

IN1010 Objektorientert programmering – eksamen våren 2022

(Bilkollektiv)

Oppgavetekst fra UiOs eksamen våren 2022 (Bilkollektiv), IN1010 (overskriften er eksamenssett.no sin; utsnittene under er klippet uendret ut av UiOs publiserte dokument)

Bilkollektiv

Du og noen venner har bestemt dere for å lage et bilkollektiv, og det er blitt din oppgave å skrive noen programmer for å administrere bilene. Det er lurt å lese gjennom hele oppgavesettet før du starter å programmere, for det finnes informasjon i senere oppgaver som kan gjøre tidligere oppgaver klarere. Du skal *ikke* lage et fullstendig program. Pass derfor på at du bare svarer på det oppgavene ber om. Svar ut over det du blir bedt om å gjøre, vil ikke telle med i beregningen av karakteren din. Hvis noe er uklart i oppgaven, gjør dine egne fornuftige forutsetninger og beskriv disse i besvarelsen.

Biler

Alle biler er subclasser av klassen `Bil`, men det skal ikke finnes objekter av klassen `Bil`. Bilkollektivet låner ut to typer biler, personbiler og varebiler.

Noen biler, både personbiler og varebiler, har den egenskapen at de er elektriske, og dette skal du programmere som et interface.

Oppgave 1. 5 poeng

Tegn opp klassehierarkiet for slike biler. Ta med interfacet.

Alle biler har et bilnummer (en `String`) og en pris (et positivt heltall) det koster å låne denne bilen en dag. Denne prisen er forskjellig for alle bilene. `Bil`-objekter skal kunne lenkes sammen i en toveis liste, og skal derfor inneholde to referansevariabler av klassen `Bil`. For elektriske biler må en kunne vite hvor stort batteriet er (et heltall). Personbiler har et antall passasjerer, mens varebiler har et lastevolum (et heltall). Alle instansvariabler,

unntatt referansevariablene av klassen `Bil`, skal være konstanter og få verdier i konstruktørene.

Alle klasser skal redefinere `toString()`-metoden slik at den tar med verdiene av alle instansvariablene unntatt referansevariablene av klassen `Bil`. Du skal bruke metoden `toString()` til utskrift senere i oppgavesettet, men det legges ikke vekt på om resultatet er oversiktlig eller pent.

Oppgave 2. 8 poeng

Programmer interfacet og alle klassene du tegnet i oppgave 1.

Brukerdialog

Programmet du skal lage senere i denne oppgaven skal kommunisere med brukeren på en skjerm. I første omgang skal denne kommunikasjonen foregå i kommandovinduet, men helt til slutt i dette oppgavesettet skal du legge inn muligheter for å kommunisere i et GUI-vindu. I programmet du skriver senere i oppgaven, skal du lett kunne skifte mellom disse to måtene å kommunisere på.

Du skal derfor først skrive et interface `Dialog` og deretter to klasser `TastaturDialog` og `GUIDialog` som implementerer dette interfacet på hver sin måte. Grensesnittet skal inneholde én metode `svarJaEllerNei(String sporsmal)` som har spørsmålet som parameter. Metoden skal stille spørsmålet og returnere **true** eller **false** avhengig av om brukeren svarer ja eller nei. I `TastaturDialog` skal brukeren svare enten "j" for ja eller "n" for nei. Du kan anta at brukeren aldri svarer noe annet, så ikke bruk tid på å lage denne metoden robust.

Hint: I klassen `TastaturDialog` må det være en instansvariabel av klassen `Scanner` som refererer et `Scanner`-objekt som lages slik: `new Scanner(System.in)`.

Oppgave 3. 8 poeng

Skriv interfacet `Dialog` og klassen `TastaturDialog`. (Klassen `GUIDialog` skal skrives i oppgave 11.)

Bilkollektiv

Vi skal nå innføre en klasse `Bilkollektiv` for å administrere bilene.

Antall biler i kollektivet er en konstant `AB` som initieres av konstruktøren. I `BilKollektiv`-klassen skal det være en array kalt `alleBilene` som er indeksert fra 0 til `AB-1`. I resten av oppgavesettet kan du regne med at denne arrayen er helt full av referanser til bil-objektene som representerer alle bilene i kollektivet, men du kan ikke anta noe om rekkefølgen av bilene i denne arrayen.

Videre skal klassen `BilKollektiv` ha en dobbeltlenket liste med alle biler som er ledige for utlån. Denne listen skal være sortert på pris, slik at den billigste bilen ligger først i listen. Klassen `BilKollektiv` skal inneholde to referansevariabler som peker på henholdsvis starten og slutten av denne listen av ledige biler. Referansevariablene som skal til for å lenke sammen elementene i den dobbeltlenkede listen er de du allerede har deklart i klassen `Bil`.

Oppgave 4. 5 poeng.

Tegn et objekt av klassen `BilKollektiv` med arrayen `alleBilene`. Anta i denne tegneoppgaven at `AB` er 3 og refererer tre bil-objekter av tre forskjellige bil-klasser. Ta med i tegningen alle instansvariablene med passe verdier i alle objektene inkludert de to referansevariablene som peker på henholdsvis starten og slutten av listen av ledige biler. Tegn opp den lenkede listen slik at to av bilene er ledige for utlån. Pass på at alle referanser (pekere) er riktig. Ikke ta med noen metoder.

Oppgave 5. 4 poeng

Skriv klassen `BilKollektiv` med instansvariabler som beskrevet ovenfor. Når et objekt av denne klassen opprettes, er ingen biler lagt inn i kollektivet ennå. Referansene i arrayen er alle null, og den dobbeltlenkede listen med ledige biler er dermed tom (referansevariablene til det første og siste elementet i listen er null). Du trenger ikke skrive noen metoder i denne oppgaven.

