UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i: INF1010 – Objektorientert programmering
Eksamensdag: Torsdag 13. august 2015
Tid for eksamen: 9:00 – 15:00
Oppgavesettet er på 4 sider
Vedlegg: Kapittelet om tråder fra læreboka
Tillatte hjelpemidler: Læreboka

Kontroller at oppgavesettet er komplett før du begynner å besvare spørsmålene.

Betaling og utleie i varehuskjeden SGBO

Les hele oppgavesettet før du starter på oppgave 1. Pass på å bare svare på det som oppgavene spør om. Hvis du synes noe er uklart i oppgavene så skriv ned og begrunn dine egne fornuftige forutsetninger.

Varehuskjeden SGBO skal oppgradere betalingssystemet sitt, og du ønsker å tilby dine tjenester og programmere en løsning for dem. Som en test på om du er flink nok til dette er varehuskjeden og du blitt enige om at du skal løse noen deler av det som kan bli et nytt moderne betalingssystem.

Innledning til oppgave 1. Klassehierarki for varer. Vekt 20%

Du skal skrive et interface som heter NavnOgPris. Alle objekter som implementerer dette interfacet skal kunne oppgi sitt (entydige) navn og sin pris.
Du skal lage et klassehierarki med klassen Vare som superklasse. Klassen Vare skal implementere grensesnittet NavnOgPris.
I dette lille eksemplet skal du foreløpig bare ta med to typer varer: maskiner og verktøy. Maskiner har et antall hestekrefter, mens et verktøy har en vekt.
Varehuskjeden leier også ut noen av varene sine. Både noen maskiner og noen verktøy leies ut.
Klassehierarkiet skal inneholde fire klasser som det skal kunne lages objekter av: MaskinerTilUtleie, VerktøyTilUtleie, MaskinerTilSalgs og VerktøyTilSalgs. Alle andre klasser i klassehierarkiet skal det ikke kunne lages objekter av. Alle klasser skal ha konstruktører som initialiserer alle variable i objektene. Alle objekter skal inneholde en variabel nettopris som den endelige prisen blir regnet ut fra.

Prisen på maskiner som er til salgs er nettoprisen pluss moms på 25%. For å øke egeninnsatsen til husbyggere har Stortinget bestemt at momsen på verktøy til salgs bare skal være 10%. Prisen på maskiner som er til utleie er i virkeligheten et depositum, og du kan regne med at dette er dobbelt så høyt som nettoprisen. På verktøy som leies ut er depositumet (dvs. prisen) 1,5 ganger nettoprisen.

MaskinerTilSalgs og VerktøyTilSalgs har ingen flere egenskaper enn henholdsvis Maskiner og Verktøy.

Alle ting som er til utleie (i denne oppgaven blir det bare objekter av MaskinerTilUtleie og VerktøyTilUtleie) skal kunne behandles på samme måte og ha følgende egenskaper: 1) Den skal kunne leies ut. 2) Den skal kunne leveres tilbake. 3) Man skal kunne spørre den om den allerede er utleid. (I oppgave 2 skal du bruke disse egenskapene.)

Oppgave 1. Tegn opp dette klassehierarkiet. Programmer alle klassene og alle interfacene.

Innledning til oppgave 2: Behandling av utleie. Vekt 25%

I oppgave 2 skal du skrive klassen VareRegister, men du skal bare programmere det å leie ut og det å levere tilbake varer som er i denne beholderen. Selv om du ikke har løst oppgave 1 skal du bruke resultatene (klassene) derfra.

I oppgaven 2 skal du for enkelhets skyld anta at det bare er én vare av hver type (entydig identifisert med varenavn) i beholderen. Beholderen inneholder både varer som kan leies ut og varer som er til salgs. Du må ta hensyn til at det senere kan være andre varer enn bare maskiner og verktøy i beholderen. Du kan selv bestemme hvordan varene er lagret i beholderen, og du må gjerne bruke klasser fra Java-biblioteket. Du skal bare skrive to (public) metoder i denne beholderen. Du skal f.eks. ikke skrive metoder for å sette inn varer. De to metodene du skal skrive er:

leie – Denne metoden har som parameter et varenavn. For å lette feilfinning er dere i dette testprogrammet blitt enige om at metoden skal returnere en String som beskriver resultatet av metoden:
1) Hvis varen med dette navnet ikke finnes i beholderen returneres "Beklager, finnes ikke".
2) Hvis varen med dette navnet allerede er utleid returneres "Beklager, allerede utleid".
3) Hvis dette er en vare som ikke leies ut, returneres "Beklager, denne leier vi ikke ut".

Dersom alt gikk bra, og varen kan leies, skal det markeres i varen at den er leid ut, og metoden skal returnere: "Vær så god, varen er din til leie". (I dette enkle eksemplet skal det ikke holdes orden på hvem som har leid hva.)

levereTilbake – Denne metoden har som parameter navnet på varen som leveres tilbake. Også denne metoden skal returnere en String som beskriver resultatet av tilbakeleveringen: Hvis navnet ikke finnes i beholderen returneres "Denne varen finnes ikke". Hvis varen ikke har vært utleid (eller ikke er til utleie) returneres: "Beklager, denne er ikke utleid". Men hvis alt gikk bra leveres varen tilbake og det returneres: "Takk for at du leverte tilbake".

Du skal i oppgave 2 ikke ta hensyn til pris eller depositum på varer som leies ut.

