanager m.m.

## Oppgåve 5 – berre for dei som ikkje har tatt midtsemesterprøve

Det skal utviklast ein applikasjon for å simulere solsystemet. Planetane liknar på ballar, men må ha masse. Dei vil bli utsett for gravitasjon frå andre himmellekamar, men vi kan ikkje nytte class GBall, for der hadde alle ballar lik gravitasjon.

### 5a (10%)

Lag class Planet som subklasse av Ball. Kvar planet har sin eigen masse, og sin egen farge. Planetklassa skal ha parameterlaus konstruktør, konstruktør med 8 parametre, du må re-implementere flytt til å ta omsyn til gravitasjon, og kollisjon slik at planeten ***ikkje*** sprett når den treff ein vegg. Kvar gong planeten flyttar seg, må den spørje teikneflata kor stor gravitasjonen blir. Gå ut i frå at teikneflata har ein metode:

```
public Gravitasjon getGravitasjon(Planet planet) { ... }
```

-der planeten må sende med seg sjølv som aktualparameter. Gravitasjonen kan peike i alle retningar, så vi nyttar eit objekt av klassa Gravitasjon:

```
class Gravitasjon {
  public int gx; // gravitasjon i x-retning
  public int gy; // gravitasjon i y-retning
}
```

### 5b (10%)

Lag class Solsystem som subklasse av Canvas, slik at den implementerer Teikneflate. Den skal vidare ha plass til vilkårleg mange planetar (vi ser her på sola som ein stor planet!), parameterlaus konstruktørmetode, ein metode for å legge til planetar:

```
public void addPlanet(Planet p) {...}
```

og ein parameterlaus start-metode som skal starte alle planetane. Klassa burde også hatt getGravitasjon-metoden, men den treng du ***ikkje*** lage. Du kan få bruk for at class Canvas har metoden: `public Dimension getSize() {...}`

Lykke til!

## Eksamen 852 Objektorientert Programmering 10.12.2002 - Bokmål

**Råd og retningslinjer.** Les oppgaveteksten godt flere ganger før du går i gang med å løse oppgaven. Deloppgavene er uavhengige av hverandre i den forstand at om du ikke får til én oppgave, kan du likevel gjøre neste, som om den første var løst. Fordél tiden godt på alle oppgavene. Om du mener en oppgave er upresis, så skriv din egen presisering.

## Oppgave 1

Det skal utvikles et animasjonsprogram basert på følgende kode:

```
class Ball extends Thread {
  static Color farge = Color.blue;  // Fargen til alle baller
  protected int x=0;                 / posisjonen til ballens senter
  protected int y=0;
  protected int dx=0;                  fart i x- og y-retning
  protected int dy=0;
  protected int diameter;            // ballens diameter
  protected Tegneflate tegneflate;   // f.eks. noe canvas-aktig

  public Ball() {}
  public Ball(int nyX, int nyY, int nyDx, int nyDy,
    int nyDiameter, Tegneflate nyTegneflate) {
    setX(nyX);
    setY(nyY);
    setDx(nyDx);
    setDy(nyDy);
    setDiameter(nyDiameter);
    setTegneflate(nyTegneflate);


  public void setX(int nyX) {...}
  public void setY(int nyY) {...}
  public void setDx(int nyDx) {...}
  public void setDy(int nyDy) {...}
  public void setDiameter(int nyDiameter) {...}
  public void setTegneflate(Tegneflate nyTegneflate)     }
  public int getX() {...}
  public int getY() {...}
  protected boolean fortsett() {...}
  protected void viskut() {...}
  protected void flytt() {...}
  protected void kollisjon() {...}
  protected void tegn() {...}
  protected void vent() {...}
  public void run() {
    while (fortsett()) {
      viskut();
      flytt();
      kollisjon();
      tegn();
      vent();



interface Tegneflate {
  public Graphics getGrafikk();
  public Dimension getDimensjon
  public Color getBakgrunn();
}
```

## 1a (10%)

Lag ferdig alle set... og get... metodene i class Ball. Dersom en metode får inn en helt opplagt gal parameterverdi skal du kaste et unntak.

## 1b (15%)

Lag følgende seks metoder:

1) fortsett-metoden skal lages slik at run-metoden går evig.
2) viskut-metoden skal tegne ballen med tegneflatens bakgrunnsfarge.
3) flytt-metoden skal flytte ballen i henhold til dx og dy verdiene.
4) kollisjons-metoden skal sørge for at ballen spretter når den treffer veggene i tegneflaten.
5) tegn-metoden skal tegne ballen med ballenes farge.
6) vent-metoden skal lages slik at run-metoden tegner ca 25 bilder i sekundet.

## Oppgave 2

Ball-objekter går i rette linjer, men her skal vi se på effekten av gravitasjon.

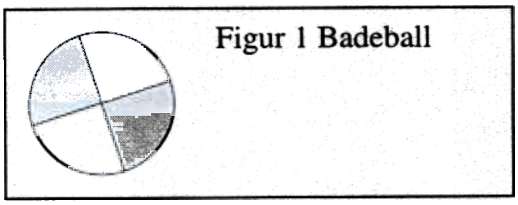
## 2a (10%)

Lag en class GBall som subklasse av Ball slik at den er utsatt for gravitasjon nedover. Legg inn en ekstra variabel som beskriver hvor kraftig gravitasjonen er, alle GBall'er er utsatt for ***like kraftig*** gravitasjon. Lag en setGravitasjon-metode for å sette ny verdi på denne variablen. Lag parameterløs konstruktørmetode, og konstruktørmetode med 6 parametre. Reimplementer flytt-metoden slik at dy endres med verdien av gravitasjonen før ballen flyttes på vanlig vis. Husk å benytte mest mulig av den koden som er i superklassen.

