Bilkollektiv

Sensurveiledning (med oppgaveteksten)

- *Skrivefeil ignoreres vanligvis, men feil som kan tyde på manglende forståelse (for eksempel kopiering av kode som ikke passer inn eller som ikke svarer på oppgaven) vil gi trekk. Norske tegn er helt OK*
- *Gjennomgående litt større feil/ mangler vil det bli trukket for (selv om de er små hver for seg)*
- *Eksempler på feil/unøyaktigheter som ignoreres:*
  - *Mangler (noen få) ';' , '}' eller '{' om koden ellers gir mening*
  - *Småfeil i (gjenkjennbare) navn*
  - *Import-setninger mangler*
  - *Det legges lite vekt på riktige aksessmodifikatorer (private og public)*

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 subklasser 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.

Her skal du vise at du vet hva et klassehierarki er og at du kan tegne opp et slikt. I tillegg skal du vise at du kan tegne interface (f.eks. kalt Elektrisk) i et slikt hierarki. Bil skal helst være merket som abstract. Det skal være tre nivåer, Bil øverst, de to klassenen f.eks. Personbil og Varebil på nivå to, og nederst, på nivå tre to klasser som også implementerer interface-et Elektrisk.

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.

Læringsmål oppgave 2.
Her skal du vise at du kan programmere klasser og interface. Det legges vekt på at alle instansvariabler er riktig deklarert og at de er deklarert som konstanter (med final), at konstruktørene er riktig programmert (med super()), at interface-et er riktig programmert med en metodesignatur og at klassene som implementerer interface-et har en integer konstant som angir størrelsen på batteriet og at metoden i interfacet returnerer denne verdien.
I tillegg skal den polymorfe metoden toString() defineres riktig med riktig kall på den samme metoden i superklassen (super.toString()).

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.)

Læringsmål oppgave 3. Her skal du vise at du kan deklarere et interface med den riktige metoden. Det er viktig at du implementerer dette interfacet i klassen TastaturDialog ved å

opprette datastrukturen som er gitt i hintet og implementerer metoden svarJaEllerNei() slik at den leser fra tastatur.

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 dobbeltlenkete listen er de du allerede har deklarert 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.

Læringsmål oppgave 4. Her skal du vise at du skjønner hvordan en datastruktur med en array der elementene refererer objekter ser ut og hvordan en dobbeltlenket liste ser ut. Alle objektene skal refereres fra arrayen, men bare to av objektene ligger i den dobbeltlenkede listen. To referansevariabler skal peke ut hhv. starten og slutten av denne listen. Minst ett av objektene må ha egenskapen Elektrisk.
Det er ok om tegningen ikke viser hvilke egenskaper som evt. arves fra superklasser

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.

Læringsmål oppgave 5. Her skal du vise at du kan deklarere en array av referanser av riktig type og lengde, to referansevariable av riktig type (Bil) og en final int kalt AB. AB og arrayen skal settes og opprettes av en konstruktør.

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 legge den inn bakerst i listen, osv. Husk at alle bilene har ulik pris.

Læringsmål oppgave 6. Her skal du vise at du kan programmere en mer kompleks algoritme som oppretter en dobbeltlenket sortert liste. Det er viktig at metoden er oversiktlig programmert. Metoden vil vanligvis innholde en dobbelt løkke som henter et og et objekt fra arrayen og legger det inn i den dobbeltlenkede listen.

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.

Læringsmål oppgave 7. Her skal du vise at du kan programmere en algoritme som fjerner et element fra en dobbeltkjedet liste. Det er viktig at programmet behandler tilfellet at det bare er én bil i listen og at objektet først og sist tas ut riktig.

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`.

Læringsmål oppgave 8. Her skal du vise at du kan programmere en litt mer kompleks algoritme og bruke metoder som allerede er skrevet tidligere i oppgavesettet. Det er viktig at parameteren d brukes riktig ved at d.svarJaEllerNei() kalles uten at metoden vet noe mer om d. Operatoren instanceof kan brukes for å teste om objektet har egenskapen Elektrisk (polymorfi er også en mulighet). Uten at det står eksplisitt i oppgaven bør tilfellet at brukeren ikke vil ha noen bil også håndteres riktig.

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.

