Øvingsforelesning 2

Læringsmål:

- Bli kjent med Java-syntaksen
- Bli kjent med objektorientert tankegang

Oppgaver

Oppgave 1

I den tilhørende koden til denne forelesningen finnes det uferdig kode til en **Book**-klasse. Denne klassen er en enkel representasjon av en bok som holder styr på tittelen til boken og hvor mange sider den består av.

a)

Implementer kode for konstruktøren og getter- og setter-metodene slik at disse lagrer og gir ut korrekt informasjon.

For å teste at koden fungerer kan main-metoden kjøres. Denne lager en bok, prøver å sette en ny tittel og et nytt sideantall, samt henter og skriver ut tittel og antall sider. Ved korrekt implementasjon av **Book**-klassen, vil main-metoden skrive ut det følgende:

```
The book "Big Java" has 100 pages.
The book "Introduction to Algorithms" has 718 pages.
forelesning2.kode.Book@e9e54c2
```

Merk: det heksadeismale tallet etter @ på siste linje, kan variere

b)

Siste linje i main-metoden prøver å skrive ut **Book**-objektet, men det som skrives ut er ikke veldig nyttig. Dette fordi det ikke finnes noen `toString`-metode i klassen.

Lag en beskrivende toString-metode for Book-klassen.

Oppgave 2

I denne oppgaven skal vi skrive kode for representere biler.

a)

Det første vi må gjøre er å finne ut hva vi trenger å holde styr på for å representere en bil, altså hvilke variabler vi trenger. Gode kandidater er; merke, modell, produksjonsår, antall kilometer kjørt, vekt og registreringsnummer.

Lage en Car-klasse med variablene over og en konstruktør for å sette disse

b)

For å kunne ha noen nytte av **Car**-klassen må det være mulig å hente ut informasjon fra **Car**-objekter.

Lag getter-metoder for alle variablene i Car-klassen

c)

Flere av variablene til **Car**-klassen vil aldri endre seg (merke, modell og produksjonsår), og det gir dermed ikke mening å lage setter-metoder for disse. Andre variable vil derimot kunne endre seg (antall kilometer kjørt, vekt og registreringsnummer), og bør derfor ha en form for setter-metoder.

Skriv setter-metoder for vekt og registreringsnummer variablene

d)

Det gir ikke mening å la det være mulig å sette kilometerstanden til en bil, da denne bare skal kunne øke. I steden for bør det finnes en metode som øker kilometerstanden med et gitt antall kilometer.

Lag en metode som øker kilometerstanden med et gitt antall kilometer. Hvis et negativt antall kilometer blir gitt inn, skal ikke kilometerstanden endre seg.

e)

En bil står ikke bare stille. Hele poenget med en bil er å kunne kjøre rundt. Vi ønsker å kunne representere om en gitt bil står stille eller om den er i bevegelse. Dette kan vi for eksempel gjøre med å ta vare på farten til en bil. Når man lager en ny bil står denne som regel stille, så farten kan settes til 0 til å starte med. For å endre på farten kan man for eksempel lage metoder for akselerasjon og bremsing.

Lag en variabel for å holde styr på farten til en bil og en getter-metode for denne. Lag også metoder for akselerering og bremsing som tar inn hvor mye farten henholdsvis øker eller minskes. Ingen av disse metodene skal endre farten hvis den gitte endringen er negativ.

Pass på at farten ikke kan bli mindre enn 0 ved bremsing.

f)

I **d)** lagde vi en metode som økte kilometerstanden med et gitt antall kilometer. Med den metoden må vi selv regne ut hvor mange kilometer bil har kjørt, noe som er litt tungvindt. Vi ønsker derfor å lage en metode som regner ut dette for oss.

Lag en metode som tar inn antall minutter det kjøres i, og som legger til kilometerstanden den avstanden som blir kjørt (altså den nåværende hastigheten til bilen ganget med antall minutter det kjøres i).

g)

Vår nåværende kode støtter kun kjøring fremover, men en bil kan også rygge. En enkel måte å representere dette på er å bruke negativ hastighet for rygging og positiv hastighet for vanlig kjøring. Dette er derimot ikke direkte intuitivt med hvordan våre akselerere- og bremsemetoder fungerer, da dette vil gjøre at vi må bruke bremse-metoden for å rygge. Vi ønsker derfor heller å holde styr på om vi rygger ved hjelp av en egen variabel. Når man rygger, går man fra å kjøre fremover til å kjøre bakover, så vi kan anta at bilen må stå stille for at det er mulig å skifte mellom rygging eller ikke.

Lag en variable som indikerer om bilen rygger, og tilhørende getter- og setter-metoder.

h)

Slik koden fungerer nå er det ingen begrensinger på hvor raskt bilen kan kjøre. Dette er ganske urealistisk så vi ønsker å sette en maksgrense på hvor raskt en bil kan kjøre. I tillegg er som regel makshastighet forskjellig for vanlig kjøring og rygging.

Utvid Car-klassen slik at vi tar vare på makshastighet for vanlig kjøring og rygging, og at det ikke er mulig å akselerere over denne. Velg makshastigheter selv.

i)

Vi begynner å få ganske mange variabler for en bil og skal vi lage mange biler, så kan det være tungvindt å måtte gi inn alle disse variablene per bil. Spesielt, hvis man skulle finne ut at makshastigheten til en gitt model er litt lavere eller høyere enn tidligere antatt, er det mye jobb å skulle endre på alle bilene individuelt. Vi ønsker derfor å skille ut en del av disse variablene i en egen klasse for biltypen. Det vil si variablene for merke, modell og de to makshastighetene. En bil vil kun ta vare på en *instans* av denne klassen som vi skiller ut, og hente dataen fra instansen når den trenger disse.

Lag en klasse for biltype og endre Car-klassen til å bruke denne som et felt

j)

En bil eies som regel av en person, og det kan være greit å kunne finne ut hvem som eier bilen fra **Car**-klassen.

Lag en Person klasse som tar vare på detaljer om en person, og utvid Car-klassen til å bruke denne for å ta vare på informasjon om eieren av bilen.