Oppgave 6. 14 poeng

Du skal nå skrive en metode `void lagBilPris()` i klassen `BilKollektiv`. Metoden skal anta at AB Bil-objekter er opprettet og refereres av `alleBiler`. (Dette gjøres av kode som ikke er en del av eksamensoppgaven.) Metoden `lagBilPris()` skal opprette den lenkede listen av ledige biler. Når metoden kalles, er alle bilene i bilkollektivet foreløpig ledige for utlån og skal derfor legges i listen. Dette er en metode som ikke skal utføres ofte, så det er viktigere at den er oversiktlig enn at den er rask. Husk at bilene i den dobbeltlenkede listen skal være sortert med billigste bil først.

Hint: Det mange måter å gjøre dette på, men en enkel måte er å først finne den billigste bilen og dens pris og legge denne bilen først i listen. Kall denne bilens pris `minForrigePris`. Finn deretter den billigste som er igjen, dvs den billigste som har verdi større enn `minForrigePris` og legg den inn bakerst i listen, osv. Husk at alle bilene har ulik pris.

Oppgave 7. 8 poeng.

Skriv metoden `void taUtBil(Bil b)` i klassen `BilKollektiv` som fjerner bilen `b` fra den lenkede listen slik at bilen ikke lenger er ledig for utlån. Vi kan anta at `b` er i listen.

Velge bil

Du skal nå skrive metoden `Bil velgBil(Dialog d)` i klassen `BilKollektiv`. Metoden skal kommunisere med brukeren og gå gjennom alle de ledige bilene til brukeren finner en han eller hun er fornøyd med. Kommunikasjonen skal foregå via dialogen `d` og altså skje enten i kommandovinduet eller i et GUI-vindu.

Programmet skal først spørre brukeren om vedkommende bare er interessert i å låne en elbil. I så fall skal programmet hoppe over alle ledige biler som ikke er elbiler.

Metoden `velgBil()` skal først skrive ut den billigste bilen (som er ledig) og spørre om brukeren ønsker å låne denne. Om brukeren svarer nei, skrives den nest billigste bilen ut, osv helt til brukeren finner en bil som vedkommende vil låne. Da skal bilen fjernes fra den lenkede listen for å reflektere at bilen er leid ut.

Gjør dine egne fornuftige antagelser om du synes noe er uklart i denne oppgaven. Ikke bruk tid på å lage metoden robust.

Oppgave 8. 14 poeng.

Skrive metoden `velgBil()` i klassen `BilKollektiv`.

Rekursivt bilvalg

Du skal nå skrive en modifisert metode av `velgBil()` som bruker rekursjon for å finne en ledig bil for brukeren. Dette gjøres med den rekursive metoden `Bil finnBilR()` som skal skrives inne i klassen `Bil`.

`finnBilR(Dialog dialog, boolean kunElektrisk)` skal ha to parametre og returnere en referanse til den bilen brukeren ønsker å låne. Den første parameteren er en referanse av interface-et du laget i oppgave 3, den andre parameteren er **true** om brukeren bare ønsker å låne en elektrisk bil. Hvis den andre parameteren sier at det bare er interesse for elektriske biler og metoden utføres inne i en bil som ikke er elektrisk, skal metoden ikke gjøre mere med denne bilen men fortsette rekursjonen om det er mulig. Hvis bilen er aktuell, skal metoden bruke den første parameteren til å spørre brukeren om vedkommende vil leie denne bilen. Hvis brukeren ønsker å leie bilen, returnerer metoden med en referanse til bilen selv, ellers fortsetter rekursjonen om mulig.

Oppgave 9. 11 poeng.

Skriv den rekursive metoden `finnBilR()` i klassen `Bil` som oppfører seg slik som beskrevet over.

Lag så en ny versjon av `velgBil()` kalt `velgBilR()`. I denne metoden skal du fremdeles først spørre brukeren om vedkommende bare vil leie elbiler. Deretter starter du rekursjonen ved å kalle på `finnBilR()` i den første ledige bilen.

Oppgave 10. 5 poeng.

Skriv den modifiserte metoden `velgBilR()`.

GUI-dialog

Nå skal du programmere `GUIDialog` som implementerer `Dialog` (se oppgave 3) og som gir brukeren mulighet til å kommunisere med programmet i et GUI-vindu som kan se slik ut:



Første gang `svarJaEllerNei()` kalles, skal metoden åpne et slikt GUI-vindu. Spørsmålet er parameter til metoden. Når brukeren klikker på JA- eller NEI-knappen, skal metoden returnere med henholdsvis **true** eller **false**.

Dette vinduet skal ikke lukkes så lenge programmet kjøres. Det betyr at neste gang `svarJaEllerNei()` kalles, skal metoden bruke samme vindu og bare endre spørsmålet.

Oppgave 11. 18 poeng

Skriv klassen `GUIDialog` med metoden `svarJaEllerNei()`.

Hint. Når metoden `svarJaEllerNei()` skal vente til brukeren klikker på JA eller NEI, må den sove. Dette kan gjøres med `Thread.sleep()` med en ganske stor verdi, for eksempel

```
try {  
    Thread.sleep(1000000);  
} catch (InterruptedException e) {}
```

Når `ActionListener`-objektet utføres (fordi brukeren har klikket på JA eller NEI), må den vekke hovedtråden. Dette kan gjøres med

```
hovedtrad.interrupt();
```

`GUIDialog`-objektet må ha en referanse til hovedtråden; dette kan den få ved å utføre

```
Thread hovedtrad = Thread.currentThread();
```