Utsatt/ny eksamen IN1010 våren 2023
med sensorveiledning

I dette oppgavesettet skal du ikke bruke andre klasser fra Java-biblioteket enn de som alltid følger med i java.lang, med unntak av klassen NoSuchElementException og det du trenger for å lage en iterator. På eksamen trenger du ikke skrive import-setninger for disse.

Pass på at du bare svarer på det hver enkelt oppgave ber om. Du skal ikke programmere et fullstendig program med en main-metode før på slutten av oppgavesettet (oppgave 9). Du kan gjerne bruke norske bokstaver. I noen oppgaver er det presisert hvilke læringsmål du skal vise du har nådd, men i de fleste oppgavene er dette opplagt.

Sensorveiledning:

Skrivefeil ignoreres vanligvis, men feil som kan tyde på manglende forståelse (for eksempel kopiering av kode som ikke passer inn eller svarer på oppgaven) skal gi trekk.

• Ignoreres (eksempler):

- *Mangler (noen) ';' , '}' eller '{' om koden ellers gir mening (NB!)*
- *Småfeil i (gjenkjennbare) navn*
- *Blander metodenavn for tilsvarende operasjon i ulike beholdere*

• Kan gi (noe) trekk

- *Bruker beholder på en måte som tyder på manglende forståelse*
- *Gjennomgående feil/ mangler*

Studentene har fått beskjed om at kommentarer og tilgangsmodifikatorer (public/ private/ protected) ikke er viktige på eksamen.

Selvkjørende biler i kolonner

For å øke veikapasiteten og også sikkerheten til selvkjørende biler er det blitt foreslått at slike biler skal kjøre tett etter hverandre og på rekke i en kolonne (på engelsk: *platoon* eller *convoy*). I programmet ditt er alle biler selvkjørende, kan kjøre i kolonne og har egenskapene til klassen Bil. Enhver bil i programmet ditt er enten en personbil eller en lastebil. En kolonne består enten av bare personbiler eller av bare lastebiler. Kun spesielle biler kan kjøre først i en kolonne og lede denne, såkalte *lederbiler*. Denne egenskapen skal du beskrive ved hjelp av et interface du kaller Lederbil.

Oppgave 1 Tegning av klassehierarki (5 poeng)

⇨ Tegn et klassehierarki som kan representere ulike typer biler som beskrevet over. Begrunn valgene dine kort som kommentarer i programmet du skriver i oppgave 2.

Sensorveiledning:
I denne oppgaven skal studenten vise at hen forstår og kan beskrive et klassehierarki. Klassehierarkiet må inneholde den abstrakte klassen Bil på toppen og de andre klassene under denne med piler opp til superklassen, se løsningsforslaget. Interfacet Lederbil må være tegnet (som et interface) og to subklasser (f.eks. LederPersonbil og LederLastebil) må ha en pil opp til dette interfacet.

Oppgave 2 Programmering av klassehierarki (15 poeng)

Bilene i en kolonne skal kunne hektes sammen i en dobbeltlenket liste ved hjelp av instansvariabler i bil-objektene. Hver bil skal også inneholde en referanse til den kolonnen den kjører i. Denne referansen skal være **null** når bilen ikke er i noen kolonne.

Enhver bil har en maksimal fart. Enhver lastebil har en maksimal lastevekt og personbiler har et maksimalt antall passasjerer. Disse instansvariablene skal initieres av klassenes konstruktører. Lederbiler har en egenskap kalt egnethet som er et heltall. Lederbilenes egnethet skal initieres av konstruktørene.

⇨ Programmer klassene og interfacet du tegnet i oppgave 1

Sensorveiledning:
***I** denne oppgaven skal studenten vise at hen kan programmere enkle klasser og et interface. Extends og implements må være med på riktige måter, se løsningsforslaget. Instansvariabler skal deklareres og konstruktørene skal kalle på super() på riktig måte. Interfacet må ha en metode som returnerer en int (egnethet) og de to klassene som implementerer dette grensesnittet må en instansvariabel for egnethet som initieres av konstruktøren.*

Oppgave 3 Klassen Kolonne (20 poeng)

I denne oppgaven skal du skrive en klasse Kolonne. Objekter av denne klassen skal inneholde pekere til første og siste bil i en dobbeltkjedet liste av biler i kolonnen. Du skal bruke instansvariablene i bil-objektene til å lage denne listen. Metodene som setter inn og tar ut (dvs. fjerner) biler fra kolonnen, må sørge for at det alltid er en lederbil først i kolonnen (men det kan gjerne være én eller flere lederbiler inne i kolonnen også). I klassen Kolonne må programmet ditt sørge for at det

enten bare er lastebiler eller bare er personbiler i kolonnen, og dette bestemmes av den første bilen som settes inn.