Oppgave 2. Skriv klassen VareRegister som skal kunne lagre varer og de to metodene leie og levereTilbake i denne klassen. Du skal programmere datastrukturen som lagrer varer, men ingen andre (public) metoder enn disse to.

Innledning til oppgave 3: Lister av varer. Vekt 25%

I oppgave 3 skal du lage deler av et nytt betalingssystem. Du skal skrive en generisk beholder, kalt ListeMedStatus, som skal kunne inneholde en lenket liste av varer som en kunde kjøper. I oppgave 3 skal du ikke bruke noen av de ferdiglagede beholderne i Javas bibliotek, men programmere listen selv.

Når en kunde plukker en vare og legger den i handlekurven, skal varen også registreres ved å legges til denne beholderen. Når kunden kommer til kassen, skal det som er i handlekurven være det samme som det som er i beholderen, og det kunden skal betale er totalprisen for alle varene i beholderen. Noen av varene kan være leide ting. Prisen på de forskjellige varene er slik som beskrevet i oppgave 1.

Oppgave 3 bygger på oppgave 1, men ikke på oppgave 2. I oppgave 3 skal du således ikke bruke VareRegisteret fra oppgave 2, men derimot kan en kunde nå kunne kjøpe flere varer av samme type (samme navn).

Noen ganger når en kunde kommer til kassen, foretas det en kontroll av at kunden har registrert alle varene i handlekurven, dvs. at kurven inneholder det samme som beholderen. Til dette formålet skal du lage to metoder:

sjekk – Denne metoden tar et varenavn som parameter og går gjennom den lenkede listen i beholderen for å finne om denne varen er registrert. Hvis den finnes i beholderen (og ikke allerede er markert) blir den sjekket (markert) som ferdig kontrollert. Metoden returnere sann (true) hvis varen ble funnet i beholderen (og ikke var markert fra før) og usann (false) ellers. I det siste tilfellet betyr dette at kunden kan ha tenkt å snike med seg denne varen uten å betale.

Når alle varene i kurven er kontrollert skal egentlig alle varene i beholderen være sjekket (markert). Men hvis kunden uforvarende har registrert en eller flere varer uten å putte dem i handlekurven, vil de nå være i beholderen uten å være markert. Du skal derfor også skrive metoden

lagListeAvAlleUmarkerte – Denne metoden returnerer en ny beholder (av samme type) av alle varene i beholderen som ikke er markert.

Oppgave 3. Skrive den generiske beholderen ListeMedStatus med metodene leggTil, totalPris, sjekk og lagListeAvAlleUmarkerte. Parameteren til leggTil er en peker til det objektet som skal legges til beholderen. I innledningen til oppgave 3 har vi beskrevet beholderen som om den inneholder varer, men du skal programmere den generiske beholderen slik at den kan inneholde objekter av enhver klasse som implementerer grensesnittet NavnOgPris fra oppgave 1. Du skal implementere beholderen som en lenket liste som du programmerer selv. Skriv til slutt en deklarasjon som oppretter et objekt av klassen ListeMedStatus som kan inneholde alle varene fra oppgave 1.

Innledning til oppgave 4. Hjelp til utleie med tråder. Vekt 30%

Oppgave 4 er en utvidelse av oppgave 2, men du kan løse oppgave 4 selv om du ikke har løst oppgave 2. I hele oppgave 4 kan du bruke alle klassene du vil i Java-biblioteket.
Varekjeden SGBO legger opp til at det skal være flere ansatte som hjelper til med å finne fram og klargjøre varer som skal leies ut. Dette skal programmet ditt beskrive ved hjelp av tråder. Det skal være en tråd for hver ansatt som hjelper til med å leie ut varer, og en tråd for hver kunde som skal leie en vare. Alle kunder legger sine forespørsler om leie i en bestillingsbeholder, og her tar de ansatte bestillingene ut i rettferdig rekkefølge, dvs, først inn – først ut (FIFO).

En skisse av hva de to trådene skal gjøre:

Kunde:
- legge inn en bestilling på å leie en vare i bestillingsbeholderen
- vent på at bestillingen er utført
- ta imot svaret (og skriv det ut på samme måte som i oppgave 2)

Ansatt:
```
while (true) {
    - hent en bestilling fra bestillingsbeholderen
          (vent hvis det ikke er noen bestillinger der)
    - finn frem varen som skal leies i VareRegisteret
          og ta imot svar (som i oppgave 2)
    - vekk opp den ventende kunden og gi ham/henne svaret.
}
```

Oppgave 4a. Forklar svært kort hva du må gjøre med VareRegisteret i oppgave 2 for at flere tråder (flere ansatte) skal kunne operere på det samtidig.

Oppgave 4b. Forklar kort hvordan du lar en kunde vente på at bestillingen hans /hennes er utført og hvordan den ansatte skal vekke opp denne kunden når den ansatte har funnet frem varen som denne kunden skal leie. Det av denne løsningen du ikke programmerer under 4c, 4d og 4e kan du programmere her under 4b.

Oppgave 4c. Lag en klasse for bestillinger og en beholder for slike bestillinger. Ta med de metodene du trenger i beholderen for å løse oppgavene til kundene og de ansatte.

Oppgave 4d. Skriv tråden Kunde. Ta med som parametre til konstruktøren pekere til alt tråden trenger av data, bl.a. navnet på den varen som denne kunden skal leie.

Oppgave 4e. Skriv tråden Ansatt. Ta også her med som parametre til konstruktøren pekere til alt tråden trenger av data.

Lykke til!

Hilsen Stein Gjessing