Avdeling for allmenne fag

EKSAMEN

852 OBJEKTORIENTERT PROGRAMMERING

20.12.2003

Tid:	9.00 – 12.00 (13.00) 3 timar for ordinære studentar. 4 timar for dei som skal ha utvida prøve – ved "konting", og ved godkjend fråver frå midtprøve.
Målform:	Nynorsk/bokmål
Sidetal:	3+3
Hjelpemiddel:	Alle trykte og skrevne
Merknader:	Berre dei som skal ha utvida prøve skal gjere oppgåve 3. Oppgåve 1-2 utgjer 75% 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 20.12.2003 - Nynorsk

Råd og retningsliner. Les oppgåveteksten godt 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 (50% av heile oppgåvesettet)

Det skal utviklast eit system for NSB for å halde styr på kvar i verda dei ulike vognene er, og i kva for vogn ein finn gods som er sendt med toget. Etter grundig drøfting er dei fylgjande vogntypane identifisert, og det er ført på kva for informasjon som skal lagrast for kvar vogn.

- Sittevogn – ei vogn med vanlege sitteplassar. Informasjon: antal sitteplassar, og for kvar plass - om plassen nå er opptatt eller ikkje.
- Sovevogn – vogn med senger. Informasjon: antal senger, og for kvar seng: ledig eller opptatt.
- Tørrbulkvogn – vogn med eitt stort, lukka rom for transport av korn, flis, kol mm. Informasjon: Varetype (tekst) og antal kilo last.
- Våtbulkvogn – vogn med eitt stort lukka rom for transport av væsker. Informasjon: Varetype (tekst), antal kilo last, og om eksplosjonsfarleg eller ikkje.
- Gassvogn – vogn med eitt stort lukka rom for transport av gass under trykk. Informasjon:Varetype (tekst), antal kilo last, og trykket vogna er sertifisert for (kommatal).
- Open vogn – vogn med flatt golv, utan tak. Kan frakte ulikt stykkgods som toler vér og vind. Informasjon: liste over alt godset.
- Lukka vogn – vogn med flatt golv, med veggar og tak. Kan frakte ulikt stykkgods. Informasjon: liste over alt godset.

Alle vogner må dessutan vite kva dei veg i tom tilstand.

1a) Lag eit klassehierarki som er egna for desse vognene. Det skal inngå ei klasse for kvar vogntype i lista over, dessutan kan det vere turvande (hensiktsmessig) med ei eller fleire abstrakte klasser. Klassehierarkiet skal teiknast, dessutan skal det for kvar klasse skrivast nokre liner kode som deklarerer klassa og dei naudsynte instansvariablane, men ingen metoder. OpenVogn og LukkaVogn skal halde på (mange) objekt av class Gods. Denne klassa er laga ferdig av andre. Både i Passasjervogn, Sovevogn, OpenVogn og LukkaVogn kan du gjerne nytte ein Vector.

1b) Lag ein eller fleire metoder
public int passasjerar()
slik at alle sittevogner og sovevogner kan rapportere antal passasjerar i vogna. Fortel kva for klasse metoden(-ane) skal plasserast i.

1c) Det er laga eit grensesnitt:

```
interface HarVekt{
  public int vekt(); // Oppgir vekt i kilo
}
```

Gods-klassa implementerer HarVekt-grensesnittet:

```
class Gods implements HarVekt {
  public int vekt() { ... }        // returnerer vekta av godset
  public int getId() { ... }       // returnerer id-nummeret til godset.
  // + instansvariable og evt fleire metoder
}
```

Lag ein eller fleire metoder:
public int godsVekt()
- slik at alle vogner kan rapportere vekta av gods/passasjerar i vogna (i kilo). Fortel i kva for klasse denne/desse metodane skal plasserast. For passasjerar vert vekta estimert til 100 kg/passasjer, for å vere på den sikre sida.

1d) Alle vogner skal kunne rapportere nettovekta – kva vogna veg tom. Lag ein eller fleire metoder
public int nettoVekt() { ... }
som returnerer nettovekta, og fortel i kva for klasse den/dei skal plasserast.

1e) Alle vogner skal kunne rapportere bruttovekta – kva vogna veg inklusive gods/passasjerar. Gjer dei naudsynte endringar i klasseheadingar og/eller lag metoder slik at alle vogn-klassene implementerer HarVekt-grensesnittet, der vekt-metoden skal rapportere bruttovekta på vogna. Fortel i kva klasse metoden(-ane) skal plasserast.