## 2b (10%)

Vi skal kunne stoppe ballen. Legg inn en ekstra instansvariabel i class GBall, lag en public parameterløs metode "stopp", og reimplementer fortsett-metoden slik at ballen stopper når stopp-metoden kalles.

## Oppgave 3 (20%)



Figur 1 Badeball

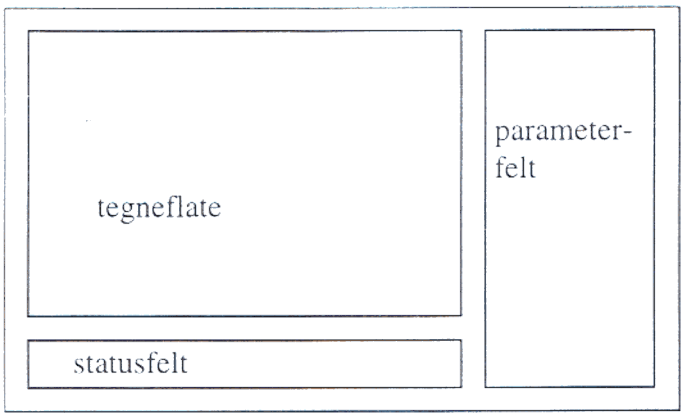
Det skal lages en class Badeball som subklasse av GBall. En badeball er laget av fire biter og kan se ut som i figur 1. Denne er egentlig gul og rød, men ser lys og mørk grå ut her.
Generelt kan hver badeball ha sine to farger.
Badeballer kan rotere mens de svever gjennom lufta, hver badeball roterer med sin rotasjonshastighet, målt i omdreininger pr. sekund. Det vanligste er å ha mindre enn én omdreining pr. sekund. class Badeball trenger altså fire nye instansvariable: to farger, rotasjonshastighet, og rotasjons"posisjon".

Lag class Badeball, med sine nye instansvaraible, set-metoder for fargene og rotasjonshastighet (trenger ikke get-metoder), parameterløs konstruktørmetode, konstruktørmetode med 9 parametre (6 "gamle" + 2 farger og rot.hast.), reimplementert flytt-metode og tegn-metode.

## Oppgave 4 (15%)

Det skal utvikles en enkel testapplikasjon for å se om de tre ball-typene virker som de skal Grensesnittet skal se ut omtrent som vist i figur 2.

Figur 2 Brukergrensesnittet




All tegning skal skje i tegneflata. Statusfeltet skal vise hvor mange Baller, GBaller og Badeballer som er oppretta, og parameterfeltet skal gi bruker mulighet for å opprette nye Baller, GBaller og Badeballer, bestemme parametre for disse (startposisjon, starthastighet, rotasjonshastighet, farger). Dessuten mulighet for å justere gravitasjonen, og en avslutt-knapp.

Tegn detaljene (med ramme rundt panel) i parameterfeltet slik du velger å designe det, og tegn komponenthierarkiet for hele brukergrensesnittet, der du setter på komponenttype, layoutmanager m.m.

## Oppgave 5 – bare for de som ikke har tatt midtsemesterprøve

Det skal utvikles en applikasjon for å simulere solsystemet. Planetene ligner på baller, men må ha masse. De vil bli utsatt for gravitasjon fra andre himmellegemer, men vi kan ikke bruke class GBall, for der hadde alle baller lik gravitasjon.

### 5a (10%)

Lag class Planet som subklasse av Ball. Hver planet har sin egen masse, og sin egen farge. Planetklassen skal ha parameterløs konstruktør, konstruktør med 8 parametre, du må re-implementere flytt til å ta hensyn til gravitasjon, og kollisjon slik at planeten ***ikke*** spretter når den treffer en vegg. Hver gang planeten flytter seg, må den spørre tegneflaten hvor stor gravitasjonen blir. Anta at tegneflaten har en metode:

```
public Gravitasjon getGravitasjon(Planet planet) { ... }
```

-der planeten må sende med seg selv som aktualparameter. Gravitasjonen kan peke i alle retninger, så vi bruker et objekt av klassen Gravitasjon:

```
class Gravitasjon {
  public int gx; // gravitasjon i x-retning
  public int gy; // gravitasjon i y-retning
```

### 5b (10%)

Lag class Solsystem som subklasse av Canvas, slik at den implementerer Tegneflate. Den skal videre ha plass til vilkårlig mange planeter (vi betrakter her sola som en stor planet!), parameterløs konstruktørmetode, en metode for å legge til planeter:

```
public void addPlanet(Planet p) {...}
```

og en parameterløs start-metode som skal starte alle planetene. Klassen burde også hatt getGravitasjon-metoden, men den behøver du ***ikke*** lage. Du kan få bruk for at class Canvas har metoden: `public Dimension getSize() {...}`

*Lykke til!*