Eksamen IN1010, våren 2024

I denne oppgaven skal du skrive deler av et lite program som behandler fly.

Les gjennom hele oppgaven før du begynner på besvarelsen. Tegning av klassehierarkiet skal leveres i oppgave 1, samtlige andre svar (det vil si all programkode) skrives i svarfeltet for oppgave 2–5. Angi gjerne hvilken oppgave du svarer på som kommentarer i programkoden.

Du kan bruke norske bokstaver i denne besvarelsen.

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.

• *Eksempler på småfeil som skal Ignorere:*

- *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 import-setninger, kommentarer, eller tilgangsmodifikatorer (public/ private/ protected) ikke er viktige på eksamen.

Oppgave 1. Klassehierarki for fly (5 poeng)

Fly representeres av klassen **Fly** som har følgende metoder:

hentId returnerer en String som identifiserer dette flyobjektet, f. eks. «LN-ABC».

hentMotorAntall returnerer et heltall som angir hvor mange motorer flyet har.

hentMTOW returnerer et heltall som angir flyets maksimale startvekt i kilogram som på engelsk heter *Maximum Take Off Weight* forkortet til MTOW.

MTOW, antal motorer (heltall) og deres totale trekkraft (heltall) i kW (kilowatt) og identifikatoren skal være parametere til klassens konstruktør. Flyobjekter skal kunne hektes sammen i en enkeltlenket liste i en senere oppgave, så du skal også ha med instansvariabler i klassen som gjør dette mulig.

I dette eksamenssettet er alt flymateriell enten av klassen **MotorFly** eller klassen **SeilFly**. Alle MotorFly-objekter er enten av klassen **LasteFly** eller klassen **PassasjerFly**.

Fly kan implementere et interface **Motordrevet** med følgende grensesnitt:

trekkraft som returnerer maksimal trekkraft av flyets motorer i antall kW som et heltall.

LasteFly har en maks tillatt lastevekt, angitt som et heltall i kilogram, og PassasjerFly har et heltall som representerer maks antall passasjerer det er plass til i flyet. Seilfly har et heltall som representerer minimum synkehastighet målt i cm/s.

Disse instansvariablene skal være parametere til konstruktørene til klassene.

- Tegn et klassehierarki for situasjonen beskrevet over.

Sensorveiledning:

I denne oppgaven skal studenten vise at hen forstår og kan beskrive et klassehierarki. Klassehierarkiet må inneholde den abstrakte klassen Fly på toppen og de andre klassene under denne med piler opp til superklassen. Interfacet Motordrevet må være tegnet (som et interface) og enten Fly eller Motorfly må ha en pil opp til dette interfacet. Også klassen Motorfly bør være abstrakt. Men andre fornuftige hierarkier er også OK. Om kandidaten har valgt å la flere klasser implementere interface Motordrevet, er dette OK. Grunnen til dette er at det ikke står eksplisitt i oppgaven hvilke fly som ikke skal ha motor. Ved eksamen har studentene fått beskjed om at seilfly kan ants IKKE å ha motor. SELVOM dette, så er det OK at Seilfly implementerer Motordrevet – trekkraft skal blot returner 0. Faktisk, så FINNES seilfly med motor, så Seilfly kan godt implementere Motordrevet og være subklasse av Motorfly.

Oppgave 2. Programmere klasser og interface (15 poeng)

I denne oppgaven skal du programmere klasser og interface som beskrevet i oppgave 1.

- Programmer klassene **Fly, MotorFly, SeilFly, LasteFly, PassasjerFly** og **Motordrevet**.

Sensorveiledning:

Hierarki: ***I*** *denne oppgaven skal studenten vise at hen kan programmere enkle klasser og et interface. Fly og Motorfly bør være abstrakt.* ***extends*** *og* ***implements*** *må være med på riktige måter (7 poeng)*

Konstruktører*: Instansvariabler skal deklareres og konstruktørene skal kalle på* ***super()*** *på riktig måte. Det er et pluss om konstante instansvariabler er deklarert som* ***final*** *(8 poeng).*

Oppgave 3. Klassen Flyformasjon (28 poeng)

I oppgave 3 skal du skrive en klasse **Flyformasjon** med variabler og metoder. En Flyformasjon er en enkelthektet lenket liste av Flyobjekter og du skal bruke instansvariablene i disse objektene til å lage denne listen. I denne oppgaven kan du aksessere instansvariabler i Fly-klassen og dens subklasser.

3a) 1 poeng

⇨ Skriv konstruktør og instansvariabler for klassen Flyformasjon med peker til første Fly-objekt i listen.

