INF1000 - Obligatorisk innlevering 5

Frist: 2. Oktober kl 22:00

Temaer denne uka: *Klasser og objekter*.

I denne obligen skal du som nytt tema jobbe med klasser og objekter, i tillegg til å bruke det du tidligere har lært om blant annet metoder og arrayer. Denne obligen er større i omfang enn de foregående obligene, så regn med at du må bruke noe lenger tid. Om det er noe du ikke har fått til, husk å spesifisere dette i README-filen.

Oppgave 5.1)

Tema: *Klasser og objekter*
Filnavn: `Oppgave51.java, Bil.java`

Du skal skrive en klasse *Bil* som skal modellere kjøringen av biler. Biler kan ikke kjøre dersom de ikke har nok bensin. Hver kilometer en bil kjøres, legges til den totale kilometerstanden.

a) Gitt følgende grensesnitt, fyll ut metodene og bruk fornuftige variabler i klassen for å få metodene til å returnere korrekte verdier.

```
class Bil {
    // Nok bensin? Kjor "km" kilometer.
    public boolean kjorTur(int km) { }

    // Fyll tanken, dersom det er plass til spesifisert mengde bensin.
    public boolean fyllTank(double liter) { }

    // Hent maksimal kjoredistanse, gitt bensinmengde på tanken.
    public double hentMaksDistanse() { }

    // Hent bilens totale kilometerstand.
    public int hentKilometerstand() { }

}
```

Vi antar her at alle biler kjøpes nye (kilometerstand 0), med full tank. Det er nødvendig å spesifisere tankstørrelse og bensinforbruk pr. kilometer ved opprettelse av en ny bil. I klassen vil du trenge instansvariablene `kilometerstand`, `tankstorrelse` og `bensinforbruk`. Det vil også være lurt å ha en egen variabel `tank` for å holde på antall liter i tanken til enhver tid.

Dersom noe skulle gå galt, for eksempel at du forsøker å kjøre lenger enn du har bensin til, skal metodene gi tilbakemelding om at dette ikke går til terminalen.

b) Skriv et program som tester klassen Bil. Lag et eller flere objekter av Bil og skriv flere tester av alle metodene der du sjekker at de reagerer som de skal for forskjellige typer input. Husk at en del av testingen bør være kall på metodene med parametere som gir feilmelding. f.eks. fyllTank(99999.99); Gir feilmeldingene dine mening?

Synes du denne oppgaven var vanskelig? Se øvingsoppgave 5.01.

Oppgave 5.2)

Tema: *Teorioppgave*
Filnavn: `Teori.txt`

Gi korte svar på spørsmålene under:

a) Hva er grensesnittet til en klasse? Hvordan skiller det seg fra implementasjonen av en klasse?

b) Hva er en instansmetode og hvordan skiller den seg fra en statisk metode (slike metoder vi har jobbet med tidligere)? Forklar dette ved å vise til både konkrete eksempler på en instansmetode og en statisk metode.

Oppgave 5.3-5.4 (gir 2 poeng!)

Tema: *Objektorientering*
Filnavn: `Oppgave54.java, Isbod.java`

Det er sommer, det er varmt, og folket vil ha is! Du skal lage et enkelt system som skal holde styr på ansatte for sjefen i en isbod. Å være ansatt i isbod er meget sesongbetinget, og hvor mange ansatte som trengs varierer fra uke til uke. Vi antar her at de som har vært ansatt kortest (lavest ansiennitet), vil bli avskjediget først, dersom noen må gå.

a) Skriv en klasse `Isbod`. Lag en privat String-array med ti plasser i klassen. Denne skal inneholde navn på de ansatte. Lag også en heltallsvariabel for å lagre antall ansatte.

b) Skriv en instansmetode `public void ansett(String navn)`, som registrerer en ny ansatt. Navnet på den nyansatte skal lagres på første ledige plass i arrayen (har man 5 ansatte, skal neste ansatte komme på plass 5 i arrayen). Husk at du må sjekke at det ikke alt er 10 ansatte – da kan du ikke ansette flere!

c) Skriv en instansmetode `public void giSistemannSparken()`, som gir den som sist ble ansatt sparken. Programmet skal da gi en utskrift med navnet på den som ble ansatt, fjerne vedkomne fra arrayen. Husk å senke antallet ansatte med én.