Klassen Kolonne skal inneholde metoden settInn som skal sette inn bilen som refereres av parameteren på slutten av kolonnen, og metoden taUt for å ta ut (fjerne) en bil (referert av parameteren) av kolonnen. Metoden taUt må sjekke at bilen virkelig er i denne kolonnen. Metoden skal ikke lete etter bilen i kolonnen. Hvis taUt prøver å ta ut den første bilen i kolonnen, skal dette bare gå bra om også den neste bilen i kolonnen er en lederbil (eller kolonnen blir tom).

Deklarer en unntaksklasse som kastes når det settes inn eller tas ut en bil som gjør at det ikke blir en lederbil først i kolonnen, når en lastebil settes inn i en personbilkolonne (eller omvendt) eller bilen som skal ut, ikke er i kolonnen. Bruke en String-parameter til unntaksklassen for å skille de forskjellige unntakssituasjonene.

- ⇨ Skriv klassen Kolonne med de to metodene settInn og taUt. Skriv også unntaksklassen som beskrevet over.

Sensorveiledning:
Her skal studenten vise at hen behersker manipulering av referanser i en lenket liste. Det er viktig at programmet ikke leter gjennom listen, men bruker siste-pekeren til å sette inn sist. Det er også viktig at taUt ikke leter (som oppgaveteksten sier) etter bilen i kolonnen, men bruker egenskapene til en dobbeltlenket liste til å ta ut.

Det er viktig at peker-håndteringen gjøres riktig for denne listen av bil-objekter og bruker referansene som ligger i disse objektene (av klassen Bil) til forrige og neste objekt i listen. Det skal ikke brukes Node-objekter. Løsninger som bærer preg av å være hentet fra andre programmer uten at studenten viser at hen kan og skjønner manipulering av referanser i en liste, gir ikke uttelling.

Programmet må ta hensyn til alle spesialtilfeller av tom liste både ved innsetting og uttak (se løsningsforslaget). Det skal sjekkes hva slags biler som manipuleres ved hjelp av instanceof og riktige unntak skal kastes og differensieres ved hjelp av parameteren til unntaksklassen studenten har skrevet.

Oppgave 4 Iterator (15 poeng)

- ⇨ Skriv en iterator i klassen Kolonne som itererer over alle bilene i kolonnen *som har leder-egenskaper*. Hvis next-metoden i iteratoren blir bedt om å returnere en bil og det ikke er flere lederbiler igjen å iterere over, skal det kastes et unntak av typen NoSuchElementException. I denne oppgaven skal du vise at du skjønner hvordan en iterator skal programmeres når ikke alle objektene i beholderen skal tas med.

Sensorveiledning:
Her skal studenten vise at hen kan programmere en iterator der ikke alle objektene i listen skal returneres og der de som returneres har en bestemt egenskap. Konstruktøren må lete etter første interessante objekt og next() må finne neste (men som vanlig returnere forrige). Objektene skal differensieres ved bruk av instanceof. Lete-variabelen i Iterator-klassen må være at type Bil, men next() skal returnere en referanse av typen Lederbil, så programmet må typekonvertere fra Bil til Lederbil.

Oppgave 5 Rekursiv metode (10 poeng)

En kolonne kan ikke kjøre raskere enn bilen med lavest maksimalfart.

⇨ Skriv metoden finnMaksfartR i klassen Kolonne som finner kolonnens maksimale fart ved å kalle en rekursiv metode i klassen Bil. Skriv også den rekursive metoden i klassen Bil.

Sensorveiledning:
Her skal studenten vise at hen kan programmere en enkel rekursiv algoritme. Metoden i klassen Kolonne må kalle den rekursive metoden i det første objektet i listen (etter at den har sjekket at listen ikke er tom). Hele listen må løpes gjennom og den laveste maksfarten finnes når rekursjonen trekker seg tilbake.

Oppgave 6 Beste lederbiler (15 poeng)

I denne oppgaven skal du skriv en metode finnBesteEgnet i klassen Kolonne som returnerer en array med referanser til de 5 lederbilene i kolonnen som har størst (best) egnethet. Om det er færre enn 5 lederbiler i kolonnen, skal arrayen fylles opp med **null**-verdier. Husk at du ikke skal bruke klasser fra Java-biblioteket (annet enn de som er nevnt først i oppgaveteksten). Du kan bruke iteratoren beskrevet i oppgave 4 selv om du ikke har programmert denne. I denne oppgaven skal du vise at du behersker å bruke arrayer og kan programmere en noe mer kompleks algoritme.

⇨ Skriv metoden med signaturen Lederbil [ ] finnBesteEgnet () i klassen Kolonne.