Oppgåve 2 (25% av heile oppgåvesettet)

Anta at det er laga ei klasse Togsett med fylgjande metoder:

```
/**
 * returnerer godsobjekt dersom det finst
 * @param int godsId – id-nummer som identifiserer gods
 * @return Gods – gods-objektet, eller null.
**/
public Gods finnGods(int godsId)

/**
 * returnerer true dersom godsobjekt finst
 * @param int godsId – id-nummer som identifiserer gods
 * @return boolean – true dersom gods-objektet finst, false elles.
**/
public boolean harGods(int godsId)

/**
 * returnerer navnet på den staden der dette togsettet sist vart registrert
* @return String – namnet på staden
**/
public String stad()
```

Det skal lagast eit socketbasert nettverkssystem slik at kundar kan sjå kvar gods dei har sendt er no. Klientdelen spør brukaren etter godsId – eit heiltal – og sender dette (som ein Integer) til tenaren. Tenaren leiter etter dette godset og rapporterer tilbake (som ein String) kor godset sist vart sett. Tenaren leiter i ein Vector av Togsett, den inneheld alle togsett. Om godset ikkje vert funne – returner feilmelding.

Skriv tenarprogrammet. Anta at leitinga etter godset går raskt – du treng ikkje støtte for fleire klientar samstundes. Ved oppstart skal Togsett-vektoren fyllast opp med data som ligg på fila "togsett.ser" – der er heile vektoren tidlegare serialisert.

Oppgåve 3 – berre for dei med utvida sluttprøve. (25% av heile oppgåvesettet)

Lag class Togsett. Klassa skal fylgje spesifikasjonen i oppgåve 2. Den skal ha instansvariable for å halde på eitt objekt av class Lokomotiv (som du ikkje skal lage!), mange vogn-objekt, og staden togsettet sist vart registrert. Lokomotiv-klassa implementerer HarVekt-grensesnittet, det skal også Togsett-klassa gjere. Vidare skal Togsett-klassa ha desse to metodane:

public int passasjerar()	Rapporterer antal passasjerar
public void stad(String s)	Fortel togsettet at det er observert på staden s

Du treng ikkje lage konstruktørmetoder.
Du kan gjerne lage hjelpemetoder i dei klassene du måtte ynskje.

Lykke til!

Eksamen 852 Objektorientert Programmering 20.12.2003 - Bokmål

Råd og retningslinjer. Les oppgaveteksten godt før du går i gang med å løse oppgava. 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 tida godt på alle oppgavene. Om du mener en oppgave er upresis, så skriv din egen presisering.

Oppgave 1 (50% av hele oppgavesettet)

Det skal utvikles et system for NSB for å holde styr på hvor i verden de ulike vognene er, og i hvilken vogn man finner gods som er sendt med toget. Etter grundig drøfting er de følgende vogntypene identifisert, og det er ført på hvilken informasjon som skal lagres for hver vogn.

- Sittevogn – en vogn med vanlige sitteplassar. Informasjon: antall sitteplasser, og for hver plass - om plassen nå er opptatt eller ikke.
- Sovevogn – vogn med senger. Informasjon: antall senger, og for hver seng: ledig eller opptatt.
- Tørrbulkvogn – vogn med ett stort, lukket rom for transport av korn, flis, kull mm. Informasjon: Varetype (tekst) og antall kilo last.
- Våtbulkvogn – vogn med ett stort lukket rom for transport av væsker. Informasjon: Varetype (tekst), antall kilo last, og om eksplosjonsfarlig eller ikke.
- Gassvogn – vogn med ett stort lukket rom for transport av gass under trykk. Informasjon:Varetype (tekst), antall kilo last, og trykket vogna er sertifisert for (kommatall).
- Åpen vogn – vogn med flatt gulv, uten tak. Kan frakte ulikt stykkgods som toler vær og vind. Informasjon: liste over alt godset.
- Lukket vogn – vogn med flatt gulv, med vegger og tak. Kan frakte ulikt stykkgods. Informasjon: liste over alt godset.

Alle vogner må dessuten vite hva de veier i tom tilstand.