Sensorveiledning: Her skal studenten vise at hen kan deklarere en svært enkel datastruktur i en beholder som skal inneholde en enkelt lenket liste. Konstruktør kan være utelatt eller tom, eller den kan brukes til å sette førstepeker.

3b) 3 poeng

⇨ Skriv metoden **leggTil** i klassen Flyformasjon som legger til et Flyobjekt først i listen. Parameter til metoden er en referanse til objektet som skal legges til.

Sensorveiledning: Her skal studenten vise at hen behersker enkel manipulering av referanser i en lenket liste. Det er viktig at programmet ikke leter gjennom listen, men bruker første-pekeren til å sette inn først.

3c) 2 poeng

⇨ Skriv metoden **erMed** i klassen Flyformasjon som sjekker om et gitt fly er med i Flyformasjonen. Parameter til metoden er en tekststreng som er flyets identifikasjon. Metoden skal returnere true, hvis flyet er med i Flyformasjonen, og ellers false.

Sensorveiledning: Her skal studenten vise at hen behersker bruk av referanser i en enkeltlenket liste. Programmet må ta hensyn til alle spesialtilfeller. Det er viktig at dette gjøres riktig for denne listen av Fly-objekter og bruker referansene som ligger i disse objektene (av klassen Fly) til 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.

3d) 6 poeng

⇨ Skriv metoden **taUt** i klassen Flyformasjon som tar ut et element i listen. Parameter til metoden er en tekststreng som er identifikasjonen til flyet som skal tas ut. Metoden skal returnere en referanse til det objektet som tas ut. Hvis flyet ikke er med, så skal det returnes NULL.

Sensorveiledning: Her skal studenten vise at hen behersker noe vanskeligere manipulering av referanser i en enkeltlenket liste, spesielt å fjerne et element fra «midten» av en enkeltlenket liste. Programmet må ta hensyn til alle spesialtilfeller. Det er viktig at dette gjøres riktig for denne listen av Fly-objekter og bruker referansene som ligger i disse objektene (av klassen Fly) til 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. Besvarelsen må inneholde en while-løkke som terminerer når programmet har funnet objektet som skal tas ut, og, om et slikt ikke finnes, terminerer når listen er ferdig gjennomløpt. Det er mulig å bruke iteratoren som defineres i oppgave 3e), men dette gir bare et lite pluss om det er gjort riktig.

3e) 8 poeng

⇨ Skriv en iterator over alle Fly-objektene i en Flyformasjon.

Sensorveiledning: Her skal studenten vise at hen kan skrive en standard iterator over en enkeltlenket liste. Siden det har vært undervist akkurat slike iteratorer i løpet av semesteret er det viktig at det gjøres riktig, at studenten viser at hen har skjønt hvordan det gjøres og har tilpasset iteratoren til akkurat denne listen som utgjør et fly. Selve Flyformasjon-klassen må implementere Iterable<Fly>, det må finnes en metode Iterator<Fly> iterator () som lager et nytt Iterator-objet og det må være deklarert en klasse som implementerer Iterator<Fly>. I den sistnevnte klassen må metodene hasNext() og next() være deklarert og programmert riktig.

3f) 8 poeng

⇨ Skriv metoden **PassasjerFly[] hentPassasjerFly( )** i klassen Flyformasjon som returnerer en array med referanser til alle PassasjerFly-objekter i Flyformasjonen.
Hint: Tell antall passasjerfly før du oppretter arrayen.

*Sensorveiledning: Her skal studenten vise at hen kan opprette og legge inn referanser i en array og returnere denne fra metoden. I tillegg skal studenten vise at hen kan undersøke om et objekt har visse egenskaper (ved bruk av **instanceof**) og kan typekonvertere (caste) (fra Fly til PassasjerFly), se løsningsforslaget.*

Oppgave 4. Maksimum vekt (16 poeng)

4a) 8 poeng

Legg til en metode, **totalVekt,** til Flyformasjon. Metoden skal summere MTOW for alle fly i Flyformasjonen. Metoden skal bruke din iterator fra 3 e).

⇨ Skriv metoden **totalVekt**.

Sensorveiledning: Her skal studenten vise at hen kan skrive en metode som bruker en iterator. Metoden må spesialbehandle tilfellet at listen er tom.

4b) 8 poeng

Legg til en metode, **maksVekt,** til Flyformasjon. Metoden skal returnere den største MTOW blant alle fly i Flyformasjonen. Metoden skal gå gjennom listen av fly rekursivt.

⇨ Skriv metoden **maksVekt** rekursivt.

Sensorveiledning: Her skal studenten vise at hen kan skrive en metode som løper igjennom listen rekursivt. Metoden må spesialbehandle tilfellet at listen er tom. Det er OK at maksVekt bruker en rekursiv hjelpemetode.