Læringsmål oppgave 9. Her skal du vise at du kan programmere en rekursiv metode inne i et objekt. Her nås rekursjonsbunnen når det ikke er flere objekter i listen (nestepeker er null), dvs at brukeren ikke har valgt noen bil . Flere løsninger ok:

- *med polymorfi der enten elbil-klassene eller Varebil/ Personbil-klassene Overrider metoden i Bil eller*
- *ved bruk av Instanceof i metoden i Bil*

Når en bil er funnet returnerer metoden med verdien this.

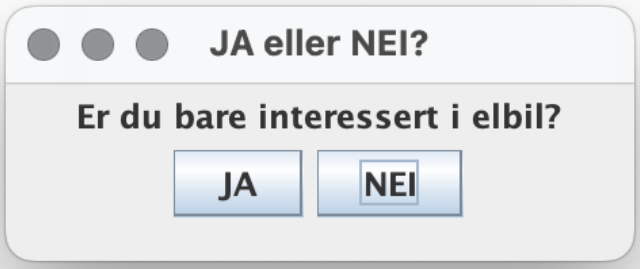
Lag så en ny versjon av `velgBil()` kalt `velgBilR()`. I denne metoden skal du fremdeles først spørre brukeren om vedkommende bare vil leie elbiler. Deretter starter du rekursjonen ved å kalle på `finnBilR()` i den første ledige bilen.

Oppgave 10. 5 poeng.
Skriv den modifiserte metoden `velgBilR()`.

Læringsmål oppgave 10. Her skal du vise at du kan starte en rekursjon på riktig måte. Metoden må starte med å kalle den rekursive metoden i det første objektet i listen.

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

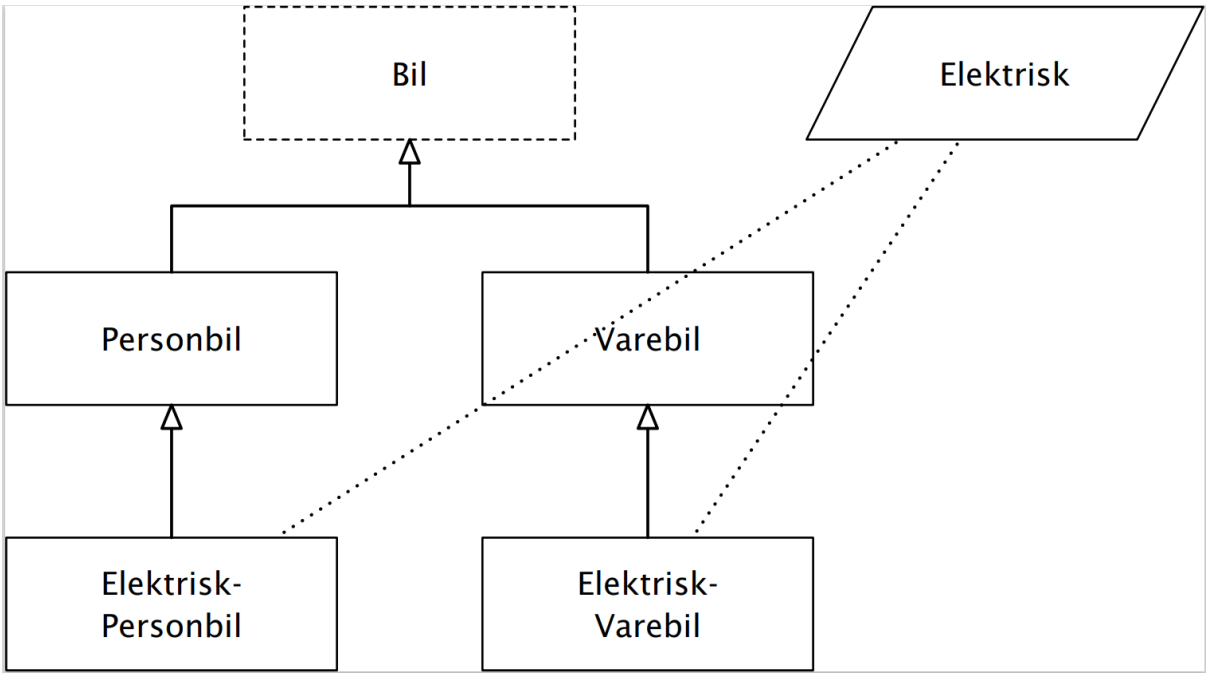
```
Thread hovedtrad = Thread.currentThread();
```

Læringsmål oppgave 11. Her skal du vise at du kan lage et GUI-vindu som kommuniserer med brukeren. Det er viktig at du oppretter vinduet riktig.
Videre er det viktig at du lager et eller to objekter med hendelseshåndteringsmetode(r) som håndterer knappetrykk på de to knappene, ja og nei.
Med de hintene som gis i oppgaven skal du helst vente når vinduet er opprettet og starte (avbryte) hoveddtråden igjen fra hendelseshåndteringsmetoden. Når svarJaellerNei() kalles for andre gang og senere skal det testes om vinduet alt er opprettet.