1a) Lag et klassehierarki som er egnet for disse vognene. Det skal inngå en klasse for hver vogntype i lista over, dessuten kan det være hensiktsmessig med en eller flere abstrakte klasser. Klassehierarkiet skal tegnes, dessuten skal det for hver klasse skrives noen linjer kode som deklarerer klassa og de nødvendige instansvariablene, men ingen metoder. ÅpenVogn og LukketVogn skal holde på (mange) objekter av class Gods. Denne klassa er laget ferdig av andre. Både i Passasjervogn, Sovevogn, ÅpenVogn og LukketVogn kan du gjerne bruke en Vector.

1b) Lag en eller flere metoder
public int passasjerer()
slik at alle sittevogner og sovevogner kan rapportere antall passasjerer i vogna. Fortell hvilken klasse metoden(-ene) skal plasserast i.

1c) Det er laget et grensesnitt:

```
interface HarVekt{
  public int vekt(); // Oppgir vekt i kilo
}
```

Gods-klassa implementerer HarVekt-grensesnittet:

```
class Gods implements HarVekt {
  public int vekt() { ... }        // returnerer vekta av godset
  public int getId() { ... }       // returnerer id-nummeret til godset.
  // + instansvariable og evt flere metoder
}
```

Lag en eller flere metoder:
public int godsVekt()
- slik at alle vogner kan rapportere vekta av gods/passasjerer i vogna (i kilo). Fortell i hvilken klasse denne/disse metodene skal plasseres. For passasjerer blir vekta estimert til 100 kg/passasjer, for å være på den sikre sida.

1d) Alle vogner skal kunne rapportere nettovekta – hva vogna veier tom. Lag en eller flere metoder
public int nettoVekt() { ... }
som returnerer nettovekta, og fortell i hvilken klasse den/de skal plasseres.

1e) Alle vogner skal kunne rapportere bruttovekta – hva vogna veier inklusive gods/passasjerer. Gjør de nødvendige endringer i klasseheadinger og/eller lag metoder slik at alle vogn-klassene implementerer HarVekt-grensesnittet, der vekt-metoden skal rapportere bruttovekta på vogna. Fortell i hvilken klasse metoden(-ene) skal plasseres.

Oppgave 2 (25% av hele oppgavesettet)

Anta at det er laget en klasse Togsett med følgende metoder:

```
/**
 * returnerer godsobjekt dersom det finnes
 * @param int godsId – id-nummer som identifiserer gods
 * @return Gods – gods-objektet, eller null.
**/
public Gods finnGods(int godsId)

/**
 * returnerer true dersom godsobjekt finnes
 * @param int godsId – id-nummer som identifiserer gods
 * @return boolean – true dersom gods-objektet finnes, false ellers.
**/
public boolean harGods(int godsId)

/**
 * returnerer navnet på det stedet der dette togsettet sist ble registrert
* @return String – navnet på stedet
**/
public String sted()
```

Det skal lages et socketbasert nettverkssystem slik at kunder kan se hvor gods de har sendt er nå. Klientdelen spør brukeren etter godsId – et heltall – og sender dette (som en Integer) til tjeneren. Tjeneren leter etter dette godset og rapporterer tilbake (som en String) hvor godset sist ble sett. Tjeneren leter i en Vector av Togsett, den inneholder alle togsett. Om godset ikke blir funnet – returner feilmelding.

Skriv tjenerprogrammet. Anta at leitinga etter godset går raskt – du trenger ikke støtte for flere klientar samtidig. Ved oppstart skal Togsett-vektoren fylles opp med data som ligger på fila "togsett.ser" – der er heile vektoren tidligere serialisert.

Oppgave 3 – bare for de med utvidet sluttprøve. (25% av hele oppgavesettet)

Lag class Togsett. Klassa skal følge spesifikasjonen i oppgave 2. Den skal ha instansvariable for å holde på ett objekt av class Lokomotiv (som du ikke skal lage!), mange vogn-objekter, og stedet togsettet sist ble registrert. Lokomotiv-klassa implementerer HarVekt-grensesnittet, det skal også Togsett-klassa gjøre. Videre skal Togsett-klassa ha disse to metodene:

public int passasjerer()	Rapporterer antal passasjerer
public void sted(String s)	Forteller togsettet at det er observert på stedet s

Du trenger ikke lage konstruktørmetoder.
Du kan gjerne lage hjelpemetoder i de klassene du måtte ønske.

Lykke til!