DAT102 våren 2020 - Øving 0

sist oppdatert 7.1

Repetisjon – Alle må ha gjort øvingen i løpet av uke 3.
Ingen innlevering.
OBS! Lag et nytt workspace, DAT102 for dette faget og bruk package no.hvl.dat102 i prosjektet.

Oppgave

Butikker selger vanligvis flere varer. I et program som holder orden på lagerføring fins det to sentrale klasser, en for varene og en for butikkene. I denne oppgaven skal vi lage de to klassene. Objektvariablene skal bare være synlige innenfor klassen.

Objekt av klassen **`Vare`** er kjennetegnet med et unikt varenummer (heltall), et navn (tegnstreng), en pris (flyttall) og antall som angir antall av denne varen. Objektvariablene skal bare være synlige innenfor klassen.

Deloppgavene a-e angår klassen **`Vare.`**
Les gjennom alle deloppgavene før du begynner å svare.

a) Foreslå objektvariabler (instansvariabler) for klassen.

b) Lag tre konstruktører, en uten parametre, en konstruktør med en parameter som skal være **`varenr,`** og en konstruktør med fire parametre.

c) Lag get -og set-metoder for hver objektvariabel.

d) Lag en metode **`lesVare`** som spør brukeren etter navn og pris og leser inn verdier til disse objektvariablene fra tastaturet. Ved forsøk på å tildele en negativ pris, skal brukeren bli varslet og bedt om en ny pris.

Vare-objekt må opprettes før man kaller **`lesVare`**. Se **`leggInnNyVare`** i klassen **`Butikk`** senere. Det kan gjøres på andre måter.

e) Lag en metode **`toString()`** som lager og returnerer en streng basert på objektvariablene.

Vi skal nå lage en klasse for butikker. Hver butikk er kjennetegnet av et navn og en tabell med varer.

Deloppgavene f- o angår klassen **`Butikk`**.
Les gjennom alle deloppgavene før du begynner å svare.

f) Sett opp objektvariable for klassen **`Butikk`**. De skal bare være synlige innenfor klassen. Butikk skal ha en referansetabell, **`varer`** for objekter av type **`Vare`** og en objektvariabel, **`antallTyper`** som angir antall ulike typer varer.

g) Lag en konstruktør med butikken sitt navn og maksimalt antall varer som parametre.

h) Lag en metode **`finnVare(int varenr)`** som returnerer posisjonen (indeksen) i tabellen der varen fins, eller -1 dersom varen ikke fins.

i) Lag en metode **`erLedigPlass()`** som returnerer sann dersom det er ledig plass i tabellen ellers usann.

j) Lag en metode **`leggInnNyVare(int varenr)`** som legger en ny vare til butikken. Etter innsettingen skal tabellen **`varer`** være sortert etter stigende varenr. Lagerbeholdning for denne varen er 0 (kun klargjøring av vare). Dersom varen fins fra før eller lageret er fullt, skal dette meldes til brukeren. Prøv først selv. Hvis du ikke får det til, kan du se algoritmen på slutten.

k) Lag en metode **`slettVare(int varenr)`** som sletter en vare fra butikken (varen trekkes tilbake for salg). Dersom varen ikke fins, skal dette meldes til brukeren. Tabellen **`varer`** skal være sortert etter uttak av en vare. Prøv først selv. Hvis du ikke får det til, kan du se algoritmen på slutten.

l) Lag en metode **`detaljSalg(int varenr)`** som reduserer antall på denne varen med 1. Dersom varen ikke er registrert, skal dette meldes til brukeren. Dersom det er 0 stykker igjen på lager skal dette meldes til brukeren.

m) Lag en metode **`grossInnkjop(int varenr, int ant`**) som øker antallet på denne varen med parameteren **`ant.`** Dersom varen ikke fins, eller dersom **`ant`** ikke er positiv skal dette meldes til brukeren.

n) Lag en metode **`salgsVerdi()`** som regner ut total salgsverdi av hele lageret.

o) Lag en en metode **`skrivUtVarer()`** som skriver opplysninger om alle varene.

p) Lag et enkelt main- program der du:

- oppretter en butikk med plass til 100 varer og prøver ut alle metodene i klassen **`Butikk.`**

Algoritme for å sette inn sortert i en tabell: Lag en tegning med eksempel!

```
//Finne rett plass
  sett i til 0
   så lenge(i mindre enn antall og element er større enn tabell[i]){
        øk i med 1
  }
// i peker nå på plassen der elementet skal inn
// lage plass ved å skifte elementene bakover f.o.m. indeks i
sett j til antall
så lenge( j er større enn i ){
     sett tabell[j] til tabell[j-1]
     mink j med 1
}
sett tabell [i] til element
øk antall med 1
}
```

Algoritme for å slette i en sortert tabell. Lag en tegning med et eksempel!

```
finn om mulig indeks til elementet som skal slettes
hvis indeks er ulik -1{
   mink antall med 1
   // skift elementene som er plassert etter det vi fjerner, en plass opp
   sett i = indeks
   så lenge( i er mindre enn antall){
        sett tabell[i] til tabell[i+1]
        øk i med 1
   }
   sett tabell[antall] til null-referanse
 }
```

Ofte vil vi returnere det elementet vi slettet, men ikke i denne oppgaven.
antall brukt i algoritmene ovenfor er det samme som antallTyper i klassen Butikk.