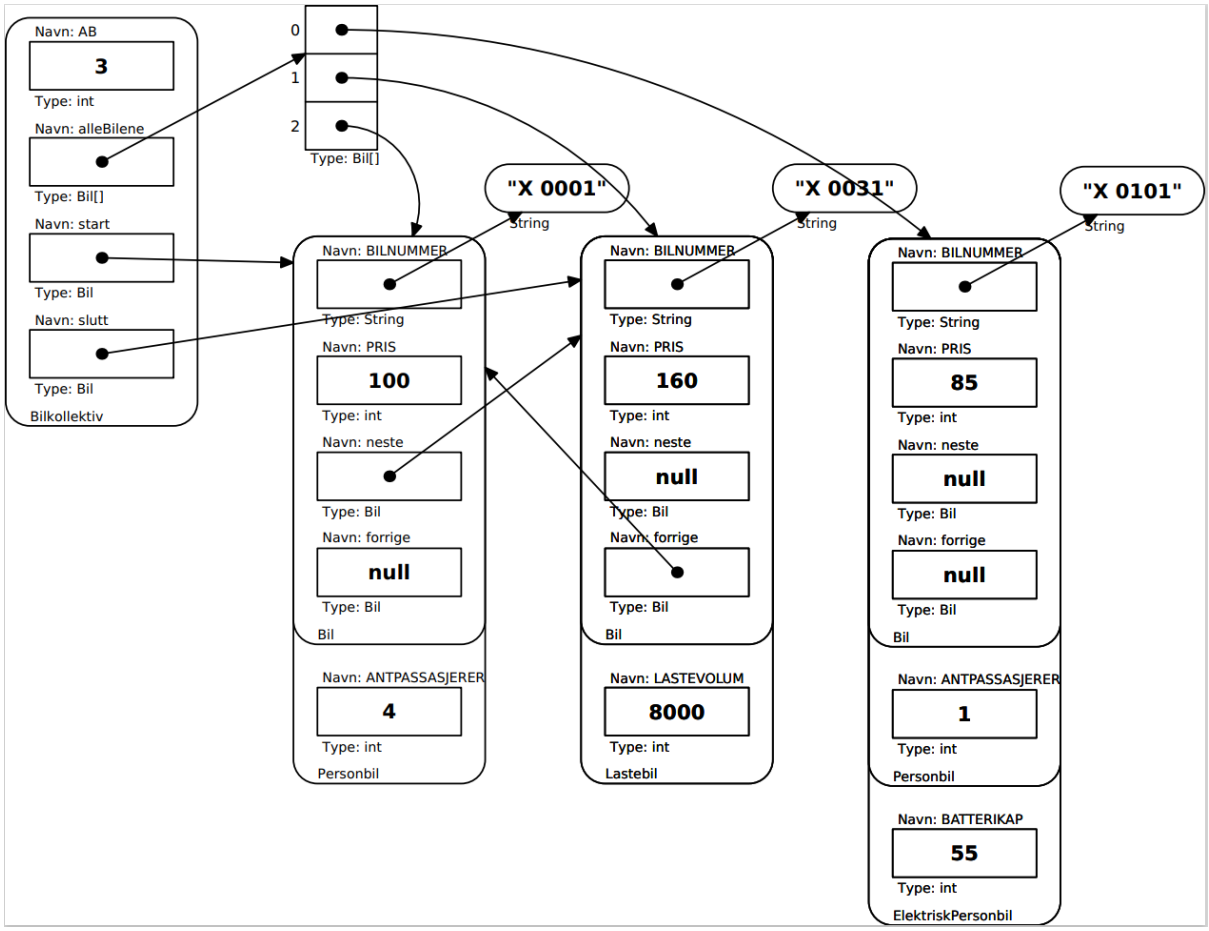
- *oppretter vinduer og knapper (viktig)*
- *håndterer hendelsene (med ActionListener) (viktig)*
- *kommuniserer svar tilbake til hovedtråd*
- *oppretter vindu kun første gang (mindre viktig)*