d) Skriv en instansmetode `public void printAlleAnsatte()`, som skriver ut alle ansatte i rekkefølge, der den første har lengst ansiennitet, og den siste har kortest. Til denne oppgaven skal du bruke en for-løkke.

e) Lag en klasse `Oppgave54`, og opprett en main-metode. Lag et Isbod-objekt. Ansett tre personer. Skriv ut alle ansatte, og sjekk at korrekte navn kommer ut. Spark så den siste, og skriv ut på nytt. Legg deretter til en ny ansatt. Sjekk at utskrift nok en gang blir korrekt. Se vedlegg A for kjøreeksempel.

f) FRIVILLIG. Lag en konstruktør til Isbod som gjør at du kan spesifisere under opprettelsen av et Isbod-objekt hvor mange ansatte de har plass til. For å teste, prøv å lag flere objekter av Isbod med plass til ulikt antall ansatte, og kall på `ansett()` helt til de har nok medarbeidere i alle isbodene. Sjekk at dette stemmer overens med tallet du sendte inn.

Du har nå implementert en konstruksjon som kalles en "stack". Dette er en datastruktur som brukes mye i "det virkelige programmeringsliv", og som du nok kommer til å jobbe mye med som programmerer!

Oppgave 5.5)

Tema: *Egen oppgave*
Filnavn: `MinOppgave5.java`

a) Skriv en egen oppgavetekst som handler om klasser og objekter.

b) Løs oppgaven du skrev i a). Løsningen må inneholde minst én ekstra klasse i tillegg til **MinOppgave5**.

Husk at du skal levere både løsningen og oppgaveteksten du skrev i a).

Vedlegg A - kjøreeksempler til oppgave 5.3 - 5.4

Kallene:

```
Isbod minIsbod = new Isbod();
minIsbod.ansett("Bjarte");
minIsbod.ansett("Steinar");
minIsbod.ansett("Tore");
minIsbod.printAlleAnsatte();
minIsbod.giSisteMannSparken();
minIsbod.ansett("Helga");
minIsbod.printAlleAnsatte();
```

skal gi lignende følgende utskrift:

```
> Ansatte Bjarte!
> Ansatte Steinar!
> Ansatte Tore!

> Printer alle ansatte:
> - Bjarte
> - Steinar
> - Tore

> Gir Tore sparken!

> Ansatte Helga!

> Printer alle ansatte:
> - Bjarte
> - Steinar
> - Helga
```

Krav til innleveringen

1. Klassenavnet og filnavnet skal være identisk.
2. Klassenavn skal skrives med stor forbokstav.
3. Variabelnavn skal ha liten forbokstav.
4. Oppgaven må kunne kompilere og kjøre på IFI sine maskiner.
5. Kun .java-filen skal innleveres.
6. Ikke bruk æ, ø eller å i .java-filene(heller ikke som kommentarer eller utskrift).
7. Filene skal inneholde gode kommentarer som forklarer hva programmet gjør.
8. Programmet skal inneholde gode utskriftssetninger som gjør det enkelt for bruker å forstå.
9. Metodenavn skal skrives med liten forbokstav.
10. Koden skal være riktig indendert. Er du usikker, se Appendix J i Big Java.
11. Hver klasse skal ligge i sin egen .java-fil.

Fremgangsmåte for innleveringer i INF1000

1. Lag en fil som heter README.txt. Følgende spørsmål skal være besvart i filen:
   (a) Hvordan synes du innleveringen var? Hva var enkelt og hva var vanskelig?
   (b) Hvor lang tid (ca) brukte du på innleveringen?
   (c) Var det noen oppgaver du ikke fikk til? Hvis ja:
   - Hvilke(n) oppgave er det som ikke fungerer i innleveringen?
   - Hvorfor tror du at oppgaven ikke fungerer?
   - Hva ville du gjort for å få oppgaven til å fungere hvis du hadde mer tid?
2. Logg inn på Devilry.
3. Lever de 8 .java-filene, samt teori.txt, README.txt og filene du bruker i den egne oppgaven din i *samme innlevering*.
4. Husk å trykke lever og sjekk deretter at innleveringen din er komplett.

Den obligatoriske innleveringen er minimum av hva du bør ha programmert i løpet av en uke. Du finner flere oppgaver for denne uken her og flere utfordringsoppgaver her.