Oppgave 5. Tråder (36 poeng)

I oppgave 5 skal du skrive én trådklasse som heter **Flygeleder** og én som heter **Pilot**. Begge disse skal implementere Runnable. Du skal også skrive en klasse **Rullebane** som skal være en monitor.
(Du skal altså ikke skrive noe fullstendig program som starter opp noen tråder – du skal bare skrive disse tre klassene med metoder og variabler.)

5 a) 10 poeng

Klassen **Rullebane** skal være en monitor hvor Flygeleder kan se hvilke fly som ønsker å starte og gir tillatelse til fly, så de kan starte. Flygelederen gir starttillatelse til ett fly om gangen ved å kalle **sjekkAvganger.** Etter retur fra **sjekkAvganger** venter Flygelederen 1 minutt (så flyet kan starte og forlate rullebanen) og kaller deretter **sjekkAvganger** igjen – for så å gi det neste flyet tillatelse ☺

Klassen **Rullebane** må ha følgende metoder:
sjekkAvganger som kalles av Flygelederen for å sjekke om der er fly som venter på starttillatelse. Hvis ikke returner Flygelederen – venter 1 minutt og deretter kaller **sjekkAvganger** igjen.
Hvis det er et eller flere fly som venter, så gir flygelederen tillatelse til å starte ved å signalere ett av de ventende flyene via en Condition variabel, **avventStartTillatelse**. Deretter returner Flygelederen – venter 1 minutt og deretter kaller **sjekkAvganger** igjen.
Metoden har ingen parametere.

hentStartTillatelse som kalles av Pilotene, når de gjerne vil starte. De venter i en Conditionvariabel **avventStartTillatelse**, som Flygelederen signalerer når et fly har fått starttillatelse.
Metoden har én parameter som er et Flyobjekt.

- ⇨ Skriv monitoren **Rullebane**, bruk **Condition** og **ReentrantLock;** *ikke* bruk synchronized.

Sensorveiledning:

***Låsing**: I denne oppgaven skal studenten vise at hen kan programmere en monitor med to metoder og en betingelse. Monitoren må enten inneholde en teller eller en beholder som kan ta vare på referanser til Fly-objekter, f.eks. en ArrayList, for Fly som venter. Låsen og Condition-variabelen må være deklarert på riktig måte og alle metodene må låses og låses opp med låsen. Studentene har sett monitorer svært like denne i løpet av semesteret, så dette bør være rimelig riktig for å få god uttelling. Metoden hentStartTillatelse må vente korrekt på starttillatelse.*

5 b) 7 poeng

- ⇨ Skriv trådklassen **Flygeleder**, som beskrevet i 5 a).

Sensorveiledning: I denne oppgaven skal studenten vise at hen kan skrive en trådklasse. Tråden må ha en run-metode, som har en løkke hvor Flygelederen kaller sjekkAvganger() og venter 1 minutt.

5 c) 7 poeng

- ⇨ Skriv trådklassen **Pilot**, som beskrevet i 5 a). Parameter til konstruktøren til Pilot er en referanse til objekt. Antag at en pilot kun har ETT fly.

Sensorveiledning: I denne oppgaven skal studenten vise at hen kan skrive en trådklasse. Tråden må ha en run-metode, hvor Pilot kaller hentStartTillatelse().

5 d) 12 poeng

Flygelederen bruker «busy waiting» og kaller **sjekkAvganger** hvert minutt.
Modifiser **sjekkAvganger** så HVIS det ikke er et fly som venter på starttillatelse, så skal Flygelederen vente på en Condition **avventFly**. Denne signaleres så av Piloter når de gjerne vil ha starttillatelse.
Modifiser **hentStartTillatelse** så Piloterne signalerer flygelederen, som så gir tillatelse.

⇨ Omskriv **sjekkAvganger** og **hentStartTillatelse** som beskrevet ovenfor.

Sensorveiledning:

I denne oppgaven skal studenten vise at hen kan konstruere og programmere en toveis synkronisering mellom tråder. Flygelederen må vente på en betingelse og skal ved signalering gi EN av de ventende pilotene rett til å starte med å signalisere tråden, og deretter returnere. Pilotene skal signalisere flygelederen hvis det er en flygeleder som venter og så selv vente på starttillatelse.

Monitoren må inneholde en teller eller en beholder som kan ta vare på referanser til Fly objekter, f.eks. en ArrayList, for Fly som venter. Låsen og Condition-variabelen må være deklarert på riktig måte og alle metodene må låses og låses opp med låsen. Studentene har sett monitorer svært like denne i løpet av semesteret, så dette bør være rimelig riktig for å få god uttelling. Metoden hentStartTillatelse må vente korrekt på starttillatelse.