Sensorveiledning:
Her skal studenten vise at hen kan (som det står i oppgaven) manipulere arrayer og programmere en litt vanskeligere algoritme. Om studenten velger å holde bilene sortert eller ikke har ingen betydning for bedømmelsen, men det bør komme klart fram fra koden eller kommentarer hvordan algoritmen er ment å virke. En lett forståelig algoritme gir bedre uttelling enn en algoritme som er vanskelig å skjønne.

Oppgave 7 Datastrukturtegning (5 poeng)

I denne oppgaven skal du tegne en datastruktur basert på klassen Kolonne i oppgave 3 og metoden finnBesteEgnet i oppgave 6. Anta at et objekt av klassen Kolonne inneholder fire lastebiler, og at både den første og den siste er lederbiler.

- ⇨ Tegn opp datastrukturen i dette Kolonne-objektet og tegn den arrayen det returneres en referanse til fra et kall på metoden finnBesteEgnet. Skriv på alle objektene hvilke klasser de er instanser av. Ikke tegn metoder. Skriv navn på variabler, men du trenger ikke skrive type og du trenger bare skrive/tegne innholdet i referansevariablene.

Sensorveiledning:
Her skal studenten vise at hen forstår og kan visualisere en datastruktur som oppstår under kjøringen av et program – se løsningsforslaget. Det er viktig at det tegnes objekter og variabler, ikke klasser. Det må være ett objekt av klassen Kolonne, to Lastebil-objekter og to LederLastebil-objekter. samt en array med 5 elementer (NB! navnene på disse lastebilklassene bestemmer studenten selv). Lederlastebilobjektene skal helst ha en instansvariabel for egnethet. Alle referansevariabler bør peke på riktige objekter, mens andre variabler kan være tomme.

Oppgave 8 Generisk klasse (10 poeng)

Du skal nå skrive en ny klasse KolonneG som er en meget forenklet generisk versjon av klassen Kolonne slik at når det deklareres et objekt av klassen KolonneG, bestemmes det om det skal være en kolonne av bare personbiler eller en kolonne av bare lastebiler. Klassen KolonneG skal ta vare på bilene på samme måte som klassen Kolonne, men for enkelhets skyld behøver det i klassen KolonneG ikke være en lederbil først i kolonnen og bil-objekter skal ikke peke på kolonnen sin. Du skal ikke gjøre forandringer i klassene i oppgave 2.

Klassen KolonneG skal ha to metoder:

- settInn skal legge inn en bil (referert av parameteren) bakerst i kolonnen.
- taUtSiste skal fjerne og returnere en referanse til den siste bilen i kolonnen. Hvis kolonnen er tom, returneres **null**.

Metodene skal ikke kaste noen unntak. I denne oppgaven er det viktig at du viser at du kan programmere en generisk klasse. Du kan legge mindre vekt på pekerlistehåndteringen (for det har du vist du kan i oppgave 3).

Tips: Ikke ta utgangspunkt i koden du skrev i oppgave 3 – programmer hele klassen på nytt.

⇨ Skriv klassen KolonneG med instansvariabler og de to metodene settInn og taUtSiste.

Sensorveiledning:
Her skal studenten vise at hen forstår og kan programmere en generisk klasse. Det er viktig at parameteren til klassen er en subklasse av Bil, at instansvariablene som peker på første og siste bil i den lenkede listen er av denne parametertypen, at parameteren til settInn er denne parametertypen og at taUt returnerer en referanse av denne typen - se løsningsforslaget.

Et poeng som ikke er fullt så viktig: Siden neste og forrige-pekerene i Bil-klassen er av typen Bil må det typekonverteres når siste-pekeren skal forandres (se løsningsforslaget).

Oppgave 9 Bruk av generisk klasse (5 poeng)

I denne oppgaven skal du vise at du behersker å bruke en generisk klasse. Du skal skrive en klasse Kolonnekjøring som skal bruke klasser fra oppgave 2 og klassen KolonneG fra oppgave 8. Til sammen skal dette kunne virke som et fullstendig program. Klassen Kolonnekjøring skal inneholde en main-metode som

1. Oppretter en (tom) kolonne av lastebiler
2. Oppretter en lastbil som er en lederbil
3. Setter denne bilen inn i kolonnen
4. Tar den siste (og eneste) bilen ut av kolonnen og skriver ut denne lederbilen sin egnethet ved hjelp av System.out.println().

⇨ Skriv klassen Kolonnekjøring med en main-metode som beskrevet over.

Sensorveiledning:
Her skal studenten vise at hen kan programmere en main-metode som bruker en generisk klasse. Deklarasjon av referanse til - og generering av objektet av typen KolonneG med Lastebil (eller det navnet studenten har brukt på lastebilene) som parameter, er det klart viktigste i denne oppgaven – se løsningsforslaget. Full uttelling gis det bare om lastebilen som tas ut typekonverteres til en lederlastebil (før egnetheten skrives ut).