Løsningsforslag Eksamen 2022 Bilkollektiv

Oppgave 1



Oppgave 4



Kode

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.Scanner;

abstract class Bil {
    final String BILNUMMER;
    final int PRIS;
    Bil neste = null, forrige = null;  // Listepekere

    Bil (String nr, int kr) {
    BILNUMMER = nr;  PRIS = kr;
    }

    boolean erElbil () {
    return false;
    }
```

```
    Bil finnBilR (Dialog dialog, boolean kunElektrisk) {
    if (! kunElektrisk && dialog.svarJaEllerNei("Liker du " + this + "?"))
        return this;

    if (neste != null)
        return neste.finnBilR(dialog, kunElektrisk);
    return null;
    }

    @Override
    public String toString () {
    return BILNUMMER + " pris " + PRIS;
    }
}

class Personbil extends Bil {
    final int ANT_PASSASJERER;

    Personbil (String nr, int kr, int ant) {
    super(nr, kr);
    ANT_PASSASJERER = ant;
    }

    @Override
    public String toString () {
    return "Personbil " + super.toString() + " " + ANT_PASSASJERER + " pas";
    }
}

class Varebil extends Bil {
    final int LASTEVOLUM;

    Varebil (String nr, int kr, int vol) {
    super(nr, kr);
    LASTEVOLUM = vol;
    }

    @Override
    public String toString () {
    return "Varebil " + super.toString() + " " + LASTEVOLUM + " volum";
    }
}

interface Elektrisk {
    int hentBatterikapasitet ();
}
```

```
class ElektriskPersonbil extends Personbil implements Elektrisk {
    final int BATTERIKAP;

    ElektriskPersonbil (String nr, int kr, int vol, int kap) {
    super(nr, kr, vol);
    BATTERIKAP = kap;
    }

    @Override
    boolean erElbil () {
    return true;
    }

    @Override
    Bil finnBilR (Dialog dialog, boolean kunElektrisk) {
    if (dialog.svarJaEllerNei("Liker du " + this + "?"))
        return this;

    if (neste != null)
        return neste.finnBilR(dialog, kunElektrisk);
    return null;
    }

    @Override
    public int hentBatterikapasitet () {
    return BATTERIKAP;
    }

    @Override
    public String toString () {
    return super.toString() + " " + BATTERIKAP + " Kwh";
    }
}

class ElektriskVarebil extends Varebil implements Elektrisk {
    final int BATTERIKAP;

    ElektriskVarebil (String nr, int kr, int vol, int kap) {
    super(nr, kr, vol);
    BATTERIKAP = kap;
    }

    @Override
    boolean erElbil () {
    return true;
    }

    @Override
```

```
    Bil finnBilR (Dialog dialog, boolean kunElektrisk) {
    if (dialog.svarJaEllerNei("Liker du " + this + "?"))
        return this;

    if (neste != null)
        return neste.finnBilR(dialog, kunElektrisk);
    return null;
    }

    @Override
    public int hentBatterikapasitet () {
    return BATTERIKAP;
    }

    @Override
    public String toString () {
    return super.toString() + " " + BATTERIKAP + " Kwh";
    }
}

interface Dialog {
    boolean svarJaEllerNei (String spoersmal);
}

class TastaturDialog implements Dialog {
    Scanner tastatur = new Scanner(System.in);

    @Override
    public boolean svarJaEllerNei (String sporsmal) {
    while (true) {
        System.out.print(sporsmal + " ");
        String svar = tastatur.nextLine().trim().toLowerCase();
        if (svar.charAt(0) == 'j') return true;
        if (svar.charAt(0) == 'n') return false;
    }
    }
}

class GUIDialog implements Dialog {
    JFrame vindu = null;
    JPanel panel;
    JLabel tekstfelt;
    JButton jaknapp, neiknapp;

    Thread hovedtrad = Thread.currentThread();
    boolean svaret = true;

    @Override
```

```
public boolean svarJaEllerNei (String sporsmal) {
if (vindu == null) {
    try {
    UIManager.setLookAndFeel(
        UIManager.getCrossPlatformLookAndFeelClassName());
    } catch (Exception e) { System.exit(1); }
    vindu = new JFrame("JA eller NEI?");
    vindu.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    panel = new JPanel();
    vindu.add(panel);

    tekstfelt= new JLabel(sporsmal);
    panel.add(tekstfelt);

    class SvarJaNei implements ActionListener {
    boolean svar;

    SvarJaNei (boolean jn) {
        svar = jn;
    }

    @Override
    public void actionPerformed (ActionEvent e) {
        svaret = svar;
        hovedtrad.interrupt();
    }
    }

    jaknapp = new JButton("JA");
    jaknapp.addActionListener(new SvarJaNei(true));
    panel.add(jaknapp);

    neiknapp = new JButton("NEI");
    neiknapp.addActionListener(new SvarJaNei(false));
    panel.add(neiknapp);

    vindu.pack();  vindu.setVisible(true);
} else {
    tekstfelt.setText(sporsmal);
}

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

return svaret;
}
```

```
}

class Bilkollektiv {
    final int AB;
    Bil[] alleBilene;
    Bil start, slutt;  // Liste av ledige biler

    Bilkollektiv (int ant) {
    AB = ant;
    alleBilene = new Bil[AB];
    start = slutt = null;
    }

    void lagBilPris () {
    int minForrigePris = -1;
    for (int n = 1;  n <= AB;  ++n) {
        // Finn den billigste bilen blant de som er igjen:
        Bil billigst = null;
        for (int i = 0;  i < AB;  ++i) {
        Bil b = alleBilene[i];
        if (b.PRIS>minForrigePris && (billigst==null || b.PRIS<billigst.PRIS))
            billigst = b;
        }

        // Sett bilen inn sist i listen:
        if (start == null) {
        start = slutt = billigst;
        } else {
        slutt.neste = billigst;
        billigst.forrige = slutt;
        slutt = billigst;
        }
        minForrigePris = billigst.PRIS;
    }
    }

    void visBilene () {
    /* Til testing (ikke bedt om i oppgaven) */
    System.out.println("Test: Bilene er (sortert):");
    Bil b = start;
    while (b != null) {
        System.out.println("   " + b);
        b = b.neste;
    }
    System.out.println();
    }

    void taUtBil (Bil b) {
```

```
    if (b==start && start==slutt) {
        start = slutt = null;
    } else if (b == start) {
        start = start.neste;
        start.forrige = null;
    } else if (b == slutt) {
        slutt = slutt.forrige;
        slutt.neste = null;
    } else {
        b.forrige.neste = b.neste;
        b.neste.forrige = b.forrige;
    }
    b.neste = b.forrige = null;
    visBilene();
    }

    Bil velgBil (Dialog d) {
    boolean kunElbil = d.svarJaEllerNei("Er du bare interessert i elbil?");
    Bil b = start;
    while (b != null) {
        if (b.erElbil() || ! kunElbil) {
        if (d.svarJaEllerNei("Liker du " + b + "?")) {
            taUtBil(b);
            return b;
        }
        }
        b = b.neste;
    }
    return null;
    }

    Bil velgBilR (Dialog d) {
    boolean kunElbil = d.svarJaEllerNei("Er du bare interessert i elbil?");
    Bil b = start.finnBilR(d, kunElbil);
    if (b != null)
        taUtBil(b);
    return b;
    }

    /* Hovedprogram (ikke bedt om i oppgaven) */
    public static void main (String[] arg) {
    Bilkollektiv kol = new Bilkollektiv(3);
    kol.alleBilene[0] = new Personbil("AA00001", 350, 4);
    kol.alleBilene[1] = new ElektriskVarebil("AA00002", 745, 21, 50);
    kol.alleBilene[2] = new ElektriskPersonbil("AA00003", 310, 3, 45);
    kol.lagBilPris();
    kol.visBilene();
```

```
    Dialog d = new TastaturDialog();
    // Dialog d = new GUIDialog();
    for (int i = 1;  i <= 3;  ++i) {
        // Bil b = kol.velgBil(d);
        Bil b = kol.velgBilR(d);
        if (b == null)
        System.out.println("Ingen bil passet.");
        else
        System.out.println("Bil nr " + i + " er " + b + ".");
        kol.visBilene();
    }
    System.exit(0);
    }
}
```